

# Pratique du



par GDX

Version 2024-05

Basé sur mon expérience et  
de nombreuses documentations :

Clefs pour MSX,

Le livre du MSX,

Le livre du disque MSX,

Pratique du MSX2,

MSX-DOS Version 2,

MSX Datapack vol. 1 à 3 de ASCII,

MSX2 Technical Hand Book,

MSX-Turbo R Technical Hand Book,

et diverses informations trouvées sur Internet.

*Spécial merci à ericb59, jipe et aoineko pour leur participation*

# Index

<b>1 Le standard MSX.....</b>	<b>6</b>
1.1 Introduction.....	6
1.2 Utiliser ce document.....	7
1.3 Configurations minimales.....	8
1.4 Conseils aux développeurs de logiciels en langage machine.....	10
1.5 Et maintenant.....	11
<b>2 Le Z80 et le R800.....</b>	<b>12</b>
2.1 Structure du Z80.....	12
2.2 Interruptions.....	12
Mode 0.....	12
Mode 1.....	13
Mode 2.....	13
2.3 Registres du Z80 et du R800.....	13
2.4 Format des données.....	17
2.5 Temps d'exécution d'une instruction.....	18
2.6 Le jeu d'instructions du Z80 et du R800.....	19
Conventions d'écriture.....	19
Groupe 1 : Transferts de données.....	20
Groupe 2 : Echanges de contenu de registre.....	41
Groupe 3 : Arithmétique.....	43
Groupe 4 : Opérations logiques.....	65
Groupe 5 : Manipulation de bit(s).....	72
Groupe 6 : Comparaisons de données et test de bit(s).....	91
Groupe 7 : Appel à une routine, sauts et pile.....	97
Groupe 8 : Entrées/Sorties.....	107
Groupe 9 : Interruptions et temps d'attente.....	114
<b>3 Les Slot et le Memory Mapper.....</b>	<b>117</b>
3.1 Carte de la mémoire principale sous Basic.....	117
3.2 Comment dépasser la limite des 64 Ko.....	118
3.3 Qu'est-ce qu'un Slot ?.....	118
3.4 Utiliser les Slot.....	121
3.5 Le Memory Mapper.....	125
<b>4 La ROM principale et la ROM auxiliaire du système.....</b>	<b>127</b>
4.1 Information système.....	127
4.2 Introduction au BIOS.....	130
4.3 Table des sauts du Bios de la ROM principale (Main-ROM).....	130
4.4 Routines de l'interpréteur Basic.....	177
Routines « Math-pack ».....	177
Routines générales de l'interpréteur.....	191
4.5 Table des sauts du Bios de la Sub-ROM.....	195
<b>5 Les variables et zones de travail du système.....</b>	<b>228</b>
5.1 Introduction aux variables système.....	228
5.2 Liste des variables et des zones de travail système.....	228
5.3 Quelques exemples d'utilisation des variables système.....	245
<b>6 Les Hooks du système.....</b>	<b>246</b>
6.1 Liste des Hooks.....	247
<b>7 Le processeur vidéo (VDP).....</b>	<b>263</b>
7.1 Avertissement.....	263
7.2 Introduction au processeur vidéo.....	263
7.3 Fonctionnement du VDP.....	263
7.4 Comment accéder au VDP.....	264
Ecrire dans un registre de contrôle (0 à 23, 25 et 32 à 46).....	266
Ecrire dans un registre de la palette des couleurs.....	268
Lire un registre de statut du VDP.....	269
Lecture et écriture dans la mémoire vidéo.....	270
7.5 Les registres de statut du VDP.....	273
7.6 Les registres de contrôle du processeur video.....	275

Registres de mode de fonctionnement (0, 1, 8, 9 et 25).....	275
Registres des tables en VRAM (2, 3, 10, 4, 5, 11 et 6).....	280
Registres de réglage des couleurs de texte et de fond, et du clignotement (7, 12 et 13).....	282
Registres d'accès (14, 15, 16 et 17).....	285
Registres d'affichage et interruption ligne (18 à 23).....	287
Registres des commandes graphiques internes du VDP (32 à 46).....	292
<b>7.7 Les commandes internes du VDP.....</b>	<b>294</b>
YMMM (Y Move to Memory from Memory).....	299
HMMM (High Speed Move to Memory from Memory).....	301
HMMV (High speed Move to Memory from VDP).....	303
LMMC (Logical Move to Memory from CPU).....	305
LMCM (Logical Move to CPU from Memory).....	307
LMMM (Logical Move to Memory from Memory).....	309
LMMV (Logical Move to Memory from VDP).....	311
LINE (draw a LINE).....	313
SRCH (SeaRCH for color).....	315
PSET (Point SET).....	317
POINT (Is POINT set?).....	319
<b>7.8 Les différents modes d'affichage.....</b>	<b>321</b>
SCREEN 0 à 40 colonnes (mode texte 1).....	321
SCREEN 0 à 80 colonnes (mode texte 2).....	325
SCREEN 1 (mode graphique 1).....	329
SCREEN 2 (mode graphique 2).....	333
SCREEN 3 (mode multicolore).....	338
SCREEN 4 (mode graphique 3).....	342
SCREEN 5 (mode graphique 4).....	344
SCREEN 6 (mode graphique 5).....	347
SCREEN 7 (mode graphique 6).....	350
SCREEN 8 (mode graphique 7 avec couleurs indexées).....	353
SCREEN 9 (mode graphique 4 ou 5).....	356
SCREEN 10 et 11 (mode graphique 7 en YJK et RGB).....	356
SCREEN 12 (mode graphique 7 en YJK).....	358
<b>7.9 Les Sprite et leur fonctionnement.....</b>	<b>360</b>
<b>7.10 A propos de la souris et du crayon optique.....</b>	<b>370</b>
<b>8 Le processeur sonore PSG.....</b>	<b>371</b>
8.1 Caractéristiques.....	371
8.2 Descriptions du PSG et accès aux registres.....	371
8.3 Registres de contrôle de fréquence (Registres 0~5).....	372
8.4 Registre de contrôle de la fréquence de bruit blanc (Registre 6).....	374
8.5 Registre de contrôle des voix et des ports d'E/S du PSG (Registre 7).....	374
8.6 Registres de contrôle de l'amplitude et du volume (Registres 8~10).....	374
8.7 Registres de contrôle la forme et de la période de l'enveloppe (Registres 11~13).....	375
8.8 Registres des ports parallèles d'entrées / sorties du PSG (Registres 14 et 15).....	376
<b>9 Le MSX-Music.....</b>	<b>384</b>
9.1 Caractéristiques.....	384
9.2 Méthode de détection du Slot du MSX-Music.....	385
9.3 Registres de l'OPLL.....	386
9.4 Le FM Bios.....	388
Table des sauts du FM Bios.....	388
9.5 Format des données d'une musique FM.....	391
L'entête.....	391
Format des données pour la mélodie.....	392
Format des données pour les sons des instruments.....	394
<b>10 Le MSX-Audio.....</b>	<b>395</b>
10.1 Caractéristiques.....	395
10.2 Mappage de la mémoire du MSX-AUSIO.....	396
10.3 Le MBIOS.....	396
Zone de travail du MBIOS.....	397
Description d'un MIDB.....	398

Description d'un CHDB.....	399
Liste des routines du MBIOS.....	400
10.4 Les registres du Y8950.....	406
Les registres de contrôle.....	406
<b>11 Le MSX-MIDI.....</b>	<b>410</b>
11.1 Accéder au MSX-MIDI.....	410
Accéder au MSX-MIDI interne.....	411
Accéder au MSX-MIDI externe.....	413
11.2 Les interruptions du MSX-MIDI.....	415
11.3 Programmation du MSX-MIDI.....	416
Méthode pour distinguer le type interne ou externe.....	416
Temps d'attente entre l'envoi d'une commande et du mode de registre.....	418
Initialisation du MSX-MIDI.....	419
<b>12 Le MSX-JE.....</b>	<b>420</b>
<b>13 Le système des disques et MSX-DOS.....</b>	<b>421</b>
13.1 Le MSX-DOS 1 et le MSX-DOS 2.....	421
13.2 Format d'un disque.....	423
13.3 Comment démarrer automatiquement un logiciel sur disque.....	425
13.4 Carte de la mémoire principale sous MSX-DOS.....	426
13.5 Base du MSX-DOS 1 et 2 (System Scratch Area).....	427
13.6 Variables d'environnement.....	429
13.7 Routines de la Disk-ROM v1.XX.....	430
Table des sauts du pilote des disques.....	430
Routines de base.....	438
Routines des fonctions BDOS.....	442
13.8 Fonctions BDOS.....	448
Le FCB (File Control Block).....	448
Les métacaractères (Wildcard).....	450
Le gestionnaire de fichier (File Handle).....	450
Le FIB (File Information Block).....	450
Attributs de fichier.....	451
Liste des fonctions BDOS du MSX-DOS 1 et 2.....	451
13.9 Traiter les erreurs du MSX-DOS 1 dans vos programmes.....	502
13.10 Codes d'erreur des fonctions du MSX-DOS 2.....	505
13.11 Zone fixe de travail, Hooks et des variables du MSX-DOS1 et du Disk-BASIC.....	508
13.12 Zone fixe de travail, Hooks et des variables du MSX-DOS2 et du Disk-BASIC 2.....	515
13.13 Le Bios étendu.....	524
<b>14 Applications types.....</b>	<b>541</b>
14.1 Revenir à l'interpréteur Basic.....	541
14.2 Quel type de MSX ?.....	542
14.3 Afficher un texte en assembleur.....	542
14.4 Système a disquettes ou pas ?.....	543
14.5 Faire de la musique en assembleur.....	543
14.6 Passage de paramètres du Basic au langage machine.....	544
14.7 Ajouter une instruction au Basic avec CMD ou IPL.....	548
14.8 Les codes de <i>contrôle</i> [CTRL].....	552
14.9 Les codes d'échappement [ESC].....	552
14.10 Afficher les caractères étendus.....	554
14.11 Détourner le Reset.....	558
14.12 Ajouter des mots clés à l'instruction CALL du Basic.....	559
14.13 Manipuler la souris.....	563
14.14 Développer un programme en cartouche.....	564
14.15 Trouver le Slot de la RAM depuis une ROM.....	571
<b>Annexes.....</b>	<b>576</b>
A – Liste des labels par ordre alphabétique.....	576
B – Les registres du VDP.....	585
C – Les cartes de la mémoire vidéo.....	589
D – Codes des caractères MSX.....	592
E – Exemples de matrices de clavier.....	597

F – Codes d'erreur du Basic et du disque Basic.....	600
G – Les ports d'entrée/sortie.....	602
<b>Lexique.....</b>	<b>611</b>
- Bruit blanc.....	611
- Chaîne ASCII.....	611
- DTA (Disk Transfer Area).....	611
- Main-RAM.....	611
- Mapper.....	611
- Megarom.....	611
- MSX-Engine.....	611
- Pattern.....	611
- PPI (Programmable Port Interface).....	611
- PSG (Programmable Sound Generator).....	611
- RAM (Random Access Memory).....	612
- ROM (Read Only Memory).....	612
- RTC (Real Time Clock).....	612
- Sprite.....	612
- Slot.....	612
- Vblank.....	612
- VRAM (Video Random Access Memory).....	612
- VDP (Video Display processor).....	612

# 1 Le standard MSX

## 1.1 Introduction

Le standard MSX a été créé par ASCII avec l'aide de Microsoft pour grand public. Il 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 des modèles les plus vendus :

- 128/256 Ko de mémoire vive extensible à 4 Mo par Slot libre
- 128 Ko de mémoire vidéo extensible à 192 Ko
- 48 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 et mémoire CMOS comprenant un système de mots de passe alimentée par pile
- 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

Par la suite, le MSX2+ est sortie au Japon avec quelques évolutions mais tous les modèles proposés ont moins de mémoire que la plupart des MSX2 les plus récents. Puis le MSX turbo R dont la principale évolution est le premier pas franchi vers un CPU 16 bit en interne.

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 document

Ce document a été conçu pour les fans de la programmation retro. Il aspire, dans un premier temps, à apprendre à tout programmeur ayant une bonne connaissance du Basic et quelques notions sur le Z80, afin de tirer le maximum de son ordinateur, qu'il s'agisse d'un MSX1 ou d'un 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 officiellement 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 ou peut utilisée, j'ai ajouté le terme anglais entre parenthèses. Il y a un lexique pour les termes spécifiques à l'informatique ou au MSX qui sont employés dans ce document.

De plus, suivant la vieille maxime des informaticiens « rien ne vaut la pratique », beaucoup d'explications se trouvent accompagnées d'un exemple de programme en langage Basic ou assembleur l'illustrant. Lisez l'explication et, 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 très superficiellement dans ce livre. Si vous avez des problèmes avec le CPU, je ne peux que vous conseiller l'acquisition du livre « Programming the 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 de processeur sonore PSG (« Programmable Sound Generator ») ni autres. Tous les manuels livrés avec les différents modèles de MSX ou extensions donnent les renseignements nécessaires à la programmation de ces composants. Voyez tout de même le paragraphe 14.5 « [Faire de la musique en assembleur](#) », page 543.

Ce manuel ne comporte que très peu d'information intrinsèque sur les différents matériels (« hardware »). Il aborde les différents éléments qu'au niveau logiciel (« software »). Il s'adresse donc avant tout aux programmeurs et non aux électroniciens.

### Retrouver un renseignement rapidement :

Lorsque vous maîtrisez l'application qui vous intéresse, vous pouvez éviter tout ce qui est note, remarque et exemple. 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 un annexe 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. Les débutants ne sont pas oubliés.

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 incluant le Basic v.1.xx)
- 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 à 3 voix sur 8 octaves avec générateur de bruit. Celui si est appelé « PSG » (Programmable Sound Generator).
- 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 (pour manette de jeu, souris, molettes, tablette tactile, etc).

#### **MSX2 :**

- Un microprocesseur central (CPU) Z80A ou compatible, 8 bits, cadencé à 3,579545Mhz.
- 64Ko de RAM.
- 48Ko de ROM (Main-ROM incluant le Basic v.2.xx 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.
- Horloge en temps réel compatible (puce RP5C01).
- 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 (pour 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 incluant le Basic v.3.xx, Sub-ROM, Disk-ROM et Kanji Driver).
- 128 Ko de ROM, police de Kanji (Kanji ROM niveau 1).
- 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.
- Une interface programmable pour les ports E/S (PPI) compatible avec le 8255 d'Intel.
- Horloge en temps réel compatible (puce RP5C01).
- 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 (pour 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 incluant le Basic v.4.xx, Sub-ROM, Disk-ROM, MSX-JE, Kanji Driver, MSX-DOS2 ROM et MSX-Music).
- 256 Ko de ROM, police de Kanji (Kanji ROM niveau 1 et, 2 en option).
- 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 à jusqu'à 15 Khz.
- Un processeur sonore YM2413 de Yamaha, 9 voix FM ou bien 6 + 5 de rythme. (MSX-Music)
- Une interface programmable pour les ports E/S (PPI) compatible avec le 8255 d'Intel.
- Horloge en temps réel compatible (puce RP5C01).
- 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.
- Un lecteur de disquette 3,5 pouce, double face.
- Différents connecteurs : sortie vidéo (RCA, RGB ou autres), imprimante, 2 ports généraux (pour manette de jeu, souris, Trackball, molettes, tablette graphique, etc).

Note : Le port du magnéto-cassette a été retiré et le Bios n'est plus compatible avec les molettes (Paddle Controller) ASCII.

## 1.4 Conseils aux développeurs de logiciels en langage machine

- 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 ou autres. Il est souvent indispensable de passer par la table des sauts du BIOS. La Main ROM, la Sub-ROM, la ROM Disk et bien d'autres possèdent 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 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 premier port de lecture et celui d'écriture du processeur vidéo. Voir le [chapitre 7](#) page 263 pour plus de précisions à ce sujet.
- Ne pas utiliser la mémoire vive située au dessus de l'adresse indiquée en mémoire à 0F672h (MEMSIZ). Cette adresse est en général 0F168h lorsqu'il n'y a pas de disque installé. Pour un programme sur disquette, disque dur ou carte Flash, cette adresse est plus basse pour faire place aux variables et zones de travail des disques (voir [chapitre 3.1](#) page 117 pour les détails).
- Il existe des différences entre les MSX commercialisés en France et ceux dans les autres pays ; ne serait-ce que le clavier dont il est parfois important de tenir compte. Les adresses mémoire 002Bh et 002Ch donnent des renseignements importants à ce sujet. Voir le [chapitre 3.1](#) page 117. Sur MSX2 ou plus récent, vous trouverez des renseignements complémentaires dans la mémoire CMOS de l'horloge. Voir la routine REDCLK (01F5h), [chapitre 3.3](#) « Table des sauts du Bios en Sub-ROM » page 195.
- Certains programmeurs placent la pile en haut de mémoire avec l'instruction `LD SP, 0000h`. Ceci ne fonctionne pas sur MSX car l'adresse FFFFh est utilisée pour accéder aux registres des Slot secondaires (voir le [chapitre 3](#) à propos des Slot page 117).
- Certains programmeurs partent du principe que la mémoire vive principale libre contient uniquement des 00h à l'allumage. Il va de soi que ce n'est pas toujours le cas et que la RAM peut contenir à peu près n'importe quoi. Elle est même en partie modifiée lors de l'initialisation du système par certains modèles ayant des logiciels internes (« Firmware »).
- 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 14](#) page 543) 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 de contrôle du processeur vidéo, il est préférable de sauvegarder la donnée dans la zone des variables systèmes réservée pour. Ainsi, la donnée pourra être relue à tout moment, ce qui serait impossible autrement. D'ailleurs, le système se base sur ces données pour diverses routines.

## ***1.5 Et maintenant...***

A présent, vous allez vous lancer dans la découverte de tout ce qui constitue le standard MSX : les Slot, le Bios, les variables systèmes, le processeur vidéo et autres. Une bonne dose de patience sera, dans la plupart des cas, appréciable et bénéfique. Et maintenant à vous de jouer... et bonne chance !

## 2 Le Z80 et le R800

Le Z80 est le microprocesseur central. C'est lui qui est chargé de commander toutes les fonctions des ordinateurs MSX. Le Z80 est un processeur dit 8 bits car ses registres et la lecture des données se fait octet par octet.

Les ordinateurs MSX Turbo R possèdent un R800 en supplément qui peut remplacer le Z80 à la demande. La commutation de l'un vers l'autre se fait par logiciel. Le R800 est un Z80 compatible qui traite les opérations internes sur 16 bits au lieu de 4 bits pour le Z80. Il a aussi un jeu d'instruction supplémentaire pour effectuer des multiplications. Il est cadencé à une fréquence deux fois plus rapide que le Z80 et les instructions prennent moins de cycle d'horloge pour leur exécution.

Dans ce document, vous trouverez une description simplifiée de toutes leur instructions ainsi que la structure nécessaire à la programmation en assembleur.

### 2.1 Structure du Z80

Le Z80 a un BUS de données composé de 8 lignes pour envoyer ou recevoir des données aux autres composants (RAM, ROM, processeur vidéo, processeur sonore, etc) par paquet de 8 bits. Il a aussi un BUS d'adresse composé de 16 lignes principalement utilisé pour accéder à la mémoire mais aussi aux registres de Slot secondaires ou autres dispositifs comme certains Mapper par exemple. Il y a aussi un BUS de commande interne. Le BUS d'entrée/Sortie utilise les 8 lignes de poids faible du BUS d'adresse. La commutation est automatique. Le tout est cadencé à 3,579545Mhz.

### 2.2 Interruptions

Le Z80 possède deux broches qui sont des entrées destinées à recevoir un signal extérieur qui provoque l'appel à une routine dédiée. Ces deux broches se nomment  $\overline{\text{NMI}}$  et  $\overline{\text{INT}}$ . L'appel provoqué par  $\overline{\text{INT}}$  est désactivable et ré-activable par programme avec les instructions DI et EI du Z80. Cette broche est généralement reliée au processeur vidéo sur MSX. Ce signal est envoyé, par défaut, cinquante ou soixante fois par seconde à chaque fin d'affichage de l'écran (affichage de l'avant-plan). La broche  $\overline{\text{NMI}}$  provoque un appel à l'adresse 0066h (après avoir terminé l'instruction en cours). Cet appel n'est pas désactivable par programme. Malgré que la routine correspondante est présente dans le Bios et le MSX-DOS, car cette broche est reliée au +5V sur MSX. Vous pouvez donc l'ignorer.

Le Z80 possède trois modes d'interruptions.

#### Mode 0

Dans ce mode, lorsque qu'un signal d'interruption  $\overline{\text{INT}}$  est envoyé au Z80, celui-ci termine l'instruction en cours puis exécute l'instruction dont le premier octet est envoyé sur le bus de données par le périphérique qui envoie la requête d'interruption. Lorsqu'il s'agit d'une instruction multi-octets, les octets suivants sont lus par une séquence de lecture de mémoire normale. Ce mode ne peut être utilisé qu'en ajoutant un dispositif matériel adéquate.

## Mode 1

Dans ce mode, lorsque qu'un signal d'interruption  $\overline{\text{INT}}$  est envoyé au Z80, celui-ci termine l'instruction en cours puis provoque un appel à l'adresse 0038h. Les MSX fonctionnent dans ce mode. La Main-ROM active ce mode au tout début de l'initialisation du MSX.

## Mode 2

Ce mode est appelé le mode d'interruption vectorisé. Lorsque qu'un signal d'interruption  $\overline{\text{INT}}$  est envoyé au Z80, celui-ci termine l'instruction en cours puis provoque un appel à l'adresse formée à l'aide du registre I (octet de poids fort) et de l'octet placé sur le bus de données (octet de poids faible) par le périphérique ayant provoqué l'interruption. Ce mode ne peut être utilisé pleinement qu'en ajoutant un dispositif matériel adéquate. Cependant, il peut être théoriquement utilisé en ignorant la valeur du bus. Cela n'a jamais été fait en pratique.

### 2.3 Registres du Z80 et du R800

Le Z80 ainsi que le R800 a 18 registres sur 8 bits et 4 sur 16 bits. Ces registres sont nommés comme suit.

Registres 8 bits		Registres 16 bits	
A	F	IX	
B	C	IY	
D	E	SP	
H	L	PC	
	I		
	R		
Registres 8 bits auxiliaires		Indicateurs d'interruption	
A'	F'	IFF1	IFF2
B'	C'		
D'	E'		
H'	L'		

Remarquez que les registres 8 bits ont un nom d'une seule lettre, et que ceux de 16 bits ont un nom de 2 lettres. Chacun des registres 8 bits des deux colonnes de gauche du tableau peuvent être associés à l'autre qui se trouvent sur la même ligne pour effectuer certaines manipulations sur 16 bits. Lorsqu'ils sont associés, on les nomme de la façon suivante : AF, BC, DE, HL ou AF', BC', DE', HL'. C'est au développeur de définir quels sont les registres auxiliaires ou pas car, il n'est pas possible de les différencier.

Voyons maintenant leurs utilisations.

- **Le registre A**, appelé « accumulateur », a un rôle prépondérant. C'est sur lui que porteront la plupart des opérations arithmétiques et logiques. Nombre d'instructions ne concernent que l'accumulateur et ne comportent pas le nom du registre. L'association AF est possible. Veillez à l'utiliser avec précaution car le registre F a une fonction particulière.
- **Le registre F** (Flag register) est un registre accessible en écriture qu'avec l'instruction `PUSH AF`. Chacun des bits qui le composent est un indicateur qui, suivant son état, (0 ou 1), renseigne sur le résultat de certaines opérations. Celui-ci est décrit en détail à la page suivante.
- **Les registres B, C, D, E, H et L** sont utilisés pour stocker temporairement des données destinées à être traitées principalement avec l'accumulateur. Le temps d'accès à ces registres est bien inférieur à celui de la mémoire externe, d'où l'intérêt de bien les utiliser. Ils sont équivalents entre eux, à l'exception de B qui peut servir de compteur de boucle et de même lorsqu'il est associé à C. H et L en association (HL), a un rôle important car il est utilisé en tant que double accumulateur (même rôle que A mais sur 16 bits). Il est aussi utilisé comme pointeur d'adresse en adressage indirect.
- **Les registres A', B', C', D', E', F', H', L'** ont exactement les mêmes fonctions que A, B, C, D, E, F, H, L mais les uns ne peuvent être utilisés qu'à la place des autres grâce aux instructions `EX AF`, `AF'` et `EXX`.
- **Les registres IX et IY** servent à indexer des données en mémoire. En général, ils contiennent une adresse, à laquelle on ajoute, ou retranche, un déplacement relatif pour trouver la donnée désirée. Les instructions qui manipulent ces registres, bien que plus lentes, peuvent s'avérer très pratiques et même faire gagner du temps.
- **Le registre I** est le registre qui sert, en mode d'interruption vectorisé, à stocker les huit bits de poids fort de l'adresse d'interruption tandis que le dispositif d'interruption externe fournit les huit bits inférieurs de l'adresse. Cette fonctionnalité permet aux routines d'interruption d'être localisées dynamiquement n'importe où dans la mémoire avec un temps d'accès minimal à la routine. Il n'est pas utilisé par le système MSX.
- **Le registre R** est un registre de rafraîchissement de la mémoire. Le CPU Z80 contient un compteur de rafraîchissement mémoire, permettant d'utiliser les mémoires dynamiques avec la même facilité que les mémoires statiques. Les sept bits de poids faible de ce registre sont automatiquement incrémentés après chaque extraction d'instruction. Le huitième bit garde la valeur qui a été écrite par `LD R, A`. Les données dans le compteur de rafraîchissement sont envoyées sur la partie inférieure du bus d'adresse avec un signal de commande de rafraîchissement pendant que le CPU décode et exécute l'instruction extraite. Ce mode de rafraîchissement est transparent pour le programmeur et ne ralentit pas le fonctionnement du processeur. Le programmeur peut charger le registre R à des fins de test, mais ce registre n'est normalement pas utilisé par le programmeur. Pendant le rafraîchissement, le contenu du registre I est placé sur les huit bits supérieurs du bus d'adresse.
- **Le registre SP** contient le pointeur de pile. Celui-ci indique l'adresse où sont stockées les adresses de retour des routines appelées par l'instruction `CALL` mais aussi celles stockées ou restaurées par `PUSH / POP`. Ces adresses sont stockées par un système d'empilement, c'est à dire que SP est automatiquement décrémenté ou incrémenté de 2 à chaque stockage ou restauration d'adresse dans la « pile ». Il faut toujours veiller à restaurer (désempiler) dans le bon ordre les adresses empilées et à ce que la pile n'écrase pas des données ou un programme.
- **Le registre PC** contient l'adresse de l'instruction à exécuter. Ce double registre est mis à jour par le CPU au début de l'exécution d'une instruction de la façon suivante : PC est incrémenté du nombre d'octet(s) de l'instruction en cours de traitement. Cette incrémentation se fait donc toujours au tout début de l'exécution d'une instruction et ce n'est pas indiquée dans la description de l'opération effectuée par volonté de simplifier.

- **Les registres IFF1 et IFF2** sont des registres internes d'un bit chacun qui servent d'indicateurs pour les interruptions. Lorsque IFF1 est à 0 les interruptions masquables sont désactivés. Au démarrage ou à l'initialisation du CPU, IFF1 et IFF2 sont mis à 0. IFF2 sert à sauvegarder temporairement l'état de IFF1 pendant qu'une interruption non masquable se produit afin de rétablir l'état de IFF1 avec l'instruction RETN puisque IFF1 est mis à 0 lorsqu'une interruption se produit. Notez que les ordinateurs MSX n'utilisent pas les interruptions non masquable.

Les instructions LD A, I et LD A, R copie l'état de IFF2 dans l'indicateur <sup>PF</sup>/<sub>VF</sub>.

## Le registre F

Le registre F est composé de 8 bits dont 6 sont significatifs (en noir dans le tableau). Il est constitué des indicateurs suivants.

		Registre F							
Bits		7	6	5	4	3	2	1	0
Indicateurs		SF	ZF	YF	HF	XF	<sup>PF</sup> / <sub>VF</sub>	NF	CF

Ces indicateurs ont chacun leur rôle. C'est un point très important en langage machine et donc en assembleur. Ils doivent être bien compris, car sur eux reposent toutes les instructions conditionnelles et de calcul.

- **L'indicateur CF (Carry Flag)**

Cet indicateur, appelé Carry, sert de 9ème bit lors des opérations arithmétiques (ADC, ADD, SBC, SUB) ainsi que dans les opérations de décalage (SLA, SRA, SRL, SLA, SRA, SRL) et de rotation (RL, RLA, RLC, RLCA, RR, RRA, RRC, RRCA).

CF est mis à 0 par les opérations logiques (AND, OR, XOR). Ce qui peut être utile car il n'y a pas d'instruction spécifique pour cela, mais seulement SCF (Set Carry Flag) pour le mettre à 1 et CCF (Complement Carry Flag) pour l'inverser.

L'instructions d'ajustement décimal (DAA) met Carry à 1 lorsque les conditions pour effectuer un ajustement décimal sont remplies.

- **Indicateur NF (Add/Subtract Flag)**

NF Indique si la dernière opération était une addition (0) ou une soustraction (1). Cet indicateur est utilisé par l'instruction d'ajustement décimal de l'accumulateur (DAA) pour savoir si l'on vient d'effectuer une addition ou une soustraction. L'ajustement se fait différemment selon que c'est l'un ou l'autre cas.

- **Indicateur <sup>PF</sup>/<sub>VF</sub> (Parity/Overflow)**

Cet indicateur a un double rôle : d'une part, il indique qu'une opération arithmétique a provoqué un débordement. Rappelons qu'un débordement est quand le résultat de l'opération dépasse le nombre de bits alloués, par exemple lors d'un changement accidentel de signe. Dans ce cas, l'indicateur prend le nom de VF, et il est mis à 1 par toutes les opérations arithmétiques ADC, ADD, SBC, SUB, CP, INC et DEC (sauf celles agissant sur des doubles registres).

D'autre part, après les opérations logiques, des décalages ou rotations de bits ne portant pas sur l'accumulateur (RL, RLD, RR, RRD, SLA, SRA, SRL), et les instructions DAA, IN B, (C) et NEG, cet indicateur passe à 1 si le nombre de bits à 1 dans le résultat est pair. Il prend alors le nom de PF. Il indique donc qu'il y a un nombre de bits passé à 1 qui est paire (1), ou impair (0).

De plus, avec des instructions de traitement de blocs (LDD, LDI, LDDR, LDIR, CPD, CPI, CPDR, CPIR), il indique 1 lorsque le registre compteur passe à 0.

- **Indicateur XF**

Cet indicateur n'a pas été documenté par Zilog. Il est censé ne pas être utilisé. Celui-ci prend le même état que celui du bit 3 du résultat d'une opération 8bit.

- **Indicateur HF (Half Carry Flag)**

Cet indicateur est mis à 1 par une retenue interne du quartet faible vers le quartet fort. Il ne sert pas au programmeur, mais est utilisé par l'instruction DAA lorsqu'on veut travailler en DCB. Dans la pratique, vous n'avez pas à vous en soucier.

- **Indicateur YF**

Cet indicateur n'a pas été documenté par Zilog. Il est censé ne pas être utilisé. Celui-ci prend le même état que celui du bit 5 du résultat d'une opération 8bit. Notez que sur Z80, DDA prend en compte l'état du bit après une opération. Et ce même si il a été modifié entre temps par un POP. Ce qui n'est pas le cas sur le R800 ou le Z180.

- **Indicateur ZF (Zero Flag)**

ZF signale que le résultat d'une opération est nul, ou qu'une comparaison a trouvé l'égalité. Attention, à ce moment ZF est mis à 1, il est effacé sinon. ZF=0 indique un résultat non nul (risque de confusion!). De même l'instruction BIT charge dans ZF l'inverse du bit testé (Z=1 si bit =0 et réciproquement).

Les instructions de chargement, à l'opposé d'autres microprocesseurs, n'affectent pas ZF, à l'exception des peu usuels LD A, I et LD A, R. De même, les incrémentations et décrémentations sur double registres n'affectent pas ZF. On devra donc comparer explicitement ces registres avec « 0 » pour savoir s'ils ont été annulés par INC ou DEC.

Les instructions d'entrée/sortie par blocs utilisent ZF pour savoir si le compteur (registre B) passe à 0 (INI, IND, OUTI, OUTD, INDR, OTIR, OTDR, INIR).

En plus des précédentes, ZF est affecté par ADD (sauf double registre), ADC, SUB, SBC, CP, NEG, AND, OR, XOR, ainsi que les décalages et rotation ne portant pas sur l'accumulateur RR, RL, RRC, RLC, RLD, RDD, SLA, SRA, SRL et DAA, IN, BIT, CPI, CPIR, CPD, CPDR.

- **Indicateur SF (Signe Flag)**

Cet indicateur prend la valeur du bit de signe (bit 7) de l'octet du résultat d'une opération 8 bit.

Les instructions de chargement n'affectent pas SF, de même que INC et DEC sur doubles registres. Par contre, il est affecté par ADD, SUB, SBC, ADC, CP, NEG, AND, OR, XOR, INC, DEC, RR, RL, RRD, RRC, RLC, RLD, SLA, SRA, SRL, DAA, IN, CPR, CPIR, CPD, CPRD, LD A, I, LD A, R.



## 2.4 Format des données

Les bits des octets sont toujours positionnés de droite la gauche, du bit de poids le plus faible vers celui de poids le plus fort.

Exemple : L'instruction en assembleur LD A, 7 stocke la valeur 7 comme ci-dessous.

Bits :	7	6	5	4	3	2	1	0
Registre A =	0	0	0	0	1	0	0	1

Les octets d'une valeur sont positionnés de bas en haut, de l'octet de poids le plus faible vers celui de poids le plus fort. Ce qui n'est pas le cas dans les doubles registres.

Exemple : Le programme suivant permet de stocker la valeur 0C520h en mémoire comme ci-dessous.

```
LD HL, 0C520h      ; H = C5h et L = 20h
LD (0C100h), HL
```

0C101h contient :	0C5h
0C100h contient :	020h

## 2.5 Temps d'exécution d'une instruction

Le temps d'exécution d'une instruction est indiqué en nombre de cycle d'horloge (T States). Un cycle s'effectue en 0,27936511484µs sur le Z80 et deux fois moins sur R800 (0,13968255742µs).

Le temps M1 ajouté avant chaque instruction du Z80 est indiqué entre parenthèses et le temps d'attente ajouté en mode R800 n'est pas indiqué car il varie en fonction de divers paramètres.

Le temps d'attente M1 prend un ou, deux cycle d'horloge pour les instructions ayant un code machine commençant par CB, DD, ED, FD, DD CB ou DD FD.

Sur R800, le temps d'attente M1 n'est pas ajouté mais c'est un temps qui varie en fonction des cas suivants.

### Lorsque le programme se trouve en RAM interne :

- Aucun cycle d'horloge n'est à ajouter, sauf si l'instruction « fetch » franchit une limite de 256 octets (« break page »). Dans ce cas, ajoutez 1 cycle.
- Ajoutez 2 cycles d'horloge pour chaque lecture/écriture de mémoire, ou 3 si l'instruction fait les deux, qu'elle soit ou non sur la même page.
- Ajoutez 2 cycles d'horloge pour l'instruction JP.

### Lorsque le programme se trouve en ROM interne :

- Ajoutez 3 cycles d'horloge.

### Lorsque le programme se trouve en mémoire externe (en cartouche) :

- Ajoutez 4 à 5 cycles d'horloge en fonction de l'alignement de l'horloge de bus.

Lors d'accès aux ports d'E/S, ajoutez 6 à 7 cycles d'horloge en fonction de l'alignement de l'horloge du bus. Lors d'accès au VDP (ports 98h-9Bh), le temps d'attente peut atteindre jusqu'à ±54 cycles !

## 2.6 Le jeu d'instructions du Z80 et du R800

Le jeu d'instructions du Z80 est très étendu puisqu'il comporte près de 700 codes d'instruction machine. Le R800 en a quelques-unes en plus pour effectuer des multiplications.

Ce chapitre contient 9 grands groupes d'instructions dont voici le détail :

[Groupe 1 : Transferts de données](#)

[Groupe 2 : Echanges de registre](#)

[Groupe 3 : Arithmétique](#)

[Groupe 4 : Opérations logiques](#)

[Groupe 5 : Manipulation de bit\(s\)](#)

[Groupe 6 : Comparaisons de données et test de bit\(s\)](#)

[Groupe 7 : Sauts, pile, sous-programmes](#)

[Groupe 8 : Entrées/Sorties](#)

[Groupe 9 : Interruptions et Temps d'attente.](#)

### Conventions d'écriture

Pour simplifier la lecture et diminuer la taille de ce document, j'ai classé les instructions semblables par groupe. Les conventions suivantes vous diront comment lire les descriptions de ces instructions. Pour une partie, il s'agit de conventions internes à ce document et, en écrivant vos programmes, vous devez bien sûr spécifier à chaque instruction les registres et les données nécessaires.

- Les noms d'instruction sont les mêmes que ceux utilisés par Zilog sauf pour les instructions de multiplication du R800 qui gardent les noms donnés par ASCII.
- Les registres sont appelés par les lettres utilisés par Zilog.
- **dr** désigne un double registre (donc 16 bits), soit BC, DE, HL, SP. Certaines instructions ne portent que sur quelques-uns de ces doubles-registres.
- **dri** désigne un registre d'index (IX ou IY).
- **sr** désigne un registre 8 bits parmi : A, B, C, D, E, H, L.
- **ri8** désigne la moitié haute ou basse d'un registre d'index (IX ou IY).
- **op** désigne un opérande. Celui-ci peut varier d'une instruction à l'autre.
- **v8** désigne une valeur sur 8 bits.
- **v16** désigne une valeur sur 16 bits.
- **vs** désignera une valeur signée sur 8 bits qui peut donc varier de -128 (80h) à +127 (7Fh).
- **<b>** désigne le numéro d'un bit d'un octet.
- **<c>** désigne une condition sur les indicateurs. Exemple, JR NC, 8000h provoque le saut à l'adresse 8000h (PC = 8000h) si, et seulement si, l'indicateur CF est à 0.

- Un double registre ou une valeur 16 bits entre parenthèses désignent le contenu de la mémoire à l'adresse correspondante. Par exemple : (dr) signifiera le contenu de la mémoire à l'adresse pointée par le double registre dr.
- Les indicateurs du registre F seront appelés par deux lettres.
- Le symbole «  $\wedge$  » entre deux opérandes signifie qu'un ET logique est effectué entre elles.
- Le symbole «  $\vee$  » entre deux opérandes signifie qu'un OU logique est effectué entre elles.
- Le symbole «  $\oplus$  » entre deux opérandes signifie qu'un OU exclusif (XOR) logique est effectué.
- Le symbole « = » signifie que la partie de gauche prend la valeur résultante de l'opération de droite.
- Le symbole «  $\leftrightarrow$  » entre deux opérandes signifie qu'il y a un échange de la valeur des deux opérandes.

## Groupe 1 : Transferts de données

Remarques générales sur ces instructions :

- Les instructions de chargement ne modifient pas les indicateurs (sauf par LD A, I et LD A, R).
- Lorsqu'on charge une valeur dans un registre ou en mémoire, elle remplace la valeur précédente. La valeur source ne change pas.
- Pour des raisons de commodité et de place, j'ai regroupé le plus possible les instructions similaires. J'indique sur quels registres elles agissent car toutes les combinaisons ne sont pas toujours possibles.
- Le déplacement dans le mode d'indexation est normalement fixé à l'écriture du programme. C'est un nombre signé de -128 à +127 qui est codé dans le dernier octet de l'instruction. Une astuce pour faire varier ce déplacement consiste à faire varier cet octet.

```
LD    A, 3
LD    LABEL+2, A
LABEL: LD    D, (IX+5)
```

Ce programme chargera en fait le contenu de IX+3 dans D si votre programme est en RAM. Cette méthode permet d'obtenir un adressage d'indexation avec le déplacement calculé et non pas fixe.

- En adressage direct, du type LD nn, (ad) ou LD (ad), nn, l'adresse ad est codée sur les deux derniers octets de l'instruction, poids faible d'abord. Une astuce du même type que ci-dessus est possible pour modifier cette adresse dans le courant du programme.

## LD A, (dr)

**Rôle :** Charger dans l'accumulateur la donnée se trouvant à l'adresse indiquée par le double-registre.  
dr peut être BC ou DE.

**Opération :** A = donnée à l'adresse indiquée par dr.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD A, (BC)	02	7 (+1 pour M1)	2
LD A, (DE)	12	7 (+1 pour M1)	2

**Exemple :**

avant		après
(C000H) = 03h		(C000H) = 03h
BC = 0C000h	LD A, (BC)	BC = C000h
A = ?		A = 03h

**Remarque :** Voir aussi [LD sr, \(HL\)](#) page 36.

## LD (dr), A

**Rôle :** Charger la valeur de l'accumulateur à l'adresse indiquée par le double-registre.  
dr peut être BC ou DE.

**Opération :** Donnée à l'adresse indiquée par dr = A

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD (BC), A	0A	7 (+1 pour M1)	2
LD (DE), A	1A	7 (+1 pour M1)	2

**Exemple :**

avant		après
BC = 9000h		BC = 9000h
A = 0FFh	LD (BC), A	A = 0FFh
(9000h) = ?		(9000h) = 0FFh

**Remarque :** Voir aussi [LD \(HL\), sr](#) page 35.

## LD A, (v16)

**Rôle :** Charger dans l'accumulateur la donnée se trouvant à l'adresse spécifiée.

v16 est une valeur entre 0 et 65535.

**Opération :** A = la donnée à l'adresse spécifiée

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD A, (v16)	3A vl vh	13 (+1 pour M1)	2

vl = octet de poids faible de v16. vh = octet de poids fort de v16,

**Exemple :**

avant		après
(C000H) = 02h A = ?	LD A, (0C000h)	(C000H) = 02h A = 02h

## LD (v16), A

**Rôle :** Charger la valeur de l'accumulateur à l'adresse spécifiée.

v16 est une valeur entre 0 et 65535.

**Opération :** La donnée à l'adresse spécifiée = A

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD (v16), A	32 vl vh	13 (+1 pour M1)	4

vl = octet de poids faible de v16. vh = octet de poids fort de v16,

**Exemple :**

avant		après
A = 0FFh (9800h) = ?	LD (09800h), A	A = 0FFh (9800h) = 0FFh

**Rôle :** Charger dans l'accumulateur la valeur du registre de vectorisation des interruptions.

**Opération :** A = I

**État des indicateurs du registre F après exécution :**

SF est mis à 1 si le registre I est négatif.

ZF est mis à 1 si le registre I est à zéro.

HF est mis à 0.

$PF/VF = IFF2$ .

NF est mis à 0.

CF ne change pas.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD A, I	ED 57	9 (+2 pour M1)	2

**Remarque :** Cette instruction très technique peut servir à savoir si une interruption est en cours ou pas. Pour cela, certains programmeurs utilisent le programme suivant.

```
LD A, I
DI
PUSH AF    ; Préserve l'indicateur PF/VF

; Votre routine

POP AF     ; Restore l'indicateur PF/VF
RET PO
EI
RET
```

Cependant, Il n'est pas fiable à 100 % dans le cas, entre autres, où des interruptions sont produites par d'autres matériels que le VDP. Veuillez plutôt utiliser le programme plus technique qui suit.

```
; Ce programme ne fonctionne que si PC est supérieur à 00FFh
; Il tient compte des interruptions qui se produiraient pendant
l'exécution
; du programme en testant si (SP-1) a été écrasé ou pas.
;
; En entrée, l'indicateur P/N indique si les interruptions sont activées
; Modifie: AF
```

CheckInterruptState:

```
XOR A
PUSH AF    ; Décrémente la pile
POP AF
LD A, I
RET PE     ; Retour si les interruptions sont activée
DEC SP
DEC SP     ; Pour confirmer que les interruptions sont bien coupées
```

```

POP  AF    ; (sp - 1) est écrasé si une interruption s'est produite
SUB   1
SBC   A,A
AND   1     ; (SP - 1) = 0 alors P/N = 1 sinon P/N = 0
RET   PE

```

; Votre routine

```
RET
```

### LD I,A

**Rôle :** Charger dans le registre de vectorisation des interruptions la valeur de l'accumulateur.

**Opération :** I = A

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD I,A	ED 47	9 (+2 pour M1)	2

### LD A,R

**Rôle :** Charger dans l'accumulateur la valeur du registre de rafraîchissement dynamique des mémoires.

**Opération :** A = R

**État des indicateurs du registre F après exécution :**

SF est mis à 1 si le registre R est négatif.  
 ZF est mis à 1 si le registre R est à zéro.  
 HF est mis à 0.  
 $PF/VF = IFF2$ .  
 NF est mis à 0.  
 CF ne change pas.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD A,R	ED 5F	9 (+2 pour M1)	2

**Remarques :**

- LD A,R vous donnera un nombre pseudo-aléatoire, ce qui peut avoir son utilité.
- Même remarque que pour LD A,I.



## LD R,A

**Rôle :** Charger la valeur de l'accumulateur dans le registre de rafraîchissement dynamique des mémoires.

**Opération :** R = A

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD R,A	ED 4F	9 (+2 pour M1)	2

**Remarque :** Vous n'avez pas à vous servir de cette instruction très techniques sur MSX.

## LD sr,sr

**Rôle :** Charger la valeur du registre de droite dans le registre de gauche.

sr peut être le registre A, B, C, D, E, H ou L.

**Opération :** sr = sr

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD B,sr	40+v	4 (+1 pour M1)	1
LD C,sr	48+v	4 (+1 pour M1)	1
LD D,sr	50+v	4 (+1 pour M1)	1
LD E,sr	58+v	4 (+1 pour M1)	1
LD H,sr	60+v	4 (+1 pour M1)	1
LD L,sr	68+v	4 (+1 pour M1)	1
LD A,sr	78+v	4 (+1 pour M1)	1

Ici, sr peut être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

**Exemple :**

avant		après
H = 46h A = ?	LD A,H	H = 46h A = 46h

**Remarque :** Il est possible de spécifier deux fois le même registre mais à quoi ça sert ?

**Rôle :** Charger dans un registre la donnée spécifiée.

sr peut être le registre A, B, C, D, E, H ou L.

v8 est une valeur entre 0 et 255.

**Opération :** sr = v8

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD B, v8	06 v8	7 (+1 pour M1)	2
LD C, v8	0E v8	7 (+1 pour M1)	2
LD D, v8	16 v8	7 (+1 pour M1)	2
LD E, v8	1E v8	7 (+1 pour M1)	2
LD H, v8	26 v8	7 (+1 pour M1)	2
LD L, v8	2E v8	7 (+1 pour M1)	2
LD A, v8	3E v8	7 (+1 pour M1)	2

**Exemple :**

avant		après
B = ?	LD B, 0FCh	B = 0FCh

## LD sr, (dri+vs)

**Rôle :** Charger dans le registre 8 bits spécifié de la donnée se trouvant à l'adresse indiquée par le registre d'indexation + la valeur qui suit.

sr peut être le registre A, B, C, D, E, H ou L.

dri peut être le registre d'indexation IX ou IY.

vs est une valeur entre -128 et +127.

**Opération :** sr = octet lu à l'adresse indiquée par dri + vs

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD B, (IX+vs)	DD 46 vs	19 (+2 pour M1)	5
LD C, (IX+vs)	DD 4E vs	19 (+2 pour M1)	5
LD D, (IX+vs)	DD 56 vs	19 (+2 pour M1)	5
LD E, (IX+vs)	DD 5E vs	19 (+2 pour M1)	5
LD H, (IX+vs)	DD 66 vs	19 (+2 pour M1)	5
LD L, (IX+vs)	DD 6E vs	19 (+2 pour M1)	5
LD A, (IX+vs)	DD 7E vs	19 (+2 pour M1)	5
LD B, (IY+vs)	FD 46 vs	19 (+2 pour M1)	5
LD C, (IY+vs)	FD 4E vs	19 (+2 pour M1)	5
LD D, (IY+vs)	FD 56 vs	19 (+2 pour M1)	5
LD E, (IY+vs)	FD 5E vs	19 (+2 pour M1)	5
LD H, (IY+vs)	FD 66 vs	19 (+2 pour M1)	5
LD L, (IY+vs)	FD 6E vs	19 (+2 pour M1)	5
LD A, (IY+vs)	FD 7E vs	19 (+2 pour M1)	5

## LD (dri+vs), sr

**Rôle :** Stocker la valeur d'un registre 8bits à l'adresse pointée par le registre d'indexation plus la valeur spécifiée.

dri peut être le double registre IX ou IY.

vs est une valeur entre -128 et +127.

sr peut être A, B, C, D, E, H ou L.

**Opération :** Donnée à l'adresse indiquée par dri + vs = sr

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD (IX+vs), sr	DD 70+v vs	19 (+2 pour M1)	5
LD (IY+vs), sr	FD 70+v vs	19 (+2 pour M1)	5

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

**Exemples :**

avant		après
E = 0 IY = 9000h (9007h) = ?	LD (IY+7), E	E = 0 IY = 9000h (9007h) = 00
avant		après
IX = 8089h (8089h) = ? C = 12h	LD (IX+0), C	IX = 8089h (8089h) = 12h C = 12h

**Rôle :** Charger l'octet de poids spécifié du double registre d'indexation dans un registre.

sr peut être un registre parmi : A, B, C, D, E.

**Opération :** Octet de point spécifié du double registre d'indexation = v8

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD B, IXH	DD 45	8 (+2 pour M1)	2
LD C, IXH	DD 4C	8 (+2 pour M1)	2
LD D, IXH	DD 55	8 (+2 pour M1)	2
LD E, IXH	DD 5C	8 (+2 pour M1)	2
LD A, IXH	DD 7C	8 (+2 pour M1)	2
LD B, IXL	DD 46	8 (+2 pour M1)	2
LD C, IXL	DD 4D	8 (+2 pour M1)	2
LD D, IXL	DD 56	8 (+2 pour M1)	2
LD E, IXL	DD 5D	8 (+2 pour M1)	2
LD A, IXL	DD 7D	8 (+2 pour M1)	2
LD B, IYH	FD 45	8 (+2 pour M1)	2
LD C, IYH	FD 4C	8 (+2 pour M1)	2
LD D, IYH	FD 55	8 (+2 pour M1)	2
LD E, IYH	FD 5C	8 (+2 pour M1)	2
LD A, IYH	FD 7C	8 (+2 pour M1)	2
LD B, IYL	FD 46	8 (+2 pour M1)	2
LD C, IYL	FD 4D	8 (+2 pour M1)	2
LD D, IYL	FD 56	8 (+2 pour M1)	2
LD E, IYL	FD 5D	8 (+2 pour M1)	2
LD A, IYL	FD 7D	8 (+2 pour M1)	2

**Exemple :**

avant		après
IX = 46CCh A = ?	LD A, IXH	IX = 46CCh A = 46h

**Remarque :** Ces instructions ne sont pas officielles pour le Z80. Elles ne sont pas documentées par Zilog, ni par ASCII sauf pour le R800.

## LD IXH, sr et LD IXL, sr

**Rôle :** Charger la valeur d'un registre dans l'octet de poids spécifié de IX.

sr peut être un registre parmi : A, B, C, D, E, IXH, IXL (avec H pour bits de poids fort, L pour bits de poids faible).

**Opération :** Octet de point spécifié de IX = v8

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD IXH, sr	DD 60+v v8	11 (+2 pour M1)	3
LD IXL, sr	DD 68+v v8	11 (+2 pour M1)	3

Ici, sr peut-être le registre B, C, D, E, IXH, IXL ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

**Exemple :**

avant		après
IX = CCCCh	LD IXH, 0FCh	IX = FCCCh

**Remarque :** Ces instructions ne sont pas officielles pour le Z80. Elles ne sont pas documentées par Zilog, ni par ASCII sauf pour le R800.

## LD IYH, sr et LD IYL, sr

**Rôle :** Charge la valeur d'un registre dans l'octet de poids spécifié de IY.

sr peut être un registre parmi : A, B, C, D, E, IXH, IXL. (avec H pour bits de poids fort, L pour bits de poids faible)

**Opération :** Octet de point spécifié de IY = v8

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD IYH, sr	FD 60+v v8	11 (+2 pour M1)	3
LD IYL, sr	FD 68+v v8	11 (+2 pour M1)	3

Ici, sr peut-être le registre B, C, D, E, IYH, IYL ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

**Remarque :** Ces instructions ne sont pas officielles pour le Z80. Elles ne sont pas documentées par Zilog, ni par ASCII sauf pour le R800.

**Rôle :** Charger une donnée dans l'octet de point spécifié d'un double registre d'indexation.

dri peut être IX ou IY (avec H pour bits de poids fort, L pour bits de poids faible).

v8 est une valeur entre 0 et 255.

**Opération :** Octet de point spécifié de dri = v8

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD IXH, v8	DD 26 v8	11 (+2 pour M1)	3
LD IXL, v8	DD 2E v8	11 (+2 pour M1)	3
LD IYH, v8	FD 26 v8	11 (+2 pour M1)	3
LD IYL, v8	FD 2E v8	11 (+2 pour M1)	3

**Exemple :**

<u>avant</u>		<u>après</u>
IY = CCCCh	Ld IYH, 0FCh	IY = FCCCh

**Remarque :** Ces instructions ne sont pas officielles pour le Z80. Elles ne sont pas documentées par Zilog, ni par ASCII sauf pour le R800.

## LD HL, (v16) et LD dri, (v16)

**Rôle :** Charger dans le double registre la donnée 16 bits pointées par l'adresse spécifiée.

dri peut être IX ou IY.

v16 est une valeur entre 0 et 65535.

**Opération :** Octet de poids faible du double registre = la valeur à l'adresse v16 ;

Octet de poids fort du double registre = la valeur à l'adresse v16+1

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD HL, (v16)	21 vl vh	16 (+1 pour M1)	5
LD IX, (v16)	DD 2A vl vh	20 (+2 pour M1)	6
LD IY, (v16)	FD 2A vl vh	20 (+2 pour M1)	6

vl = octet de poids faible de v16 et vh = octet de poids fort de v16.

**Exemples :**

avant		après
(9503h) = 5Ch (9502h) = 0ABh HL = ?	LD HL, (8402h)	(9503h) = 5Ch (9502h) = 0ABh HL = 5CABh
avant		après
(8403h) = 4Ch (8402h) = 0AAh IX = ?	LD IX, (8402h)	(8403h) = 4Ch (8402h) = 0AAh IX = 4CAAh



**Rôle :** Stocker la valeur du double registre à l'adresse spécifiée. L'octet à l'adresse spécifiée prend la valeur du registre L et l'octet à l'adresse suivante celle de H.

v16 est une valeur entre 0 et 65535.

dri peut être IX ou IY.

**Opération :** Valeur à l'adresse v16 = octet de poids faible du double registre spécifié ;

Valeur à l'adresse v16+1 = octet de poids fort du double registre spécifié.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD (v16), HL	22 vl vh	16 (+1 pour M1)	5
LD (v16), IX	DD 22 vl vh	20 (+2 pour M1)	6
LD (v16), IY	FD 22 vl vh	20 (+2 pour M1)	6

vl = octet de poids faible de v16. vh = octet de poids fort de v16,

**Exemples :**

avant		après
(FF31h) = ? (FF30h) = ? HL = 018Ch	LD (0FF30H), HL	(FF31h) = 01h (FF30h) = 8Ch HL = 018Ch
avant		après
(E141h) = ? (E140h) = ? IY = 027Bh	LD (0E140H), IY	(E141h) = 02h (E140h) = 7Bh IY = 027Bh

## LD dr, (v16)

**Rôle :** Charge dans le double registre spécifié la donnée 16 bits pointées par l'adresse spécifiée.

dr peut être BC, DE, SP.

v16 est une valeur entre 0 et 65535.

**Opération :** L'octet de poids faible du double registre = l'octet lu à l'adresse v16,

l'octet de poids fort du double registre = l'octet lu à l'adresse v16+1.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD BC, (v16)	ED 4B vl vh	20 (+2 pour M1)	6
LD DE, (v16)	ED 5B vl vh	20 (+2 pour M1)	6
LD SP, (v16)	ED 7B vl vh	20 (+2 pour M1)	6

vl = octet de poids faible de v16 et vh = octet de poids fort de v16,

**Exemple :**

avant		après
(8403h) = 4Bh	LD DE, (8402h)	(8403h) = 4Bh
(8402h) = 0DAh		(8402h) = 0DAh
DE = ?		DE = 4BDAh

**Remarque :** Il y a la même instruction, plus rapide, mais qui agit seulement sur HL.

## LD (v16), dr

**Rôle :** Stocke la valeur du double registre à l'adresse spécifiée. L'octet à l'adresse spécifiée prend la valeur des 8 bits de poids faible du double registre et l'octet à l'adresse suivante celle des 8 bits de poids fort.

v16 est une valeur entre 0 et 65535.

dr peut être BC, DE ou SP.

**Opération :** (v16) = l'octet de poids faible du double registre,  
(v16+1) = l'octet de poids fort du double registre.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD (v16), BC	ED 43 vl vh	20 (+2 pour M1)	6
LD (v16), DE	ED 53 vl vh	20 (+2 pour M1)	6
LD (v16), SP	ED 73 vl vh	20 (+2 pour M1)	6

vl = octet de poids faible de v16 et vh = octet de poids fort de v16,

**Exemple :**

avant		après
(DF31h) = ?		(DF31h) = 01h
(DF30h) = ?	LD (0DF30H), BC	(DF30h) = 8Ch
BC = 018Ch		BC = 018Ch

**Remarque :** Il y a la même instruction, plus rapide, mais qui agit seulement avec HL.

## LD (HL), sr

**Rôle :** Stocker la valeur du registre 8 bits spécifié à l'adresse pointée par HL.

sr peut être A, B, C, D, E, H ou L.

**Opération :** sr = (HL)

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD (HL), sr	70+v	7 (+1 pour M1)	2

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

## LD sr, (HL)

**Rôle :** Stocker dans un registre 8 bits la valeur à l'adresse pointée par HL.

sr peut être A, B, C, D, E, H ou L.

**Opération :** sr = (HL)

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD B, (HL)	46	7 (+1 pour M1)	2
LD C, (HL)	4E	7 (+1 pour M1)	2
LD D, (HL)	56	7 (+1 pour M1)	2
LD E, (HL)	5E	7 (+1 pour M1)	2
LD H, (HL)	66	7 (+1 pour M1)	2
LD L, (HL)	6E	7 (+1 pour M1)	2
LD A, (HL)	7E	7 (+1 pour M1)	2

**Exemple :**

avant		après
(8566h) = 47h HL = 8546h	LD L, (HL)	(8566h) = 47h HL = 8547h

Dans cet exemple, H contient 85H et que L contient 66H.

## LD (HL), v8

**Rôle :** Stocker une valeur 8 bits à l'adresse pointée par HL.

v8 est une valeur entre 0 et 255.

**Opération :** (HL) = v8

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD (HL), v8	36 v8	10 (+1 pour M1)	2

**Exemple :**

avant		après
HL = 8596h (8596h) = ?	LD (HL), 0	HL = 8596h (8596h) = 0

Dans cet exemple, H contient 85H et que L contient 66H.

## LD dr,v16

**Rôle :** Charger une donnée 16 bits dans le double registre spécifiée.

dr peut être BC, DE, HL ou SP.

v16 est une valeur entre 0 et 65535.

**Opération :** dr = v16

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD BC,v16	01 vl vh	10 (+1 pour M1)	3
LD DE,v16	11 vl vh	10 (+1 pour M1)	3
LD HL,v16	21 vl vh	10 (+1 pour M1)	3
LD SP,v16	31 vl vh	10 (+1 pour M1)	3

vl = octet de poids faible de v16 et vh = octet de poids fort de v16,

**Exemple :**

avant		après
BC = ?	LD BC,0A000h	BC = A000h

**Remarque :** Utilisée avec SP, cette instruction permet de changer la pile de place : c'est utile mais... attention au retours de sous-programmes !

## LD dri,v16

**Rôle :** Charger une donnée 16 bits dans le registre d'indexation spécifiée.

dri peut être IX ou IY.

v16 est une valeur entre 0 et 65535.

**Opération :** dri = v16

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD IX,v16	DD 21 vl vh	20 (+2 pour M1)	4
LD IY,v16	FD 21 vl vh	20 (+2 pour M1)	4

vl = octet de poids faible de v16 et vh = octet de poids fort de v16,

**Exemples :**

avant		après
IX = ?	LD IX,0C91Eh	IX = C91Eh

## LD SP,dri

**Rôle :** Charger dans le registre pointeur de pile la valeur d'un registre d'indexation.

dri peut être IX ou IY.

**Opération :** SP = dri

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD SP, IX	DD 21 vl vh	10 (+2 pour M1)	2
LD SP, IY	FD 21 vl vh	10 (+2 pour M1)	2

vl = octet de poids faible de v16 et vh = octet de poids fort de v16,

**Exemple :**

avant		après
SP = ? IX = 8000h	LD SP, IX	SP = 8000h IX = 8000h

**Remarque :** Cette instruction permet de changer la pile de place : c'est utile mais... attention au retours de sous-programmes !

## LD SP,HL

**Rôle :** Charger dans le registre pointeur de pile la valeur du double registre HL.

**Opération :** SP = HL

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LD SP, HL	F9	6 (+1 pour M1)	1

**Exemple :**

avant		après
SP = ? HL = 9000h	LD SP, HL	SP = 9000h HL = 9000h

**Remarque :** Cette instruction permet de changer la pile de place : c'est utile mais... attention au retours de sous-programmes !

**Rôle :** LDD et LDDR servent à transférer un bloc de données du haut vers le bas. Pour cela, le contenu de l'adresse pointée par HL est chargé dans l'adresse pointée par DE puis BC, DE et HL sont décrémentés tous les trois.

**Opération :** (DE) = (HL) puis HL = HL - 1, DE = DE - 1 et BC = BC - 1

L'opération est répétée automatiquement jusqu'à ce que BC = 0 avec LDDR.

**État du registre F après exécution :**

SF ne change pas.

ZF ne change pas.

HF est mis à 0.

<sup>PF</sup>/<sub>VF</sub> est mis à 1 si BC ne passe pas à 0.

NF est mis à 0.

CF ne change pas.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LDD	ED A8	16 (+2 pour M1)	4
LDDR	ED B8	21 (+2 pour M1) si BC ≠ 0 16 (+2 pour M1) si BC = 0	4

**Exemple :**

avant		après
BC = 02A5h	LDD	BC = 02A4h
DE = 5200h		DE = 51FFh
HL = 876Dh		HL = 876Ch
(5200h) = ?		(5200h) = 37h
(87D6h) = 37h		(87D6h) = 37h

**Remarque :** Vous pouvez remplir une zone en mémoire avec la valeur désirée placée au préalable dans HL et en mettant HL-1 dans DE comme adresse de destination.

```
LD    HL, 09000h
LD    (HL), 0
LD    D, H
LD    E, L
DEC   DE
LD    BC, 04FFh
LDIR                      ; Rempli la zone de 09000h à 8500h de 0
```

**Rôle :** LDI et LDIR servent à transférer un bloc de données du bas vers le haut. Pour cela, le contenu de l'adresse pointée par HL est chargé dans l'adresse pointée par DE puis, HL et DE sont incrémentés tandis que BC est décrémenté.

**Opération :** (DE) = (HL) puis  $HL = HL + 1$ ,  $DE = DE + 1$  et  $BC = BC - 1$

L'opération est répétée automatiquement jusqu'à ce que  $BC = 0$  avec LDIR.

**État du registre F après exécution :**

SF ne change pas.

ZF ne change pas.

HF est mis à 0.

$PF/VF$  est mis à 1 si BC ne passe pas à 0.

NF est mis à 0.

CF ne change pas.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
LDI	ED A0	16 (+2 pour M1)	4
LDIR	ED B0	21 (+2 pour M1) si $BC \neq 0$ 16 (+2 pour M1) si $BC = 0$	4

**Remarque :** Vous pouvez remplir une zone en mémoire avec la valeur désirée placée au préalable dans HL et en mettant HL+1 dans DE comme adresse de destination. Voici un petit programme qui fait cela.

```
LD    HL, 09000h
LD    (HL), 255
LD    D, H
LD    E, L
INC   DE
LD    BC, 0FFFh
LDIR                      ; Rempli la zone de 09000h à A000h de 255
```



## Groupe 2 : Echanges de contenu de registre.

Ces instructions sont peu nombreuses. Elles permettent d'utiliser les registres auxiliaires.

### EX AF, AF'

**Rôle :** Échange le contenu de l'accumulateur et du registre d'état avec celui de leurs homologues.

**Opération :** AF ↔ AF'

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
EX AF, AF'	08	4 (+1 pour M1)	1

**Remarque :** Bien entendu, les indicateurs prennent les valeurs avaient des bits de F'.

### EX DE, HL

**Rôle :** Échange le contenu du double registre DE avec celui de HL.

**Opération :** DE ↔ HL

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
EX DE, HL	EB	4 (+1 pour M1)	1

**Exemple :**

avant		après
DE = 4D6Bh HL = 67E2h	EX DE, HL	DE = 67E2h HL = 4D6Bh

**Remarque :** Il n'y a pas d'instruction EX BC, HL. Voici donc un exemple pour faire la même chose en un minimum de temps.

```
push bc
ld c, l
ld b, h
pop hl
```

Autre exemple plus lent.

```
push bc
ex (sp), hl
pop bc
```

## EX (SP),HL et EX (SP),dri

**Rôle :** Échange le contenu du double registre spécifié avec celui du sommet de la pile.

**Opération :** Octet de poids faible du double registre  $\leftrightarrow$  (SP) ;

Octet de poids fort du double registre  $\leftrightarrow$  (SP+1)

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
EX (SP),HL	E3	19 (+1 pour M1)	5
EX (SP),IX	DD E3	23 (+2 pour M1)	6
EX (SP),IY	FD E3	23 (+2 pour M1)	6

**Exemple :**

avant		après
IY = B000h	EX (SP),IY	IY = BCC8h
SP = F390h		SP = F390h
(F391H) = BCh		(F391H) = B0h
(F390H) = C8h		(F390H) = 00h

**Remarques :**

- Attention, après l'exécution de l'une de ces instructions, le prochain désempilement affectera l'ancienne valeur du double registre. Cela peut être utile, mais gare aux retours de sous-programmes si vous avez modifié la pile ! (Un sous-programme doit toujours rendre la pile dans l'état où il l'a trouvée lors de l'appel, à moins que vous voulez qu'il en soit autrement...).
- Ces trois instructions fonctionnent de la même manière. C'est pourquoi je les ai regroupées.

## EXX

**Rôle :** Permute les contenus de BC, DE et HL avec ceux des registres auxiliaires correspondants.

**Opération :** BC  $\leftrightarrow$  BC' ; DE  $\leftrightarrow$  DE' ; HL  $\leftrightarrow$  HL'

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
EXX	D9	4 (+1 pour M1)	1

**Remarque :** EXX et EX AF, AF' sont les seules instructions agissant sur les registres auxiliaires. En fait, celles-ci agissent comme un permutateur : Le jeu de doubles registres auxiliaires est utilisés ou l'autre, jamais les deux à la fois. Cela permet de stocker 6 octets et de les récupérer ensemble avec une seule instruction.

## Groupe 3 : Arithmétique

Ce groupe contient les instructions en rapport avec l'arithmétique (additions, soustractions, multiplications, etc).

### ADC A, op

**Rôle :** L'opérande et l'indicateur CF sont additionnées à l'accumulateur. Attention, cette opération faisant intervenir CF, vous devez donc en connaître le contenu auparavant.

op peut être : A, B, C, D, E, H, L, (HL), (IX+vs), (IY+vs) ou un octet.

**Opération :**  $A = A + op + CF$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est nul.

HF est mis à 1 si l'addition des 3 bits de poids faible provoque une retenue.

<sup>PF</sup>/<sub>VF</sub> mis à 1 lors d'un débordement.

NF est mis à 0.

CF est mis à 1 si l'addition provoque une retenue.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
ADC A, sr	88+v	4 (+1 pour M1)	1
ADC A, v8	CE v8	7 (+1 pour M1)	2
ADC A, (HL)	8E	7 (+1 pour M1)	2
ADC A, (IX+vs)	DD 8E vs	19 (+2 pour M1)	5
ADC A, (IY+vs)	FD 8E vs	19 (+2 pour M1)	5

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

**Exemple :**

avant		après
A = 2Ah	ADD A, C	A = 5Dh
C = 33h		C = 33h
CF = 0		CF = 0

**Rôle :** La valeur de CF et le contenu de la valeur de l'octet de poids spécifié du double registre d'indexation sont ajoutés l'accumulateur.

dri peut être IX ou IY (avec H pour bits de poids fort, L pour bits de poids faible).

**Opération :**  $A = A + \text{la valeur de l'octet de poids spécifié du double registre d'indexation} + CF$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est nul.

HF est mis à 1 si l'addition des 3 bits de poids faible provoque une retenue.

$PF/VF$  mis à 1 lors d'un débordement.

NF est mis à 0.

CF est mis à 1 si l'addition provoque une retenue.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
ADC A, IXH	DD 8C	8 (+2 pour M1)	2
ADC A, IXL	DD 8D	8 (+2 pour M1)	2
ADC A, IYH	FD 8C	8 (+2 pour M1)	2
ADC A, IYL	FD 8D	8 (+2 pour M1)	2

**Remarque :** Ces quatre instructions ne sont pas officielles. Elles ne sont pas documentées par Zilog, ni par ASCII.

**Rôle :** Le contenu du double registre spécifié et l'indicateur CF sont additionnés au double registre HL. Attention, cette opération faisant intervenir CF, vous devez donc en connaître le contenu auparavant. A noter que cette instruction est avec `ADC HL, dr` la seule à effectuer des additions sur 16 bits. Le résultat est placé dans HL.

dr peut être BC, DE, HL ou SP.

**Opération :**  $HL = HL + dr + CF$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est nul.

HF est mis à 1 si l'addition des 11 bits de poids faible provoque une retenue.

<sup>PF</sup>/<sub>VF</sub> mis à 1 lors d'un débordement.

NF est mis à 0.

CF est mis à 1 si l'addition provoque une retenue.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
<code>ADC HL, BC</code>	ED 4A	15 (+2 pour M1)	1
<code>ADC HL, DE</code>	ED 5A	15 (+2 pour M1)	1
<code>ADC HL, HL</code>	ED 6A	15 (+2 pour M1)	1
<code>ADC HL, SP</code>	ED 7A	15 (+2 pour M1)	1

**Exemple :**

avant		après
HL = A35Eh		HL = B5FEh
BC = 12A0h	<code>ADC HL, BC</code>	BC = 12A0h
CF = 0		CF = 0

**Remarque :** L'instruction `ADC HL, HL` peut permettre de multiplier HL par 2 de manière élégante (CF doit bien entendu être mise à 0 auparavant par exemple à l'aide de `OR A`).

## ADD A, op

**Rôle :** Additionne le contenu de l'opérande au contenu de l'accumulateur (sans tenir compte de la retenue à la différence de l'instruction ADC A, op). Le résultat est placé dans l'accumulateur (registre A).

op peut être A, B, C, D, E, H, L, (HL), (vs+IX), (IY+vs) ou un octet.

**Opération :**  $A = A + op$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est nul.

HF est mis à 1 si l'addition des 3 bits de poids faible provoque une retenue.

<sup>PF</sup>/<sub>VF</sub> mis à 1 lors d'un débordement.

NF est mis à 0.

CF est mis à 1 si l'addition provoque une retenue.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
ADD A, sr	80+v	4 (+1 pour M1)	1
ADD A, v8	C6 v8	7 (+1 pour M1)	2
ADD A, (HL)	86	7 (+1 pour M1)	2
ADD A, (IX+vs)	DD 86 vs	19 (+2 pour M1)	5
ADD A, (IY+vs)	FD 86 vs	19 (+2 pour M1)	5

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

**Exemple :**

avant		après
A = E4h CF = ?	ADD A, 2AH	A = 1Eh CF = 1

Il peut vous paraître bizarre que l'indicateur CF à 1 lors de l'exécution de cette instruction. Pourtant, ceci n'a rien de contradictoire avec ce que nous venons de dire : en effet, l'addition est effectuée sans prise en compte de l'état de CF. Cependant, si cette addition engendre une retenue, CF est alors mise à 1.

**Rôle :** Additionne le contenu de la valeur de l'octet de poids et du double registre d'indexation spécifiés à l'accumulateur. Le résultat est placé dans l'accumulateur (registre A).

dri peut être IX ou IY (avec H pour bits de poids fort, L pour bits de poids faible).

**Opération :**  $A = A + \text{la valeur de l'octet de poids spécifié du double registre d'indexation}$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est nul.

HF est mis à 1 si l'addition des 3 bits de poids faible provoque une retenue.

$PF/VF$  mis à 1 lors d'un débordement.

NF est mis à 0.

CF est mis à 1 si l'addition provoque une retenue.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
ADD A, IXH	DD 84	8 (+2 pour M1)	2
ADD A, IXL	DD 85	8 (+2 pour M1)	2
ADD A, IYH	FD 84	8 (+2 pour M1)	2
ADD A, IYL	FD 85	8 (+2 pour M1)	2

**Remarque :** Ces quatre instructions ne sont pas officielles. Elles ne sont pas documentées par Zilog, ni par ASCII.

## ADD HL, dr

**Rôle :** Additionne le contenu du double registre HL au contenu du double registre dr. Le résultat est placé dans HL. L'addition est effectuée sans prise en compte de l'état de CF. Le résultat est placé dans le double registre HL.

dr peut être BC, DE, HL ou SP.

**Opération :**  $HL = HL + dr$

**État du registre F après exécution :**

SF ne change pas.

ZF ne change pas.

HF est mis à 1 si l'addition des 11 bits de poids faible provoque une retenue.

<sup>PF</sup>/<sub>VF</sub> ne change pas.

NF est mis à 0.

CF est mis à 1 si l'addition provoque une retenue.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
ADD HL, BC	09	11 (+1 pour M1)	1
ADD HL, DE	19	11 (+1 pour M1)	1
ADD HL, HL	29	11 (+1 pour M1)	1
ADD HL, SP	39	11 (+1 pour M1)	1

**Exemple :** Multiplier HL par 2.

avant		après
HL = 5040h	ADD HL, HL	HL = A08Ah
CF = ?		CF = 0

**Remarque :** Seuls CF et HF sont affectés. SF, ZF et PF ne sont pas modifiés.



**Rôle :** Le contenu du registre d'index est additionné au contenu du double registre dr ou dri. Le résultat est placé dans le registre d'index. L'addition ignore l'état de CF.

dri peut être IX ou IY.

op peut être BC, DE, SP, IX ou IY.

**Opération :** dri = dri + dr

**État du registre F après exécution :**

SF ne change pas

ZF ne change pas

HF est mis à 1 si l'addition des 4 bits de poids faible provoque une retenue.

<sup>PF</sup>/<sub>VF</sub> ne change pas.

NF est mis à 0

CF est mis à 1 si l'addition provoque une retenue.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
ADD IX, BC	DD 09	15 (+2 pour M1)	2
ADD IX, DE	DD 19	15 (+2 pour M1)	2
ADD IX, IX	DD 29	15 (+2 pour M1)	2
ADD IX, SP	DD 39	15 (+2 pour M1)	2
ADD IY, BC	FD 09	15 (+2 pour M1)	2
ADD IY, DE	FD 19	15 (+2 pour M1)	2
ADD IY, IY	FD 29	15 (+2 pour M1)	2
ADD IY, SP	FD 39	15 (+2 pour M1)	2

**Exemple :**

avant		après
IX = A790h	ADD IX, BC	IX = 7BE0h
BC = D450h		BC = D450h
CF = ?		CF = 1

**Remarque :** Avec cette instruction, on ne peut pas additionner IX à IY (ou IY à IX). On peut additionner seulement IX à IX ou IY à IY.

**Rôle :** Ajustement décimal de l'accumulateur pour effectuer une addition ou une soustraction en décimale. Comme le Z80 ne sait faire que des opérations arithmétiques en binaire, il faut donc effectuer une correction d'ajustement sur le résultat. C'est ce que se charge de faire l'instruction DAA. De ce fait, DAA est censée être utilisée juste après une instruction d'addition (ADD, ADC, INC) ou de soustraction (SUB, SBC, DEC, NEG). En fait, DAA se base sur l'indicateur NF pour distinguer si une instruction d'addition ou de soustraction a été exécutée. Les instructions d'addition mettent l'indicateur NF à 0 et celles de soustraction le mettent à 1. L'accumulateur est ajusté en fonction des indicateurs CF et HF. Un octet placé dans A peut être décomposé en deux quartets : un quartet de poids fort Qh (quartet haut) et un quartet de poids faible Qb (quartet bas). Pour travailler en décimale, chacun de ces quartets doit contenir un nombre compris entre 0 et 9. Le tableau suivant indique comment s'effectue cet ajustement.

**Opération :** A est ajusté en décimal de la façon suivante.

- Si le registre A est supérieur à 0x99, ou bien que le drapeau CF= 1, alors les quatre bits de poids fort du facteur de correction sont réglés sur 6 et CF restera à 1. Autrement les quatre bits de poids fort du facteur de correction sont mis à 0 et CF est mis à 0.
- Si les quatre bits de poids faible du registre A sont supérieurs à 9 (A AND 0Fh), ou si le HF est à 1, alors les quatre bits de poids faible du facteur de correction sont définis sur 6. Autrement, les quatre bits de poids faible du facteur de correction sont définis sur 0.
- Il en résulte donc un facteur de correction de 00h, 06h, 60h ou 66h.
- Dès lors, si le drapeau NF est à 0, alors le facteur de correction est ajouté au registre A. Autrement, il y est soustrait.

**État du registre F après exécution :**

SF prend la valeur du bit 7 du résultat.

ZF est mis à 1 si le résultat est zéro.

HF 1 si DAA a provoqué une retenue ou un emprunt du bit 3 au bit 4.

C'est à dire, HF = bit 4 de A (avant DAA) XOR bit 4 de A (après DAA).

<sup>PF</sup>/<sub>VF</sub> est mis à 1 si le nombre de bit à 1 du résultat est pair.

NF ne change pas.

CF 1 ou 0 comme indiqué dans la première étape de l'opération ci-dessus.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
DAA	27	4 (+1 pour M1)	1

**Remarque :** Cette instruction n'est efficace que si le résultat l'opération est compris entre 0 et 63h.

**Rôle :** Décrémente de 1 le contenu de l'opérande spécifiée.

op peut être A, B, C, D, E, H, L, (HL), (vs+IX), (IY+vs) ou un octet.

**Opération :**  $op = op - 1$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 0 si le résultat est égal à zéro.

HF est mis à 1 si il y a une retenue provenant du bit 4 de l'accumulateur.

$PF/VF$  est mis à 1 lors d'un débordement.

NF est mis à 1.

CF ne change pas.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
DEC B	05	4 (+1 pour M1)	1
DEC C	0D	4 (+1 pour M1)	1
DEC D	15	4 (+1 pour M1)	1
DEC E	1D	4 (+1 pour M1)	1
DEC H	25	4 (+1 pour M1)	1
DEC L	2D	4 (+1 pour M1)	1
DEC A	3D	4 (+1 pour M1)	1
DEC (HL)	35	11 (+1 pour M1)	4
DEC (IX+vs)	DD 35 vs	23 (+2 pour M1)	7
DEC (IY+vs)	FD 35 vs	23 (+2 pour M1)	7

**Exemple :**

avant		après
HL = 3456h (3456h) = F7h	DEC (HL)	HL = 3456h (3456H) = F6h

**Rôle :** Décrémente de 1 le double registre dr.

dr peut être BC, DE, HL ou SP. dri peut être IX ou IY.

**Opération :**  $dr = dr - 1$  ou bien  $dri = dri - 1$

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
DEC BC	0B	6 (+1 pour M1)	1
DEC DE	1B	6 (+1 pour M1)	1
DEC HL	2B	6 (+1 pour M1)	1
DEC SP	3B	6 (+1 pour M1)	1
DEC IX	DD 2B	10 (+2 pour M1)	2
DEC IY	FD 2B	10 (+2 pour M1)	2

**Exemples :**

<b>avant</b>		<b>après</b>
DE = 0000h	DEC DE	DE = FFFFh
<b>avant</b>		<b>après</b>
IY = 35A0h	DEC IY	IY = 359Fh

**Remarque :** Cette instruction ne modifie pas le contenu du registre F. Vous devez faire vous-même tester de passage à zéro du double registre si vous en avez besoin.

**Exemple :** Voici un petit programme pour savoir si DE est à zéro.

```
DE_NUL:
    dec DE
    ld  A,E
    or  D           ; A = 0 si les bits de D et E sont à 0
    jr  z,DE_NUL    ; Saute si DE = 0
                    ; sinon l'exécution continue ici...
```

**Rôle :** Décrémente la valeur de l'octet de poids spécifié du double registre d'indexation.

dri peut être IX ou IY (avec H pour bits de poids fort, L pour bits de poids faible).

**Opération :** driH = driH - 1 ou bien driL = driL - 1

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 0 si le résultat est égal à zéro.

HF est mis à 1 si il y a une retenue provenant du bit 4 de l'accumulateur.

<sup>PF</sup>/<sub>VF</sub> est mis à 1 lors d'un débordement.

NF est mis à 1.

CF ne change pas.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
DEC IXH	DD 25	8 (+2 pour M1)	2
DEC IXL	DD 2D	8 (+2 pour M1)	2
DEC IYH	FD 25	8 (+2 pour M1)	2
DEC IYL	FD 2D	8 (+2 pour M1)	2

**Remarque :** Ces quatre instructions ne sont pas officielles. Elles ne sont pas documentées par Zilog, ni par ASCII.

**Rôle :** Incrémente le contenu de l'opérande op spécifiée.

op peut être A, B, C, D, E, H, L, (HL), (IX+vs), (IY+vs).

**Opération :**  $op = op + 1$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est égal à 0.

HF = est mis à 1 lorsqu'il provient du bit 3.

<sup>PF</sup>/<sub>VF</sub> est mis à 1 si op prend la valeur 80h.

NF est mis à 0.

CF ne change pas.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
INC B	04	4 (+1 pour M1)	1
INC C	0C	4 (+1 pour M1)	1
INC D	14	4 (+1 pour M1)	1
INC E	1C	4 (+1 pour M1)	1
INC H	24	4 (+1 pour M1)	1
INC L	2C	4 (+1 pour M1)	1
INC A	3C	4 (+1 pour M1)	1
INC (HL)	34	11 (+1 pour M1)	4
INC (IX+vs)	DD 34 vs	23 (+2 pour M1)	7
INC (IY+vs)	FD 34 vs	23 (+2 pour M1)	7

**Exemple :**

avant		après
B = 27h	INC B	B = 28h

**Rôle :** Incrémente le double registre spécifié.

dr peut être BC, DE, HL ou SP. dri peut être IX ou IY.

**Opération :**  $dr = dr + 1$  ou bien  $dri = dri + 1$

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
INC BC	03	6 (+1 pour M1)	1
INC DE	13	6 (+1 pour M1)	1
INC HL	23	6 (+1 pour M1)	1
INC SP	33	6 (+1 pour M1)	1
INC IX	DD 23	10 (+2 pour M1)	2
INC IY	FD 23	10 (+2 pour M1)	2

**Exemples :**

<u>avant</u>		<u>après</u>
BC = 2E5Fh	INC BC	BC = 2E60h
<u>avant</u>		<u>après</u>
IX = 3F45h	INC IX	IX = 3F46h

**Remarque :** Vous devez garder présent à l'esprit que l'incrémementation d'un double registre n'affecte pas le registre F. De ce fait, le passage à zéro du double registre ne sera pas signalé par l'indicateur Z. Vous devez donc vous-même tester la valeur du registre double pour savoir s'il y a eu ou non passage à zéro. Pour cela, vous pouvez d'abord tester la partie basse, puis, si elle est nulle, la partie haute.

**Rôle :** Incrémente la valeur de l'octet de poids spécifié du double registre d'indexation.

dri peut être IX ou IY (avec H pour bits de poids fort, L pour bits de poids faible).

**Opération :** driH = driH + 1 ou bien driL = driL + 1

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est égal à 0.

HF = est mis à 1 lorsqu'il provient du bit 3.

<sup>PF</sup>/<sub>VF</sub> est mis à 1 si op prend la valeur 80h.

NF est mis à 0.

CF ne change pas.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
INC IXH	DD 24	8 (+2 pour M1)	2
INC IXL	DD 2C	8 (+2 pour M1)	2
INC IYH	FD 24	8 (+2 pour M1)	2
INC IYL	FD 2C	8 (+2 pour M1)	2

**Remarque :** Ces quatre instructions ne sont pas officielles. Elles ne sont pas documentées par Zilog, ni par ASCII.



**Rôle :** Soustrait l'accumulateur à la valeur 0 et place le résultat dans l'accumulateur pour obtenir l'opposé de l'accumulateur. Cette opération s'appelle une complémentation à 2.

**Opération :**  $A = 0 - A$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est égal à zéro.

HF est mis à 1 si la soustraction des 4 bits de poids faible provoque une retenue.

$PF/VF$  est mis à 1 si l'accumulateur était à 80h avant l'opération.

NF est mis à 1.

CF est mis à 1 si l'accumulateur était à 00h avant l'opération.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
NEG	ED 44	8 (+2 pour M1)	2

**Exemple :**

avant		après
$A = ACh$	NEG	$A = 54h$

**Remarque :** Le complément à 2 est en fait le complément à 1 incrémenté de 1.

## SBC A, op

**Rôle :** La valeur de CF et celle de l'opérateur spécifié sont chacune soustraites au contenu de l'accumulateur. Il s'agit d'une opération faisant donc intervenir Carry, vous devez donc en connaître le contenu au préalable. Le résultat est placé dans l'accumulateur (registre A).

op peut représenter : A, B, C, D, E, H, L, (HL), (IX+vs), (IY+vs) ou un octet.

**Opération :**  $A = A - op - CF$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est égal à zéro.

HF est mis à 1 si la soustraction des 4 bits de poids faible provoque une retenue.

$PF/VF$  est mis à 1 lors d'un débordement.

NF est mis à 1.

CF est mis à 1 si la soustraction provoque une retenue.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
SBC A, sr	98+v	4 (+1 pour M1)	1
SBC A, v8	DE v8	7 (+1 pour M1)	2
SBC A, (HL)	9E	7 (+1 pour M1)	5
SBC A, (IX+vs)	DD 9E vs	19 (+2 pour M1)	7
SBC A, (IY+vs)	FD 9E vs	19 (+2 pour M1)	7

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

**Exemples :**

<b>avant</b>		<b>après</b>
A = 3Fh D = 12h CF = 1	SBC A, D	A = 2Ch D = 12h CF = 0
<b>avant</b>		<b>après</b>
A = 16h HL = 3433h (3433h) = 5 CF = 1	SBC A, (HL)	A = 10h HL = 3433h (3433h) = 5 CF = 0
<b>avant</b>		<b>après</b>
A = ? CF = 1	SBC A, A	A = FFh CF = 1

**Rôle :** La valeur de CF et celle de l'octet de poids spécifié du double registre d'indexation sont chacune soustraites au contenu de l'accumulateur. Le résultat est placé dans l'accumulateur (registre A).

dri peut être IX ou IY (avec H pour bits de poids fort, L pour bits de poids faible).

**Opération :**  $A = A - \text{la valeur de l'octet de poids spécifié du double registre d'indexation} - CF$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est égal à zéro.

HF est mis à 1 si la soustraction des 4 bits de poids faible provoque une retenue.

$^{PF}/_{VF}$  est mis à 1 lors d'un débordement.

NF est mis à 1.

CF est mis à 1 si la soustraction provoque une retenue.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
SBC A, IXH	DD 9C	8 (+2 pour M1)	2
SBC A, IXL	DD 9D	8 (+2 pour M1)	2
SBC A, IYH	FD 9C	8 (+2 pour M1)	2
SBC A, IYL	FD 9D	8 (+2 pour M1)	2

**Remarque :** Ces quatre instructions ne sont pas officielles. Elles ne sont pas documentées par Zilog, ni par ASCII.

**Rôle :** Soustrait le contenu du double registre spécifié au contenu du double registre HL, puis y soustrait aussi le contenu de CF. Le résultat est placé dans le registre HL.

dr peut être BC, DE, HL ou SP.

**Opération :**  $HL = HL - dr - CF$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est égal à zéro.

HF est mis à 1 si la soustraction des 12 bits de poids faible provoque une retenue.

$PF/VF$  est mis à 1 lors d'un débordement.

NF est mis à 1.

CF est mis à 1 si la soustraction provoque une retenue.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
SBC HL, BC	ED 42	15 (+1 pour M1)	1
SBC HL, DE	ED 52	15 (+1 pour M1)	2
SBC HL, HL	ED 62	15 (+1 pour M1)	2
SBC HL, SP	ED 72	15 (+1 pour M1)	5

**Exemple :**

avant		après
HL = 3F3Eh		HL = 0534h
BC = 3A0Ah	SBC HL, BC	BC = 3A0Ah
CF = 0		CF = 0

**Remarque :** Si CF est à 0, un SBC HL, HL annule le registre double HL. Ceci peut être utile car l'instruction SBC HL, HL n'occupe que deux octets alors que l'instruction LD HL, 0 en nécessite trois. En revanche, l'instruction SBC HL, HL nécessite 15 cycles d'horloge alors que LD HL, 0 n'en a besoin que de 10. Le gain de place se fait donc au détriment d'une perte de temps.

**Rôle :** Le contenu de l'opérande est soustrait au contenu de l'accumulateur. Contrairement à l'instruction SBC A, op, on n'a pas à se préoccuper de l'état de l'indicateur CF pour exécuter cette instruction. Le résultat est placé dans l'accumulateur (registre A).

op peut être A, B, C, D, E, H, L, (HL), (IX+vs) ou (IY+vs) ou un octet.

**Opération :**  $A = A - op$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est égal à zéro.

HF est mis à 1 si la soustraction des 4 bits de poids faible provoque une retenue.

<sup>PF</sup>/<sub>VF</sub> est mis à 1 lors d'un débordement.

NF est mis à 1.

CF est mis à 1 si la soustraction provoque une retenue.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
SUB sr	90+v	4 (+1 pour M1)	1
SUB v8	D6 v8	7 (+1 pour M1)	2
SUB (HL)	96	7 (+1 pour M1)	2
SUB (IX+vs)	DD 96 vs	19 (+2 pour M1)	5
SUB (IY+vs)	FD 96 vs	19 (+2 pour M1)	5

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

**Exemple :**

avant		après
A = A5h		A = 52h
HL = 8F70h	SUB (HL)	HL = 8F70h
(8F70h) = 53h		(8F70h) = 53h

**Remarque :** Bien que certains assembleurs admettent la syntaxe SUB A, op (cohérent avec SBC A, op), la syntaxe officielle est bien SUB op puisque A est sous-entendu comme pour CP op. L'exécution de SUB A met l'accumulateur à 0 en 4 cycles avec 1 octet alors que LD A, 0 le fait en 7 cycles avec 2 octets. La première façon est donc plus économique aussi bien au point de vue temps que place mais modifie des indicateurs.

**Rôle :** La valeur de l'octet de poids spécifié du double registre d'indexation est soustraite au contenu de l'accumulateur. Le résultat est placé dans l'accumulateur (registre A).

dri peut être IX ou IY (avec H pour bits de poids fort, L pour bits de poids faible).

**Opération :**  $A = A - \text{la valeur de l'octet de poids spécifié du double registre d'indexation}$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif, c'est à dire si op est inférieur à A.

ZF est mis à 1 si le résultat est zéro, c'est à dire si  $A = \text{la valeur de l'octet de poids spécifié du double registre d'indexation}$

HF est mis à 1 si la soustraction des 4 bits de poids faible provoque une retenue

$PF/VF$  est mis à 1 si la soustraction provoque un dépassement

NF est mis à 1

CF est mis à 1 si la soustraction provoque une retenue

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
SUB IXH	DD 94	8 (+2 pour M1)	2
SUB IXL	DD 95	8 (+2 pour M1)	2
SUB IYH	FD 94	8 (+2 pour M1)	2
SUB IYL	FD 95	8 (+2 pour M1)	2

**Remarque :** Ces quatre instructions ne sont pas officielles. Elles ne sont pas documentées par Zilog, ni par ASCII.

## MULUB A, op

**Rôle :** Cette instruction n'est disponible que sur le R800. Elle multiplie l'accumulateur par l'opérande op et place le résultat dans le double registre HL.

op peut être le simple registre B, C, D ou E seulement. A, H, L ou (HL) est aussi possible mais ces opérations ne sont pas garanties par ASCII.

**Opération :**  $HL = A \times op$

**État du registre F après exécution :**

SF est mis à 0.

ZF est mis à 1 si le résultat est zéro.

HF ne change pas.

$PF/VF$  est mis à 0.

NF ne change pas.

CF est défini lorsque le résultat dépasse les 8 bits de l'accumulateur.

**Temps d'exécution pour le R800 :**

Instruction	Code machine	Cycles d'horloge (R800)
MULUB A, B	ED C1	14
MULUB A, C	ED C9	14
MULUB A, D	ED D1	14
MULUB A, E	ED D9	14
<i>MULUB A, H</i>	<i>ED E1</i>	<i>14</i>
<i>MULUB A, L</i>	<i>ED E9</i>	<i>14</i>
<i>MULUB A, (HL)</i>	<i>ED F1</i>	<i>14</i>
<i>MULUB A, A</i>	<i>ED F9</i>	<i>14</i>

**Remarque :** Cette instruction n'est disponible que sur le R800.

## MULUW HL, dr

**Rôle :** Multiplie HL par le double registre dr et place le résultat dans DEHL. (16 bits de poids faible dans HL et les 16 bits de poids fort dans DE.)

dr peut être BC ou SP seulement. HL ou DE est aussi possible mais ces opérations ne sont pas garanties par ASCII.

Opération :  $DEHL = HL \times dr$

**État du registre F après exécution :**

SF est mis à 0.

ZF est mis à 1 si le résultat est zéro.

HF ne change pas.

$PF/VF$  est mis à 0.

NF ne change pas.

CF est défini lorsque le résultat dépasse les 16 bits de HL.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (R800)
MULUW HL, BC	ED C3	36
<i>MULUW HL, DE</i>	<i>ED 52</i>	<i>36</i>
<i>MULUW HL, HL</i>	<i>ED 62</i>	<i>36</i>
MULUW HL, SP	ED F3	36

**Remarque :** Cette instruction n'est disponible que sur le R800.



## Groupe 4 : Opérations logiques

### AND op

**Rôle :** Effectue un ET logique entre l'accumulateur et l'opérande spécifiée. Le résultat est placé dans l'accumulateur (registre A).

op peut être un octet, A, B, C, D, E, H, L, (HL), (IX+vs) ou (IY+vs)

**Opération :**  $A = A \wedge \text{opérande}$

En binaire :  $0 \wedge 0 = 0$

$1 \wedge 0 = 0$

$0 \wedge 1 = 0$

$1 \wedge 1 = 1$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est zéro.

HF est mis à 0.

PF est mis à 1 si résultat est un nombre pair.

NF est mis à 0.

CF est mis à 0.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
AND sr	A0+v	4 (+1 pour M1)	1
AND v8	E6 v8	7 (+1 pour M1)	2
AND (HL)	A6	7 (+1 pour M1)	2
AND (IX+vs)	DD A6 vs	19 (+2 pour M1)	5
AND (IY+vs)	FD A6 vs	19 (+2 pour M1)	5

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

**Exemple :**

avant		après
A = 32h	AND 08H	A = 00h

**Remarque :** AND A a pour unique effet de mettre Carry à 0. Ce qui peut être utile car il n'y a pas d'instruction spécifique pour cela, mais seulement SCF pour le mettre à 1 et CCF pour l'inverser.

**Rôle :** Effectue un ET logique entre l'accumulateur et les 8 bits de poids spécifié du double registre.  
Le résultat est placé dans l'accumulateur (registre A).

dri peut être IX ou IY (avec H pour bits de poids fort, L pour bits de poids faible).

**Opération :**  $A = A \wedge 8 \text{ bits de point spécifié de dri}$

En binaire :  $0 \wedge 0 = 0$   
 $1 \wedge 0 = 0$   
 $0 \wedge 1 = 0$   
 $1 \wedge 1 = 1$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.  
 ZF est mis à 1 si le résultat est zéro.  
 HF est mis à 0.  
 PF est mis à 1 si résultat est un nombre pair.  
 NF est mis à 0.  
 CF est mis à 0.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
AND IXH	DD A4	8 (+2 pour M1)	2
AND IXL	DD A5	8 (+2 pour M1)	2
AND IYH	FD A4	8 (+2 pour M1)	2
AND IYL	FD A5	8 (+2 pour M1)	2

**Exemple :**

avant		après
A = 32h IY = 09FFh	AND IYH	A = 00h IY = 09FFh

**Remarque :** Ces quatre instructions ne sont pas officielles. Elles ne sont pas documentées par Zilog, ni par ASCII.

**Rôle :** Complémente à 1 le contenu de l'accumulateur. La complémentation à 1 signifie que l'on inverse la valeur de chaque bit.

**Opération :**  $A = \overline{A}$

En binaire : 0 = 1

1 = 0

**État du registre F après exécution :**

SF ne change pas.

ZF ne change pas.

HF est mis à 1.

$PF/VF$  ne change pas.

NF est mis à 1.

CF ne change pas.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
CPL	2F	4 (+1 pour M1)	1

**Exemple :**

avant		après
A = 6Dh (01101101b)	CPL	A = 92h (10010010b)

**Rôle :** Effectue un OU logique entre l'accumulateur et l'opérande spécifiée. Le résultat est placé dans l'accumulateur (registre A).

op peut être un octet, A, B, C, D, E, H, L, (HL), (IX+vs), (IY+vs) ou un octet.

**Opération :**  $A = A \vee \text{opérande}$

*En binaire :*  $0 \vee 0 = 0$

$1 \vee 0 = 1$

$0 \vee 1 = 1$

$1 \vee 1 = 1$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est zéro.

HF est mis à 0.

$PF/VF$  est mis à 1 le nombre de bit à 1 du résultat est pair.

NF est mis à 0.

CF est mis à 0.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
OR sr	B0+v	4 (+1 pour M1)	1
OR v8	F6 v8	7 (+1 pour M1)	2
OR (HL)	B6	7 (+1 pour M1)	2
OR (IX+vs)	DD B6 vs	19 (+2 pour M1)	5
OR (IY+vs)	FD B6 vs	19 (+2 pour M1)	5

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

**Exemple :**

avant		après
A = 22h		A = 33h
HL = 6904h	OR (HL)	HL = 6904h
(6904h) = 11h		(6904h) = 11h

**Remarque :** OR A a pour unique effet de mettre Carry à 0. Ce qui peut être utile car il n'y a pas d'instruction spécifique pour cela, mais seulement SCF pour le mettre à 1 et CCF pour l'inverser.

**Rôle :** Effectue un OU logique entre l'accumulateur et les 8 bits de poids fort ou de poids faible du double registre spécifié. Le résultat est placé dans l'accumulateur (registre A).

dri peut être IX ou IY (avec H pour bits de poids fort, L pour bits de poids faible).

**Opération :**  $A = A \vee 8 \text{ bits de point spécifié du double registre dri}$

En binaire :  $0 \vee 0 = 0$

$1 \vee 0 = 1$

$0 \vee 1 = 1$

$1 \vee 1 = 1$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est zéro.

HF est mis à 0.

PF est mis à 1 si résultat est un nombre pair.

NF est mis à 0.

CF est mis à 0.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
OR IXH	DD B4	8 (+2 pour M1)	2
OR IXL	DD B5	8 (+2 pour M1)	2
OR IYH	FD B4	8 (+2 pour M1)	2
OR IYL	FD B5	8 (+2 pour M1)	2

**Exemple :**

avant		après
A = 22h IX = 0911h	OR IXL	A = 33h IX = 0911h

**Remarque :** Ces quatre instructions ne sont pas officielles. Elles ne sont pas documentées par Zilog, ni par ASCII.

**Rôle :** Effectue l'opération logique « OU exclusif » entre l'accumulateur et le contenu de l'opérande spécifié. Le résultat est placé dans l'accumulateur (registre A).

op peut être un octet, A, B, C, D, E, H, L, (HL), (IX+vs) ou (IY+vs) ou un octet.

**Opération :**  $A = A \oplus \text{opérande}$

En binaire :  $0 \oplus 0 = 0$   
 $1 \oplus 0 = 1$   
 $0 \oplus 1 = 1$   
 $1 \oplus 1 = 0$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.  
 ZF est mis à 1 si le résultat est zéro.  
 HF est mis à 0.  
 PF est mis à 1 si résultat est un nombre pair.  
 NF est mis à 0.  
 CF est mis à 0.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
XOR sr	A8+v	4 (+1 pour M1)	1
XOR v8	EE v8	7 (+1 pour M1)	2
XOR (HL)	AE	7 (+1 pour M1)	2
XOR (IX+vs)	DD AE vs	19 (+2 pour M1)	5
XOR (IY+vs)	FD AE vs	19 (+2 pour M1)	5

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

**Exemples :**

avant		après
A = 22h B = 21h	XOR B	A = 03h B = 21h
avant		après
A = 22h	XOR 080h	A = A2h

**Remarque :** XOR A a pour effet de mettre à 00h l'accumulateur (en un octet) et ainsi que CF à 0. Ça peut être intéressant dans certains cas puisque c'est plus rapide que LD A, 0.

**Rôle :** Effectue un «OU exclusif » logique entre l'accumulateur et les 8 bits de poids fort ou de poids faible du double registre spécifié. Le résultat est placé dans l'accumulateur (registre A).

dri peut être IX ou IY (avec H pour bits de poids fort, L pour bits de poids faible).

**Opération :**  $A = A \oplus 8 \text{ bits de point spécifié du double registre dri}$

En binaire :  $0 \oplus 0 = 0$

$1 \oplus 0 = 1$

$0 \oplus 1 = 1$

$1 \oplus 1 = 0$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est zéro.

HF est mis à 0.

PF est mis à 1 si résultat est un nombre pair.

NF est mis à 0.

CF est mis à 0.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
XOR IXH	DD AC	8 (+2 pour M1)	2
XOR IXL	DD AD	8 (+2 pour M1)	2
XOR IYH	FD AC	8 (+2 pour M1)	2
XOR IYL	FD AD	8 (+2 pour M1)	2

**Exemple :**

avant		après
A = 22h IX = 8011h	OR IXH	A = A2h IX = 8011h

**Remarque :** Ces quatre instructions ne sont pas officielles. Elles ne sont pas documentées par Zilog, ni par ASCII.

## Groupe 5 : Manipulation de bit(s)

Ces instructions agissent sur un ou plusieurs bits par décalage, inversion, rotation, etc.

### CCF

**Rôle :** CCF (Complement Carry Flag) inverse l'état de l'indicateur CF.

**Opération :**  $CF = \overline{CF}$

**État du registre F après exécution :**

SF ne change pas.

ZF ne change pas.

HF prend la précédente valeur de CF.

<sup>PF</sup>/<sub>VF</sub> ne change pas.

NF est mis à 0.

CF prend la valeur inversée de sa valeur précédente.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
CCF	3F	4 (+1 pour M1)	1

**Remarque :** Il n'y a pas d'instruction de mise à 0 de Carry, mais vous pouvez le faire facilement avec XOR A (qui annule aussi l'accumulateur), OR A ou AND A, ou encore CP A qui met aussi Z à 1.



**Rôle :** Le bit <b> de l'opérande spécifiée est mis à 0.

op peut être A, B, C, D, E, H, L, (HL), (vs+IX), (vs+IY).

<b> peut être une valeur de 0 à 7.

**Opération :** Le bit <b> d'op = 0

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
RES 0, sr	CB 80+v	8 (+2 pour M1)	2
RES 1, sr	CB 88+v	8 (+2 pour M1)	2
RES 2, sr	CB 90+v	8 (+2 pour M1)	2
RES 3, sr	CB 98+v	8 (+2 pour M1)	2
RES 4, sr	CB A0+v	8 (+2 pour M1)	2
RES 5, sr	CB A8+v	8 (+2 pour M1)	2
RES 6, sr	CB B0+v	8 (+2 pour M1)	2
RES 7, sr	CB B8+v	8 (+2 pour M1)	2

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

Voir la suite du tableau à la page suivante.

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
RES 0, (HL)	CB 86+v	15 (+2 pour M1)	5
RES 1, (HL)	CB 8E+v	15 (+2 pour M1)	5
RES 2, (HL)	CB 96+v	15 (+2 pour M1)	5
RES 3, (HL)	CB 9E+v	15 (+2 pour M1)	5
RES 4, (HL)	CB A6+v	15 (+2 pour M1)	5
RES 5, (HL)	CB AE+v	15 (+2 pour M1)	5
RES 6, (HL)	CB B6+v	15 (+2 pour M1)	5
RES 7, (HL)	CB BE+v	15 (+2 pour M1)	5
RES 0, (vs+IX)	DD CB vs 86	19 (+2 pour M1)	7
RES 1, (vs+IX)	DD CB vs 8E	19 (+2 pour M1)	7
RES 2, (vs+IX)	DD CB vs 96	19 (+2 pour M1)	7
RES 3, (vs+IX)	DD CB vs 9E	19 (+2 pour M1)	7
RES 4, (vs+IX)	DD CB vs A6	19 (+2 pour M1)	7
RES 5, (vs+IX)	DD CB vs AE	19 (+2 pour M1)	7
RES 6, (vs+IX)	DD CB vs B6	19 (+2 pour M1)	7
RES 7, (vs+IX)	DD CB vs BE	19 (+2 pour M1)	7
RES 0, (vs+IY)	FD CB vs 86	19 (+2 pour M1)	7
RES 1, (vs+IY)	FD CB vs 8E	19 (+2 pour M1)	7
RES 2, (vs+IY)	FD CB vs 96	19 (+2 pour M1)	7
RES 3, (vs+IY)	FD CB vs 9E	19 (+2 pour M1)	7
RES 4, (vs+IY)	FD CB vs A6	19 (+2 pour M1)	7
RES 5, (vs+IY)	FD CB vs AE	19 (+2 pour M1)	7
RES 6, (vs+IY)	FD CB vs B6	19 (+2 pour M1)	7
RES 7, (vs+IY)	FD CB vs BE	19 (+2 pour M1)	7

**Exemple :**

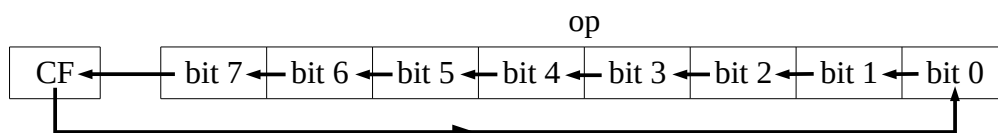
<u>avant</u>		<u>après</u>
A = 11101101b	RES 0, A	A = 11101100b

**Remarque :** Si le bit en question est déjà à 0, cette instruction n'a pas d'effet.

**Rôle :** Les bits de l'opérande spécifiée font une rotation d'un bit dans le sens inverse des aiguilles d'une montre (vers la gauche). Le bit 0 prend la valeur de Carry et celui-ci prend la valeur du bit 7. La rotation se fait donc sur 9 bits.

op peut être A, B, C, D, E, H, L, (HL), (IX+vs), (IY+vs).

**Opération :**



**État du registre F après exécution :**

SF est mis à 1 si op est négative après la rotation.

ZF est mis à 1 si op = 0 après la rotation.

HF est mis à 0.

<sup>PF</sup>/<sub>VF</sub> est mis à 1 si le nombre de bit à 1 de op est pair après la rotation.

NF est mis à 0.

CF prend la valeur précédente du bit 7.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
RL sr	CB 10+v	8 (+2 pour M1)	2
RL (HL)	CB 16	15 (+2 pour M1)	5
RL (IX+vs)	DD CB vs 16	23 (+2 pour M1)	7
RL (IY+vs)	FD CB vs 16	23 (+2 pour M1)	7

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

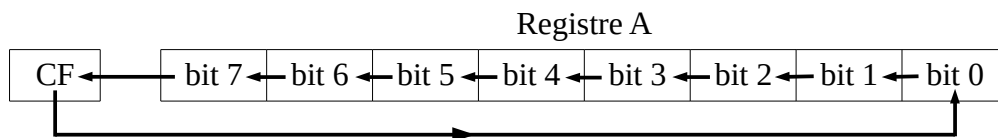
**Exemple :**

avant		après
L = 01100111b	RL L	L = 11001110b
CF = 0		CF = 0

**Remarque :** Cette opération revient à multiplier op par 2 et y additionner Carry (vérifiez-le à titre d'exercice).

**Rôle :** Les bits de l'accumulateur font une rotation d'un bit dans le sens inverse des aiguilles d'une montre (vers la gauche). Le bit 0 prend la valeur de Carry et celui-ci prend la valeur du bit 7. La rotation se fait donc sur 9 bits.

**Opération :**



**État du registre F après exécution :**

- SF ne change pas.
- ZF ne change pas.
- HF est mis à 0.
- <sup>PF</sup>/<sub>VF</sub> ne change pas.
- NF est mis à 0.
- CF prend la valeur précédente du bit 7.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
RLA	17	4 (+1 pour M1)	1

**Exemple :**

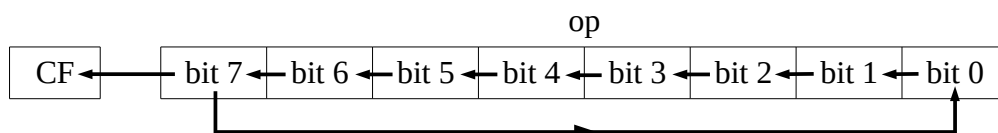
avant		après
A = 11010001b	RLA	A = 10100010b
CF = 0		CF = 1

**Remarque :** Cette opération revient à multiplier op par 2 et y additionner la valeur de CF. Vous pouvez aussi obtenir le même effet avec l'instruction RL op en mettant A comme opérande. La différence est que RLA ne prend qu'un octet contre 2 pour RL A, et qu'elle est donc plus rapide (adressage inhérent contre adressage registre). En contrepartie, RLA ne modifie ni SF, ni ZF, ni PF.

**Rôle :** Les bits de l'opérande spécifiée font une rotation d'un bit dans le sens inverse des aiguilles d'une montre (vers la gauche). En même temps que le bit 7 passe dans le bit 0, il est copié dans l'indicateur CF.

op peut être A, B, C, D, E, H, L, (HL), (IX+vs), (IY+vs).

**Opération :**



**État du registre F après exécution :**

SF est mis à 1 si op est négative après la rotation.

ZF est mis à 1 si op = 0 après la rotation.

HF est mis à 0.

<sup>PF</sup>/<sub>VF</sub> est mis à 1 si le nombre de bit à 1 de op est pair après la rotation.

NF est mis à 0.

CF prend la valeur précédente du bit 7.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
RLC sr	CB 00+v	8 (+2 pour M1)	2
RLC (HL)	CB 06	15 (+2 pour M1)	5
RLC (IX+vs)	DD CB vs 06	23 (+2 pour M1)	7
RLC (IY+vs)	FD CB vs 06	23 (+2 pour M1)	7

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

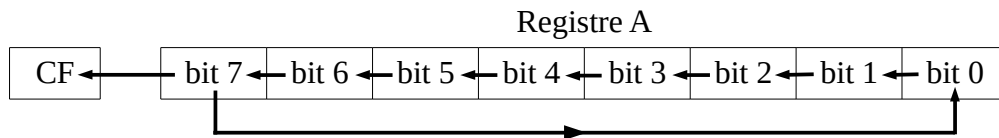
**Exemple :**

avant		après
H = 00111011b	RLC H	H = 01110110b
CF = 0		CF = 0

## RLCA

**Rôle :** Les bits de l'accumulateur font une rotation d'un bit dans le sens inverse des aiguilles d'une montre (vers la gauche). En même temps que le bit 7 passe dans le bit 0, il est copié dans l'indicateur CF.

**Opération :**



**État du registre F après exécution :**

SF ne change pas.

ZF ne change pas.

HF est mis à 0.

<sup>PF</sup>/<sub>VF</sub> ne change pas.

NF est mis à 0.

CF prend la valeur précédente du bit 7.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
RLCA	07	4 (+1 pour M1)	1

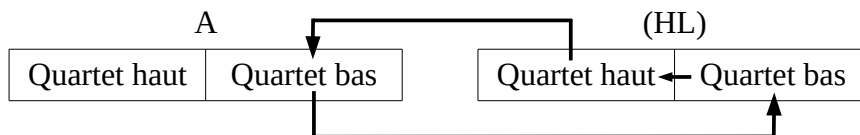
**Exemple :**

avant		après
A = 01110101b	RLCA	A = 11101010b
CF = 0		CF = 0

**Remarque :** Vous pouvez aussi obtenir le même effet avec l'instruction précédente en faisant RLC A. La différence est que RLCA ne prend qu'un octet contre 2 pour RLC A, et qu'elle est plus rapide. En contrepartie, RLCA ne modifie pas SF, ZF, et PF.

**Rôle :** Rotation par quartets de l'octet pointé par HL en passant par l'accumulateur dans le sens inverse des aiguilles d'une montre (vers la gauche). Le quartet faible du contenu de l'octet pointée par HL passe dans le quartet fort de ce même octet. Le quartet fort de l'octet va dans le quartet faible du registre A, et le quartet faible de A va dans le quartet faible de l'octet pointée par HL.

**Opération :**



**État du registre F après exécution :**

SF est mis à 1 si A est négative après la rotation.

ZF est mis à 1 si A = 0 après la rotation.

HF est mis à 0.

<sup>PF</sup>/<sub>VF</sub> est mis à 1 si le nombre de bit de A à 1 est pair après la rotation.

NF est mis à 0.

CF ne change pas.

**Code machine et temps d'exécution :**

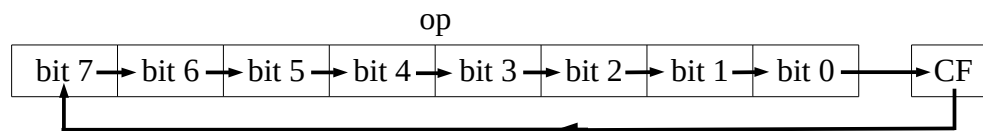
Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
RLD	ED 6F	18 (+2 pour M1)	2

**Remarque :** Cette instruction est utilisée pour faciliter les calculs en décimale.

**Rôle :** Les bits de l'opérande spécifiée font une rotation d'un bit dans le sens des aiguilles d'une montre (vers la droite) en passant par Carry. Le bit 7 prend la valeur de CF, celui-ci prend la valeur du bit 0. La rotation se fait donc sur 9 bits.

op peut être A, B, C, D, E, H, L, (HL), (IX+vs), (IY+vs).

**Opération :**



**État du registre F après exécution :**

SF est mis à 1 si op est négative après la rotation.

ZF est mis à 1 si op = 0 après la rotation.

HF est mis à 0.

<sup>PF</sup>/<sub>VF</sub> est mis à 1 si le nombre de bit à 1 de op est pair après la rotation.

NF est mis à 0.

CF prend la valeur précédente du bit 0.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
RR sr	CB 18+v	8 (+2 pour M1)	2
RR (HL)	CB 1E	15 (+2 pour M1)	5
RR (IX+vs)	DD CB vs 1E	23 (+2 pour M1)	7
RR (IY+vs)	FD CB vs 1E	23 (+2 pour M1)	7

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

**Exemple :**

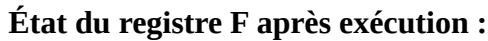
avant		après
IX = CBDCh		IX = CBDCh
(CBDEh) = 01110010b	RR (IX+2)	(CBDEh) = 10111001b
CF = 1		CF = 0

**Remarque :** Si CF = 0 avant, il s'agit d'une division par 2 de op, le reste étant dans CF (vérifiez-le à titre d'exercice).



## RRA

### Opération :



### Code machine et temps d'exécution :

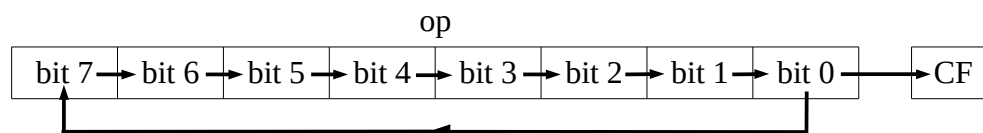
### Example :

Le Z80 et le R800 Page 81

**Rôle :** Les bits de l'opérande spécifiée effectuent une rotation d'un bit dans le sens des aiguilles d'une montre (vers la droite). Le bit 0 passe dans le bit 7 et est à la fois copié dans l'indicateur CF. La rotation se fait donc sur les 8 bits de op.

op peut être A, B, C, D, E, H, L, (HL), (IX+vs), (IY+vs).

**Opération :**



**État du registre F après exécution :**

SF est mis à 1 si op est négative après la rotation.

ZF est mis à 1 si op = 0 après la rotation.

HF est mis à 0.

<sup>PF</sup>/<sub>VF</sub> est mis à 1 si le nombre de bit à 1 de op est pair.

NF est mis à 0.

CF prend la valeur précédente du bit 0.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
RRC sr	CB 10+v	8 (+2 pour M1)	2
RRC (HL)	CB 0E	15 (+2 pour M1)	5
RRC (IX+vs)	DD CB vs 0E	23 (+2 pour M1)	7
RRC (IY+vs)	FD CB vs 0E	23 (+2 pour M1)	7

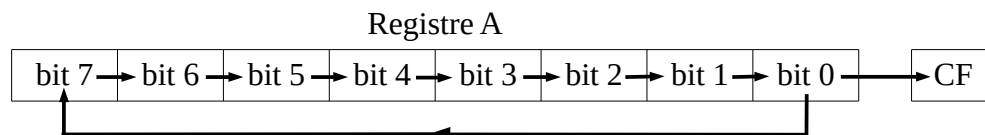
Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

**Exemple :**

avant		après
B = 10111011b	RRC B	B = 11011101b
CF = ?		CF = 1

**Rôle :** Les bits de l'accumulateur effectuent une rotation d'un bit dans le sens des aiguilles d'une montre (vers la droite). Le bit 0 passe dans le bit 7 et est à la fois copié dans l'indicateur CF. La rotation se fait donc sur les 8 bits de A.

**Opération :**



**État du registre F après exécution :**

- SF ne change pas.
- ZF ne change pas.
- HF est mis à 0.
- <sup>PF</sup>/<sub>VF</sub> ne change pas.
- NF est mis à 0.
- CF prend la valeur précédente du bit 0.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
RRCA	07	4 (+1 pour M1)	1

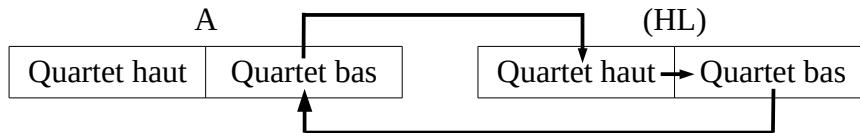
**Exemple :**

avant		après
A = 01110101b	RRCA	A = 10111010b
CF = ?		CF = 1

**Remarque :** Vous pouvez aussi obtenir le même effet avec l'instruction RRC A à la différence que RRCA ne prend qu'un octet contre 2 pour RRC A, et qu'elle est donc plus rapide. En contrepartie, RRCA ne modifie ni SF, ni ZF, ni PF.

**Rôle :** Rotation par quartets de l'octet pointé par HL en passant par l'accumulateur dans le sens des aiguilles d'une montre (vers la droite). Le quartet fort du contenu de l'octet pointée par HL passe dans le quartet faible de ce même octet. Le quartet faible de l'octet va dans le quartet faible du registre A, et le quartet faible de A va dans le quartet fort de l'octet pointée par HL.

**Opération :**



**État du registre F après exécution :**

SF est mis à 1 si A est négative après la rotation.

ZF est mis à 1 si A = 0 après la rotation.

HF est mis à 0.

<sup>PF</sup>/<sub>VF</sub> est mis à 1 si le nombre de bit à 1 de A est pair après la rotation.

NF est mis à 0.

CF ne change pas.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
RRD	ED 67	18 (+2 pour M1)	5

**Remarque :** Cette instruction est utilisée pour faciliter les calculs DCB.

## SCF

**Rôle :** SCF (Set Carry Flag) met l'indicateur CF à 1.

**Opération :** CF = 1

**État du registre F après exécution :**

SF ne change pas.  
ZF ne change pas.  
HF est mis à 0.  
<sup>PF</sup>/<sub>VF</sub> ne change pas.  
NF est mis à 0.  
CF est mis à 1.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
SCF	37	4 (+1 pour M1)	1

**Remarque :** Utilisée par exemple avant une rotation de bits.

## SET <b>, op

**Rôle :** Le bit b de l'opérande spécifiée est mis à 1.

<b> peut être une valeur de 0 à 7.

op peut être A, B, C, D, E, H, L, (HL), (IX+vs), (IY+vs).

**Opération :** bit <b> de op = 1

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
SET 0, sr	CB C0+v	8 (+2 pour M1)	2
SET 1, sr	CB C8+v	8 (+2 pour M1)	2
SET 2, sr	CB D0+v	8 (+2 pour M1)	2
SET 3, sr	CB D8+v	8 (+2 pour M1)	2
SET 4, sr	CB E0+v	8 (+2 pour M1)	2
SET 5, sr	CB E8+v	8 (+2 pour M1)	2
SET 6, sr	CB F0+v	8 (+2 pour M1)	2
SET 7, sr	CB F8+v	8 (+2 pour M1)	2

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

Voir la suite du tableau à la page suivante.

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
SET 0, (HL)	CB B6+v	15 (+2 pour M1)	3
SET 1, (HL)	CB BE+v	15 (+2 pour M1)	3
SET 2, (HL)	CB C6+v	15 (+2 pour M1)	3
SET 3, (HL)	CB CE+v	15 (+2 pour M1)	3
SET 4, (HL)	CB D6+v	15 (+2 pour M1)	3
SET 5, (HL)	CB DE+v	15 (+2 pour M1)	3
SET 6, (HL)	CB E6+v	15 (+2 pour M1)	3
SET 7, (HL)	CB EE+v	15 (+2 pour M1)	3
SET 0, (vs+IX)	DD CB vs B6	19 (+2 pour M1)	5
SET 1, (vs+IX)	DD CB vs BE	19 (+2 pour M1)	5
SET 2, (vs+IX)	DD CB vs C6	19 (+2 pour M1)	5
SET 3, (vs+IX)	DD CB vs CE	19 (+2 pour M1)	5
SET 4, (vs+IX)	DD CB vs D6	19 (+2 pour M1)	5
SET 5, (vs+IX)	DD CB vs DE	19 (+2 pour M1)	5
SET 6, (vs+IX)	DD CB vs E6	19 (+2 pour M1)	5
SET 7, (vs+IX)	DD CB vs EE	19 (+2 pour M1)	5
SET 0, (vs+IY)	FD CB vs B6	19 (+2 pour M1)	5
SET 1, (vs+IY)	FD CB vs BE	19 (+2 pour M1)	5
SET 2, (vs+IY)	FD CB vs C6	19 (+2 pour M1)	5
SET 3, (vs+IY)	FD CB vs CE	19 (+2 pour M1)	5
SET 4, (vs+IY)	FD CB vs D6	19 (+2 pour M1)	5
SET 5, (vs+IY)	FD CB vs DE	19 (+2 pour M1)	5
SET 6, (vs+IY)	FD CB vs E6	19 (+2 pour M1)	5
SET 7, (vs+IY)	FD CB vs EE	19 (+2 pour M1)	5

**Exemple :**

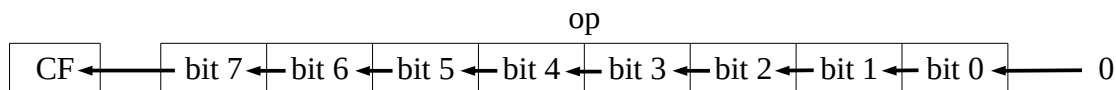
<u>avant</u>		<u>après</u>
D = 00000000b	SET 5, D	D = 00100000b

**Remarque :** Si le bit en question est déjà à 1, cette instruction n'a pas d'effet.

**Rôle :** Décalage à gauche des bits d'op. Le bit 7 passe dans l'indicateur CF, le bit 0 est mis à 0.

op peut être A, B, C, D, E, H, L, (HL), (IX+vs), (IY+vs).

**Opération :**



**État du registre F après exécution :**

SF est mis à 1 si op est négative après le décalage.

ZF est mis à 1 si op = 0 après le décalage.

HF est mis à 0.

<sup>PF</sup>/<sub>VF</sub> est mis à 1 si le nombre de bit à 1 de op est pair après le décalage.

NF est mis à 0.

CF prend la valeur précédente du bit 7.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
SLA sr	CB 10+v	8 (+2 pour M1)	2
SLA (HL)	CB 0E	15 (+2 pour M1)	5
SLA (IX+vs)	DD CB vs 0E	23 (+2 pour M1)	7
SLA (IY+vs)	FD CB vs 0E	23 (+2 pour M1)	7

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

**Exemple :**

avant		après
L = 100010011b	SLA L	L = 00100110b
CF = ?		CF = 1

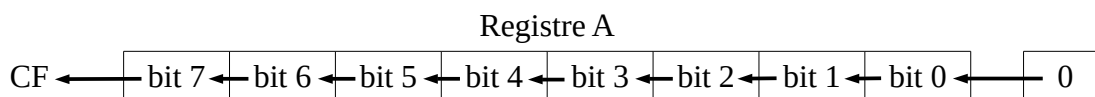
**Remarque :** Bien que la lettre A figure dans la mnémonique (SLA – *Shift Left Arithmetical*, décalage arithmétique vers la gauche), cette instruction peut aussi agir sur les autres registres ou en mode indirect.

SLA effectue en fait la multiplication par 2 de op.

**Rôle :** Décalage à gauche des bits de op et mise à 0 du bit 0. Le bit 7 tombe dans l'indicateur CF.

op peut être A, B, C, D, E, H, L, (HL), (IX+vs), (IY+vs).

**Opération :**



**État du registre F après exécution :**

SF est mis à 0.

ZF est mis à 1 si op = 0 après le décalage.

HF est mis à 0.

$PF/VF$  est mis à 1 si le nombre de bit à 1 de op est pair après le décalage.

NF est mis à 0.

CF prend la valeur précédente du bit 7.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
SRL sr	CB 30+v	8 (+2 pour M1)	2
SRL (HL)	CB 36	15 (+2 pour M1)	5
SRL (IX+vs)	DD CB vs 36	23 (+2 pour M1)	7
SRL (IY+vs)	FD CB vs 36	23 (+2 pour M1)	7

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

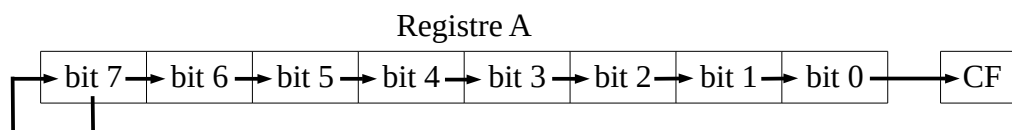
**Remarque :** Ces instructions ne sont pas officielles pour le Z80. Elles ne sont pas documentées par Zilog, ni par ASCII. Le R800 n'a pas ces instructions. Elles ont été remplacées par SLA.



**Rôle :** Décalage à droite des bits de op. Le bit 0 passe dans l'indicateur CF, le bit 7 est inchangé.

op peut être A, B, C, D, E, H, L, (HL), (IX+vs), (IY+vs).

**Opération :**



**État du registre F après exécution :**

SF est mis à 1 si op est négative après le décalage.

ZF est mis à 1 si op = 0 après le décalage.

HF est mis à 0.

<sup>PF</sup>/<sub>VF</sub> est mis à 1 si le nombre de bit à 1 de op est pair après le décalage.

NF est mis à 0.

CF prend la valeur précédente du bit 0.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
SRA sr	CB 28+v	8 (+2 pour M1)	2
SRA (HL)	CB 2E	15 (+2 pour M1)	5
SRA (IX+vs)	DD CB vs 2E	23 (+2 pour M1)	7
SRA (IY+vs)	FD CB vs 2E	23 (+2 pour M1)	7

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

**Exemple :**

avant		après
IY = BC12h		IY = BC12h
(BC12h) = 11001011b	SRA (IY)	(BC12h) = 11100101b
CF = ?		CF = 1

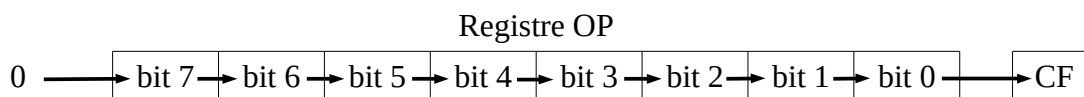
**Remarque :** Bien que la lettre A figure dans la mnémonique (SRA – *Shift Right Arithmetical*, décalage arithmétique vers la droite), cette instruction peut aussi agir sur les autres registres ou en mode indirect.

## SRL op

**Rôle :** Décalage à droite des bits de op et mise à 0 du bit 7. Le bit 0 tombe dans l'indicateur CF.

op peut être A, B, C, D, E, H, L, (HL), (IX+vs), (IY+vs).

**Opération :**



**État du registre F après exécution :**

SF est mis à 0.

ZF est mis à 1 si op = 0 après le décalage.

HF est mis à 0.

$PF/VF$  est mis à 1 si le nombre de bit à 1 de op est pair après le décalage.

NF est mis à 0.

CF prend la valeur précédente du bit 0.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
SRL sr	CB 38+v	8 (+2 pour M1)	2
SRL (HL)	CB 3E	15 (+2 pour M1)	5
SRL (IX+vs)	DD CB vs 3E	23 (+2 pour M1)	7
SRL (IY+vs)	FD CB vs 3E	23 (+2 pour M1)	7

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

**Exemple :**

avant		après
HL = A0A0h (A0A0h) = 11100001 C = ?	SRL (HL)	HL = A0A0h (A0A0h) = 01110000b CF = 1

**Remarque :** SRL op effectue en fait la division par 2 de l'opérateur spécifié, le reste étant déposé dans Carry. Toutes ces instructions de décalage se ressemblent un peu. N'hésitez pas à revenir consulter ces pages à chaque fois que vous en avez besoin.

## Groupe 6 : Comparaisons de données et test de bit(s)

Ces instructions permettent de tester la valeur d'un bit ou d'un octet.

### BIT <b>, op

**Rôle :** Teste le bit <b> de l'opérande spécifiée.

op peut être A, B, C, D, E, H, L, (HL), (IX+vs), (IY+vs).

**Opération :** ZF = valeur inversée du bit <b> de op

**État du registre F après exécution :**

SF est indéterminé.

ZF prend la valeur inverse du bit testé.

HF est mis à 1.

<sup>PF</sup>/<sub>VF</sub> est indéterminé.

NF est mis à 0.

CF ne change pas.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
BIT 0, sr	CB 40+v	8 (+2 pour M1)	2
BIT 1, sr	CB 48+v	8 (+2 pour M1)	2
BIT 2, sr	CB 50+v	8 (+2 pour M1)	2
BIT 3, sr	CB 58+v	8 (+2 pour M1)	2
BIT 4, sr	CB 60+v	8 (+2 pour M1)	2
BIT 5, sr	CB 68+v	8 (+2 pour M1)	2
BIT 6, sr	CB 70+v	8 (+2 pour M1)	2
BIT 7, sr	CB 78+v	8 (+2 pour M1)	2

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

Voir la suite du tableau à la page suivante.

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
BIT 0, (HL)	CB 46+v	15 (+2 pour M1)	3
BIT 1, (HL)	CB 4E+v	15 (+2 pour M1)	3
BIT 2, (HL)	CB 56+v	15 (+2 pour M1)	3
BIT 3, (HL)	CB 5E+v	15 (+2 pour M1)	3
BIT 4, (HL)	CB 66+v	15 (+2 pour M1)	3
BIT 5, (HL)	CB 6E+v	15 (+2 pour M1)	3
BIT 6, (HL)	CB 76+v	15 (+2 pour M1)	3
BIT 7, (HL)	CB 7E+v	15 (+2 pour M1)	3
BIT 0, (vs+IX)	DD CB vs 46	19 (+2 pour M1)	5
BIT 1, (vs+IX)	DD CB vs 4E	19 (+2 pour M1)	5
BIT 2, (vs+IX)	DD CB vs 56	19 (+2 pour M1)	5
BIT 3, (vs+IX)	DD CB vs 5E	19 (+2 pour M1)	5
BIT 5, (vs+IX)	DD CB vs 66	19 (+2 pour M1)	5
BIT 5, (vs+IX)	DD CB vs 6E	19 (+2 pour M1)	5
BIT 6, (vs+IX)	DD CB vs 76	19 (+2 pour M1)	5
BIT 7, (vs+IX)	DD CB vs 7E	19 (+2 pour M1)	5
BIT 0, (vs+IY)	FD CB vs 46	19 (+2 pour M1)	5
BIT 1, (vs+IY)	FD CB vs 4E	19 (+2 pour M1)	5
BIT 2, (vs+IY)	FD CB vs 56	19 (+2 pour M1)	5
BIT 3, (vs+IY)	FD CB vs 5E	19 (+2 pour M1)	5
BIT 4, (vs+IY)	FD CB vs 66	19 (+2 pour M1)	5
BIT 5, (vs+IY)	FD CB vs 6E	19 (+2 pour M1)	5
BIT 6, (vs+IY)	FD CB vs 76	19 (+2 pour M1)	5
BIT 7, (vs+IY)	FD CB vs 7E	19 (+2 pour M1)	5

**Exemple :**

avant		après
H = 01110001b Z = ?	BIT 2, H	H = 01110001b Z = 1

**Remarque :** N'oubliez pas que les bits d'un octet sont numérotés de 0 à 7 à partir de la droite. Cette instruction, très pratique, est souvent utilisée après la lecture d'un port d'entrée. Elle est généralement suivie d'un saut conditionnel sur ZF.

**Rôle :** Compare la valeur de l'opérande spécifié avec celle de l'accumulateur. La comparaison se fait en interne en soustrayant l'opérande à l'accumulateur (A - op). Le résultat n'est pas accessible mais les indicateurs sont modifiés en fonction du résultat.

op peut être A, B, C, D, E, H, L, (HL), (IX+vs) ou (IY+vs) ou un octet.

**Opération :** A - op

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est zéro, c'est à dire si A = op.

HF est mis à 1 si la soustraction des 4 bits de poids faible provoque une retenue.

<sup>PF</sup>/<sub>VF</sub> est mis à 1 si la soustraction provoque un dépassement.

NF est mis à 1.

CF est mis à 1 si la soustraction provoque une retenue, c'est à dire si A est inférieur à op.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
CP sr	B8+v	4 (+1 pour M1)	1
CP v8	FE v8	7 (+1 pour M1)	2
CP (HL)	BE	7 (+1 pour M1)	2
CP (IX+vs)	DD BE vs	19 (+2 pour M1)	5
CP (IY+vs)	FD BE vs	19 (+2 pour M1)	5

Ici, sr peut-être le registre B, C, D, E, H, L ou A. Chacun ayant respectivement la valeur v de 0 à 5 ou 7 à ajouter pour obtenir le code machine.

**Exemple :**

avant		après
A = 32h	CP 08h	SF ZF HF PF NF CF
		0 0 1 0 1 0

**Remarques :** CF est mis à 1 si A est inférieur à op (ceci est valable pour des valeurs positives : si vous travaillez en arithmétique signée, c'est un peu plus compliqué, car il faut tenir compte de VF. En pratique, considérez que les comparaisons opèrent sur des nombres non signés et intégrez le bit de signe dans la valeur de l'octet pour votre raisonnement).

**Rôle :** CP compare la valeur de l'octet de poids spécifié du double registre d'indexation avec celle de l'accumulateur. La comparaison se fait en interne en soustrayant l'opérande à l'accumulateur ( $A - op$ ). Le résultat n'est pas accessible mais les indicateurs sont modifiés en fonction du résultat.

dri peut être IX ou IY (H pour spécifier les bits de poids fort, L pour les bits de poids faible).

**Opération :**  $A -$  la valeur de l'octet de poids spécifié du double registre d'indexation

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est zéro, c'est à dire si  $A =$  la valeur de l'octet de poids spécifié du double registre d'indexation.

HF est mis à 1 si la soustraction des 4 bits de poids faible provoque une retenue.

$PF/VF$  est mis à 1 si la soustraction provoque un dépassement.

NF est mis à 1.

CF est mis à 1 si la soustraction provoque une retenue, c'est à dire si  $A$  est inférieur à  $op$ .

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
CP IXH	DD BC	8 (+2 pour M1)	2
CP IXL	DD BD	8 (+2 pour M1)	2
CP IYH	FD BC	8 (+2 pour M1)	2
CP IYL	FD BD	8 (+2 pour M1)	2

**Remarque :** Ces quatre instructions ne sont pas officielles. Elles ne sont pas documentées par Zilog, ni par ASCII.

**Rôle :** CPD compare le contenu de l'adresse pointée par HL avec l'accumulateur. La comparaison se fait en interne en soustrayant l'opérande à l'accumulateur. Les indicateurs du registre F sont modifiés en fonction du résultat. Ensuite HL et BC sont tous deux décrémentés (et prêts pour faire la comparaison avec l'octet précédant). CPDR effectue la même opération jusqu'à ce que op égale A ou que BC égale zéro.

**Opération :**  $A - (HL)$  ;  $HL = HL - 1$  ;  $BC = BC - 1$  ; si  $BC \neq 0$  alors  $PC = PC - 2$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est zéro, c'est à dire si  $A = (HL)$ .

HF est mis à 1 si la soustraction des 4 bits de poids faible provoque une retenue.

$^{PF}/_{VF}$  est mis à 1 si  $BC - 1$  est différent de 0.

NF est mis à 1.

CF ne change pas.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
CPD	ED A9	16 (+2 pour M1)	4
CPDR	ED B9	23 (+2 pour M1) si $BC \neq 0$ 18 (+2 pour M1) si $BC = 0$	5

**Remarques :**

- Ici c'est PF qui indique que BC est passé à 0 (dans d'autres instructions, ce peut être ZF), et ZF qui indique l'égalité dans la comparaison. BC sert de compteur d'octets c'est pourquoi il est décrémenté automatiquement ; HL doit au préalable être chargé avec l'adresse du 1<sup>er</sup> octet à comparer.
- CPDR est une instruction puissante qui peut par exemple chercher automatiquement un octet donné dans une table. HL doit pointer sur la fin de la table (du côté des adresses hautes), et BC doit contenir la longueur de cette table. En sortie, si la table contient la valeur cherchée, HL contient l'adresse de la 1<sup>ère</sup> occurrence à partir des adresses hautes, sinon BC contient 0.

**Rôle :** CPI compare le contenu de l'adresse pointée par HL avec celui de l'accumulateur. La comparaison se fait en interne en soustrayant le contenu de l'adresse pointée par HL à celui de l'accumulateur. Les indicateurs du registre F sont modifiés en fonction du résultat. Ensuite HL est incrémenté et BC décrémentés (et prêts pour faire la comparaison avec l'octet suivant). CPIR effectue la même opération jusqu'à ce que op égale A ou que BC égale zéro.

**Opération :**  $A - (HL)$  ;  $HL = HL + 1$  ;  $BC = BC - 1$  ; si  $BC \neq 0$  alors  $PC = PC - 2$

**État du registre F après exécution :**

SF est mis à 1 si le résultat est négatif.

ZF est mis à 1 si le résultat est zéro, c'est à dire si  $A = (HL)$ .

HF est mis à 1 si la soustraction des 4 bits de poids faible provoque une retenue.

<sup>PF</sup>/<sub>VF</sub> est mis à 1 si  $BC - 1$  est différent de 0.

NF est mis à 1.

CF ne change pas.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
CPI	ED A1	16 (+2 pour M1)	4
CPIR	ED B1	23 (+2 pour M1) si $BC \neq 0$ 18 (+2 pour M1) si $BC = 0$	5

**Remarque :**

- Ici c'est PF qui indique que BC est passé à 0 (dans d'autres instructions, ce peut être ZF), et ZF qui indique l'égalité dans la comparaison. BC sert de compteur d'octets c'est pourquoi il est décrémenté automatiquement ; HL doit au préalable être chargé avec l'adresse du 1<sup>er</sup> octet à comparer.
- CPIR est une instruction puissante qui peut par exemple chercher automatiquement un octet donné dans une table. HL doit pointer sur le début de la table (du côté des adresses basses), et BC doit contenir la longueur de cette table. En sortie, si la table contient la valeur cherchée, HL contient l'adresse de la 1<sup>ère</sup> occurrence à partir des adresses hautes, sinon BC contient 0.



## Groupe 7 : Appel à une routine, sauts et pile

### CALL v16

**Rôle :** CALL permet d'appeler un sous-programme placée à l'adresse spécifiée.

**Opération :**  $SP = SP - 2$  ;  $(SP) = PC$  ;  $PC =$  l'adresse spécifié

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
CALL v16	CD ob oh	16 (+1 pour M1)	5

ob est l'octet de poids faible de l'adresse spécifiée (v16) et oh l'octet de poids fort.

**Remarque :** L'instruction RET permet de revenir à l'instruction suivante après l'exécution du sous-programme.

Si vous avez besoin de la pile pendant l'exécution du sous-programme veillez à la remettre en état comme après l'exécution du CALL avant le RET. Cela veut dire que dans un sous-programme, il doit y avoir le même nombre de PUSH que de POP, sous peine de ne pas retourner ailleurs qu'à la suite du CALL. Bien entendu, vous pouvez jouer avec PUSH et POP pour retourner là où vous voulez et non après le CALL mais cette technique est délicate, et déconseillée aux débutants.

## CALL <C>, v16

**Rôle :** CALL permet d'appeler un sous-programme en fonction de l'état d'un indicateur du registre F. Cette instruction teste donc l'indicateur correspondant à la condition spécifiée, puis appelle le sous-programme à l'adresse spécifiée si la condition est vraie, sinon elle passe à l'instruction suivante ( $PC = PC + 3$ ).

<c> peut être :

Z	(vrai si $ZF = 1$ )
NZ	(vrai si $ZF = 0$ )
C	(vrai si $CF = 1$ )
NC	(vrai si $CF = 0$ )
PE	(vrai si $PF/VF = 1$ )
PO	(vrai si $PF/VF = 0$ )
M	(vrai si $SF = 1$ )
P	(vrai si $SF = 0$ )

**Opération :**  $SP = SP - 2$  ;  $(SP) = PC$  ;  $PC =$  l'adresse spécifiée si <c> est vrai.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
CALL Z, v16	CC ob oh	17 (+1 pour M1) si vrai 11 (+1 pour M1) si faux	5 si vrai 3 si faux
CALL NZ, v16	C4 ob oh	17 (+1 pour M1) si vrai 11 (+1 pour M1) si faux	5 si vrai 3 si faux
CALL C, v16	DC ob oh	17 (+1 pour M1) si vrai 11 (+1 pour M1) si faux	5 si vrai 3 si faux
CALL NC, v16	D4 ob oh	17 (+1 pour M1) si vrai 11 (+1 pour M1) si faux	5 si vrai 3 si faux
CALL PE, v16	EC ob oh	17 (+1 pour M1) si vrai 11 (+1 pour M1) si faux	5 si vrai 3 si faux
CALL PO, v16	E4 ob oh	17 (+1 pour M1) si vrai 11 (+1 pour M1) si faux	5 si vrai 3 si faux
CALL M, v16	FC ob oh	17 (+1 pour M1) si vrai 11 (+1 pour M1) si faux	5 si vrai 3 si faux
CALL P, v16	F4 ob oh	17 (+1 pour M1) si vrai 11 (+1 pour M1) si faux	5 si vrai 3 si faux

ob est l'octet de poids faible de l'adresse spécifiée (v16) et oh, l'octet de poids fort.

**Exemples :**

- Si l'indicateur  $ZF = 0$ , CALL Z, ad n'effectue pas d'appel.
- Si l'indicateur  $CF = 1$ , CALL C, ad effectue l'appel.
- Si l'indicateur  $SF = 1$ , CALL M, ad n'effectue pas d'appel.

**Remarque :** L'instruction RET permet de revenir à l'instruction suivante après l'exécution du sous-programme. Voir les instructions arithmétiques, de comparaison, de décalage et de rotation de bits pour le sens des indicateurs après une comparaison.

## DJNZ vs

**Rôle :** DJNZ décrémente B, et effectue un saut relatif, comptés à partir de l'adresse suivant le DJNZ . Le déplacement en avant ou en arrière est spécifié en nombre d'octet par vs. Si B atteint 0 le saut n'est pas effectuée.

vs est un nombre signé entre -128 et +127.

**Opération :**  $B = B - 1$  ; si  $B \neq 0$  ;  $PC = PC + vs$

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
DJNZ vs	10 vs	14 (+1 pour M1) si vrai 9 (+1 pour M1) si faux	3 si vrai 2 si faux

**Remarque :** Instruction intéressante qui permet de créer facilement des boucles. Le registre B sert de compteur et doit être chargé avec le nombre de boucles à exécuter. Si vous possédez ODIN de Loricels, ou autre ASSEMBLEUR symbolique, il calculera d pour vous : vous n'aurez qu'à indiquer l'adresse absolue de branchement de manière explicite ou symbolique.

## JP v16

**Rôle :** JP permet de sauter à une adresse. L'adresse spécifiée est chargée dans PC, ce qui provoque le saut.

**Opération :**  $PC = v16$

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
JP v16	C3 ob oh	11 (+1 pour M1)	3

ob est l'octet de poids faible de l'adresse spécifiée (v16) et oh, l'octet de poids fort.

**Remarque :** Cette instruction, contrairement à CALL v16, ne permet pas de retour. Elle est utilisée continuer l'exécution du programme à une autre adresse.

**Rôle :** JP cond,v16 permet de sauter à l'adresse spécifiée en fonction de l'état d'un indicateur du registre F. Cette instruction teste donc l'indicateur correspondant à la condition spécifiée puis, charge l'adresse spécifiée dans PC lorsque la condition est vraie sinon, l'instruction suivante est exécutée.

<c> peut être :

Z	(vrai si ZF = 1)
NZ	(vrai si ZF = 0)
C	(vrai si CF = 1)
NC	(vrai si CF = 0)
PE	(vrai si $PF/VF = 1$ )
PO	(vrai si $PF/VF = 0$ )
M	(vrai si SF = 1)
P	(vrai si SF = 0)

**Opération :** Si <c> est vraie PC = adresse spécifiée.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
JP Z,v16	CA ob oh	17 (+1 pour M1) si vrai 11 (+1 pour M1) si faux	3
JP NZ,v16	C2 ob oh	17 (+1 pour M1) si vrai 11 (+1 pour M1) si faux	3
JP C,v16	DA ob oh	17 (+1 pour M1) si vrai 11 (+1 pour M1) si faux	3
JP NC,v16	D2 ob oh	17 (+1 pour M1) si vrai 11 (+1 pour M1) si faux	3
JP PE,v16	EA ob oh	17 (+1 pour M1) si vrai 11 (+1 pour M1) si faux	3
JP PO,v16	E2 ob oh	17 (+1 pour M1) si vrai 11 (+1 pour M1) si faux	3
JP M,v16	FA ob oh	17 (+1 pour M1) si vrai 11 (+1 pour M1) si faux	3
JP P,v16	F2 ob oh	17 (+1 pour M1) si vrai 11 (+1 pour M1) si faux	3

ob est l'octet de poids faible de l'adresse spécifiée (v16) et oh, l'octet de poids fort.

**Exemples :**

- JP C,06789H effectue le saut si l'indicateur CF = 1
- JP C,0F56CH n'effectue pas le saut si l'indicateur CF = 0
- JP NZ,6664H effectue le saut si l'indicateur ZF = 0.

**Remarque :** Cette instruction est importante. Le test des indicateurs est la base de la programmation : il permet d'appliquer tel ou tel traitement suivant les résultats du calcul précédent. Voir les instructions arithmétiques, de comparaison, de décalage et de rotation de bits pour le sens des indicateurs après une comparaison.

## JP (HL) et JP (dri)

**Rôle :** Ces instructions permettent de sauter à l'adresse contenue dans celle spécifiée par le double-registre spécifié. Le contenu de la valeur (16bit) pointée par le double-registre est chargé dans PC, provoquant ainsi le saut.

dri, IX ou IY.

**Opération :**  $PC = \text{Le contenu de l'adresse pointée par double-registre spécifié}$

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
JP (HL)	E9	4 (+1 pour M1)	1
JP (IX)	DD E9	8 (+2 pour M1)	2
JP (IY)	FD E9	8 (+2 pour M1)	2

**Exemple :** Si HL = 7080h, alors JP (HL) provoque un saut à l'adresse 7080h.

**Remarque :** Les parenthèses dans la mnémonique sont là pour indiquer qu'il s'agit du contenu de l'adresse pointée par le double-registre spécifié et non pas sa valeur. Ceci a été voulu par ZILOG. Cette instruction permet le saut à une adresse qui peut-être variable.

## JR vs

**Rôle :** Cette instruction effectue un saut relatif. La valeur fournie vs est un nombre signé. Le déplacement possible est donc de -128 octets en arrière à +127 octets en avant, comptés à partir de l'adresse de l'instruction qui suit le JR (puisque PC prend d'abord la valeur l'adresse de la prochaine instruction).

**Opération :**  $PC = PC + 2 + vs$

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
JR vs	18 vs	12 (+1 pour M1)	2

**Exemple :** JR 0F0H effectue un saut en arrière de 16 octets par rapport l'adresse de l'instruction suivante.

**Remarque :** La plupart des assembleurs calculent vs pour vous en utilisant un label.

**Rôle :** L'indicateur est testé. Si la condition est remplie, un saut relatif vs comme décrit pour JR vs. Si la condition n'est pas remplie, l'exécution passe à l'instruction suivante.

vs est une valeur entre -128 et +127.

<c> peut être	Z	(vrai si ZF = 1)
	NZ	(vrai si ZF = 0)
	C	(vrai si CF = 1)
	NC	(vrai si CF = 0)

**Opération :** PC = PC + 2 et aussi PC = PC + vs si <c> est vraie.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
JR Z, vs	28 vs	12 (+1 pour M1) si vrai 7 (+1 pour M1) si faux	3 si vrai 2 si faux
JR NZ, vs	20 vs	12 (+1 pour M1) si vrai 7 (+1 pour M1) si faux	3 si vrai 2 si faux
JR C, vs	38 vs	12 (+1 pour M1) si vrai 7 (+1 pour M1) si faux	3 si vrai 2 si faux
JP NC, vs	30 vs	12 (+1 pour M1) si vrai 7 (+1 pour M1) si faux	3 si vrai 2 si faux

**Exemples :**

- si l'indicateur ZF = 1, JR Z, 67H effectue un saut en avant de 103 octets en avant (comptés à partir de l'instruction suivante).
- si l'indicateur CF = 0, JR C, 9 est ignoré.

**Remarque :** Cette instruction prend 2 octets alors que JP <c>, v16 en prend 3. De plus, le saut relatif permet un relogement facile du programme en cas de besoin. Cependant, malgré ces avantages, elle ne peut tester que 2 indicateurs au lieu de 4 pour JP. Elle ne peut donc pas toujours remplacer celle-ci. Voir les instructions arithmétiques, de comparaison, de décalage et de rotation de bits pour le sens des indicateurs après une comparaison.

**Rôle :** Le contenu de l'adresse pointée par SP est chargé dans la partie faible du double registre spécifié : SP est incrémenté. Le contenu de la nouvelle adresse pointée par SP est chargée dans la partie forte du double registre spécifié. SP est de nouveau incrémenté pour pointer sur la prochaine valeur à déplier.

dr peut être AF, BD, DE ou HL. dri peut être IX ou IY.

**Opération :** Octet de poids faible de dr = (SP) ; SP = SP + 1 ;

Octet de poids fort de dr = (SP) ; SP = SP + 1

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
POP BC	C1	10 (+1 pour M1)	4
POP DE	D1	10 (+1 pour M1)	4
POP HL	E1	10 (+1 pour M1)	4
POP AF	F1	10 (+1 pour M1)	4
POP IX	DD E1	14 (+2 pour M1)	5
POP IY	FD E1	14 (+2 pour M1)	5

**Remarque :** Les instructions de chargement direct entre doubles registres sont quasi inexistantes sur le Z80, à l'exception de EX DE, HL et de LD SP, dr. On peut cependant les simuler à l'aide de PUSH et POP :

```
PUSH IX
POP BC
```

équivalent à LD BC, IX qui n'existe pas. Vous pouvez ainsi fabriquer toutes les combinaisons entre doubles registres.

## PUSH dr et PUSH dri

**Rôle :** SP est décrémenté ; la partie forte du double registre est chargée à la nouvelle adresse pointée par SP, puis SP est de nouveau décrémenté, et la partie basse du double registre est chargée à l'adresse pointée par SP. A la fin de l'opération, SP pointe donc sur la partie basse de la valeur empilée, et SP + 1 sur sa partie haute.

dr peut être AF, BD, DE ou HL. dri peut être IX ou IY.

**Opération :**  $SP = SP - 1$  ; (SP) = Octet de poids fort du double registre spécifié ;

$SP = SP - 1$  ; (SP) = Octet de poids faible du double registre spécifié

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
PUSH BC	C5	11 (+1 pour M1)	4
PUSH DE	D5	11 (+1 pour M1)	4
PUSH HL	E5	11 (+1 pour M1)	4
PUSH AF	F5	11 (+1 pour M1)	4
PUSH IX	DD E5	15 (+2 pour M1)	5
PUSH IY	FD E5	15 (+2 pour M1)	5

**Remarque :** La pile croît vers le bas : chaque nouvel empilement se place sous l'adresse empilée précédemment.

## RET

**Rôle :** RET sert à reprendre l'exécution d'un programme après l'appel d'un sous-programme par un CALL. Pour faire cela, RET récupère la dernière adresse qui a été mis sur la pile afin de continuer l'exécution du programme à cette adresse en la plaçant dans PC. Cette instruction pourrait s'appeler POP PC en fait.

**Opération :**  $PC = (SP)$  ;  $SP = SP + 2$

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
RET	C9	4 (+1 pour M1)	3

**Remarque :** On peut appeler de nombreux sous-programmes les uns dans les autres. Chaque RET renvoie juste après le CALL précédant. En cas de PUSH et de POP à l'intérieur d'un sous-programme, mettez-en le même nombre de chacun à l'intérieur pour que le RET reprenne la bonne adresse de retour. L'emploi est équivalent à celui de GOSUB... RETURN du BASIC.



**Rôle :** Retour conditionnel de sous-programme. L'indicateur spécifié est testé. Si la condition est vraie, la dernière adresse qui a été mis sur la pile est placée dans PC, provoquant le retour du sous-programme afin de continuer l'exécution du programme. Si la condition est pas remplie, l'exécution continue à l'instruction suivante.

<c> peut être :

Z	(vrai si ZF = 1)
NZ	(vrai si ZF = 0)
C	(vrai si CF = 1)
NC	(vrai si CF = 0)
PO	(vrai si PF = 1)
PE	(vrai si PF = 0)
M	(vrai si SF = 1)
P	(vrai si SF = 0)

**Opération :** PC = (SP) si <c> est vraie, sinon ignoré

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
RET Z	C8	12 (+1 pour M1) si vrai 6 (+1 pour M1) si faux	3 si vrai 1 si faux
RET NZ	C0	12 (+1 pour M1) si vrai 6 (+1 pour M1) si faux	3 si vrai 1 si faux
RET C	D8	12 (+1 pour M1) si vrai 6 (+1 pour M1) si faux	3 si vrai 1 si faux
RET NC	D0	12 (+1 pour M1) si vrai 6 (+1 pour M1) si faux	3 si vrai 1 si faux
RET PE	E8	12 (+1 pour M1) si vrai 6 (+1 pour M1) si faux	3 si vrai 1 si faux
RET PO	E0	12 (+1 pour M1) si vrai 6 (+1 pour M1) si faux	3 si vrai 1 si faux
RET M	F8	12 (+1 pour M1) si vrai 6 (+1 pour M1) si faux	3 si vrai 1 si faux
RET P	F0	12 (+1 pour M1) si vrai 6 (+1 pour M1) si faux	3 si vrai 1 si faux

**Exemple :** Si l'indicateur ZF = 1, RET Z provoque le retour du sous-programme.

**Rôle :** Effectuer un appel à la routine dont l'adresse est 0000h, 0008h, 0010h, 0018h, 0020h, 0028h, 0030h ou 0038h. L'exécution est plus rapide qu'un CALL v16.

v8 peut être 0H, 8H, 10H, 18H, 20H, 28H, 30H ou 38H. (Le H est souvent omis)

**Opération :**  $SP = SP - 2$  ;  $(SP) = PC$  ;  $PC = \text{l'adresse spécifiée}$

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
RST 0	C7	11 (+1 pour M1)	1
RST 8	CF	11 (+1 pour M1)	1
RST 10	D7	11 (+1 pour M1)	1
RST 18	DF	11 (+1 pour M1)	1
RST 20	E7	11 (+1 pour M1)	1
RST 28	EF	11 (+1 pour M1)	1
RST 30	F7	11 (+1 pour M1)	1
RST 38	FF	11 (+1 pour M1)	1

**Exemple :** RST 28

**Remarque :** Cette instruction ne prend qu'un octet et elle est plus rapide qu'un CALL. Sur MSX, elle donne accès à 8 routines de la Main-ROM fréquemment appelées par l'interpréteur BASIC dont je donne les détails dans [le chapitre du BIOS de la ROM principale](#).

## Groupe 8 : Entrées/Sorties

Le CPU des MSX dispose de 256 ports d'entrée et autant de sortie. Ils permettent d'accéder aux périphériques ou certaines puces spécialisées. Voici les instructions disponibles sur le Z80 et le R800.

### IN sr, (C)

**Rôle :** Le port d'entrée dont le numéro est indiqué par le registre C est lu et, le résultat est mis dans le registre spécifié.

sr peut être A, B, C, D, E, H ou L. sr peut être omis !

**Opération :** sr = valeur lue au port indiqué par le registre C

**État du registre F après exécution après exécution de IN (C) :**

SF est mis à 1 si la valeur lue est négative.

ZF est mis à 1 si la valeur lue est zéro.

HF est mis à 0.

<sup>PF</sup>/<sub>VF</sub> est mis à 1 si le nombre de bit à 1 de la valeur lue est pair.

NF est mis à 1.

CF ne change pas.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
IN B, (C)	ED 40	12 (+2 pour M1)	3
IN C, (C)	ED 48	12 (+2 pour M1)	3
IN D, (C)	ED 50	12 (+2 pour M1)	3
IN E, (C)	ED 58	12 (+2 pour M1)	3
IN H, (C)	ED 60	12 (+2 pour M1)	3
IN L, (C)	ED 68	12 (+2 pour M1)	3
IN (C)	ED 70	12 (+2 pour M1)	3
IN A, (C)	ED 78	12 (+2 pour M1)	3

**Exemple :**

avant		après
C = A8h		C = A8h
port A8h = 03h	IN A, (C)	port A8h = 03h
A = ?		A = 03h

**Remarques :**

- Les indicateurs sont modifiés seulement lorsque le registre est omis.
- Voir [la liste des ports d'entrée/sortie](#) page 602 pour avoir un aperçu des ports utilisés sur MSX.

**Rôle :** Le port du port d'entrée v8 est lu et la valeur est mise dans l'accumulateur.

v8 est une valeur entre 0 et 255.

**Opération :** A = valeur lue au port spécifié.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
IN A, (v8)	DB v8	11 (+2 pour M1)	3

**Exemple :**

avant		après
A = ? port 79H = F3h	IN A, (79H)	A = F3h port 79H = F3h

**Remarques :**

- Les parenthèses sont logiques car c'est bien le contenu lu au port spécifié qui est transféré.
- Voir [la liste des ports d'entrée/sortie](#) page 602 pour avoir un aperçu des ports utilisés sur MSX.

**Rôle :** L'instruction **IND** lit port spécifié par le registre C et envoie le résultat à l'adresse pointée par HL. Ensuite, décrémente le registre B et le double registre HL (pour préparer l'entrée suivante). **INDR** fait pareil mais en répétant l'exécution jusqu'à ce que le registre B soit égale à zéro.

**Opération :**  $(HL) = (C)$  ;  $B = B - 1$  ;  $HL = HL \times 2$

**État du registre F après exécution :**

SF est indéterminé.

ZF est mis à 1 si  $B-1 = 0$ .

HF est mis à 1 si  $(HL) + ((C - 1) \& 255)$  est supérieur à 255.

$PF/VF$  est mis à 1 si  $((HL) + ((C - 1) \& 255)) \& 7$  xor B est une valeur paire.

NF est mis à 1.

CF est mis à 1 si  $(HL) + ((C - 1) \& 255)$  est supérieur à 255.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
IND	ED AA	16 (+2 pour M1)	2
INDR	ED BA	21 (+1 pour M1) si vrai 16 (+1 pour M1) si faux	2

**Exemple :**

avant		après
HL = 7050h		HL = 704Fh
B = 5		B = 4
(7050h) = ?	IND	(7050H) = A7h
C = A0h		C = A0h
port A0H = A7h		port A0H = A7h

**Remarques :**

- Le registre B sert de compteur d'octets lus. Cette instruction permet de transférer en mémoire une suite d'octets lus au port spécifié par le registre C à l'aide d'une boucle.
- Certains périphériques demandent un petit laps de temps entre deux lectures. Vous devez le connaître afin de ne pas lire les données trop rapidement.

**Rôle :** L'instruction INI lit le port spécifié par le registre C et envoie le résultat à l'adresse pointée par HL. Ensuite, décrémente le registre B et incrémente le double registre HL (pour préparer l'entrée suivante). INIR fait pareil mais en répétant l'exécution jusqu'à ce que le registre B soit égale à zéro.

**Opération :**  $(HL) = (\text{port } C)$  ;  $B = B - 1$  ;  $HL = HL + 1$

**État du registre F après exécution :**

SF est indéterminé.

ZF est mis à 1 si  $B - 1 = 0$ .

HF est mis à 1 si  $(HL) + ((C + 1) \& 255)$  est supérieur à 255.

$PF/VF$  est mis à 1 si  $((HL) + ((C + 1) \& 255)) \& 7$  xor B est une valeur paire.

NF est mis à 1.

CF est mis à 1 si  $(HL) + ((C + 1) \& 255)$  est supérieur à 255.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
INI	ED A2	16 (+2 pour M1)	2
INIR	ED B2	21 (+2 pour M1) si vrai 16 (+2 pour M1) si faux	2

**Remarque :** Certains périphériques demandent un petit laps de temps entre deux lectures. Vous devez le connaître afin de ne pas lire les données trop rapidement.

## OUT (C) , sr

**Rôle :** Le contenu registre spécifié après la virgule est envoyé sur le port de sortie dont le numéro est indiqué par le registre C.

sr peut être A, B, C, D, E, H ou L.

**Opération :** (port C) = sr

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
OUT (C) , B	ED 41	12 (+2 pour M1)	3
OUT (C) , C	ED 49	12 (+2 pour M1)	3
OUT (C) , D	ED 51	12 (+2 pour M1)	3
OUT (C) , E	ED 59	12 (+2 pour M1)	3
OUT (C) , H	ED 61	12 (+2 pour M1)	3
OUT (C) , L	ED 69	12 (+2 pour M1)	3
OUT (C) , A	ED 79	12 (+2 pour M1)	3

**Exemple :**

avant		après
H = 08h C = 97h port 97h = ?	OUT (C) , H	H = 08h C = 97h port 97h = 08h

**Remarque :** Voir [la liste des ports d'entrée/sortie](#) page 602 pour avoir un aperçu des ports utilisés sur MSX.

## OUT (v8) , A

**Rôle :** Envoyer le contenu de l'accumulateur sur le port de sortie dont le numéro est indiqué par v8.

v8 est un nombre entre 0 et 255.

**Opération :** (port v8) = A

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
OUT (v8) , A	D3 v8	11 (+1 pour M1)	2

**Exemple :**

avant		après
(port 98h) = ? A = 5	OUT (98h) , A	(port 98h) = 5 A = 5

**Rôle :** Le contenu de l'adresse pointée par le double registre HL est envoyé sur le port d'écriture indiqué par le registre C. Ensuite, le registre B et le double registre HL sont décrémentés (pour préparer l'envoi de la donnée suivante). L'opération est répétée automatiquement avec OTDR jusqu'à ce que B = 0.

**Opération :** (port C) = (HL) ; B = B - 1 ; HL = HL - 1

**État du registre F après exécution :**

SF est indéterminé.

ZF est mis à 1 seulement si B - 1 = 0 avec OUTD. Il est mis à 1 avec OTDR.

HF est mis à 1 si (HL) + L est supérieur à 255.

$PF/VF$  est mis à 1 si (((HL) + L) & 7) xor B) est valeur paire.

NF est mis à 1.

CF est mis à 1 si (HL) + L est supérieur à 255.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
OUTD	ED AB	16 (+2 pour M1)	2
OTDR	ED BB	21 (+2 pour M1) si vrai 16 (+2 pour M1) si faux	2

**Remarques :**

- B sert de compteur (8 bits). Cette instruction peut servir à écrire un bloc de données dans un périphérique. (La mémoire vidéo ou l'imprimante par exemple.)
- Certains périphériques demandent un petit temps entre deux écritures et dans ce cas, OTDR est parfois inutilisable.



**Rôle :** Le contenu de l'adresse pointée par le double registre HL est envoyé sur le port d'écriture indiqué par le registre C. Ensuite, le registre B est décrémenté et le double registre HL est incrémenté (pour préparer l'envoi de la donnée suivante). L'opération est répétée automatiquement avec OTIR jusqu'à ce que  $B = 0$ .

**Opération :** (port C) = (HL) ;  $B = B - 1$  ;  $HL = HL + 1$

**État du registre F après exécution :**

SF est indéterminé.

ZF est mis à 1 seulement si  $B - 1 = 0$  avec OUTI. Il est mis à 1 avec OTIR.

HF est mis à 1 si  $(HL) + L$  est supérieur à 255.

$^{PF}/_{VF}$  est mis à 1 si  $((((HL) + L) \& 7) \text{ xor } B)$  est valeur paire.

NF est mis à 1.

CF est mis à 1 si  $(HL) + L$  est supérieur à 255.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
OUTI	ED A3	16 (+2 pour M1)	2
OTIR	ED B3	21 (+2 pour M1) si vrai 16 (+2 pour M1) si faux	2

**Remarques :**

- B sert de compteur (8 bits). Cette instruction peut servir à écrire un bloc de données dans un périphérique. (La mémoire vidéo ou l'imprimante par exemple.)
- Certains périphériques demandent un petit temps entre deux écritures et dans ce cas, OTDR est parfois inutilisable.

## Groupe 9 : Interruptions et temps d'attente

Un certain nombre de ces instructions du Z80 ne sont pas utilisées sur le système MSX.

### DI

**Rôle :** Cette instruction désactive les interruptions « masquables » juste après son exécution en mettant IFF1 et IFF2 à 0. Dès lors, le CPU ignore les demandes d'interruption reçues par la broche  $\overline{\text{INT}}$ .

**Opération :** IFF1 = 0 ; IFF2 = 0

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
DI	F3	4 (+1 pour M1)	1

**Remarque :** Il est parfois nécessaire d'interdire les interruptions mais, n'oubliez pas de les ré-autoriser ensuite, sinon, vous n'aurez plus le contrôle de l'ordinateur, car il n'y aura plus de scrutation du clavier.

### EI

**Rôle :** Cette instruction sert à autoriser les interruptions « masquables ». En pratique, les interruptions ne sont pas autorisées dès l'exécution de EI mais juste après de l'instruction qui suit.

**Opération :** IFF1 = 1 ; IFF2 = 1

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
EI	FB	4 (+1 pour M1)	1

## HALT

**Rôle :** Suspendre l'exécution du programme jusqu'à la prochaine interruption ou une réinitialisation.

**Opération :** Exécute des NOP jusqu'à la prochaine interruption.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
HALT	76	4 (+1 pour M1)	1

**Remarque :** Cette instruction peut servir à générer des délais assez importants. La fréquence des interruptions sur MSX européen est de 50 Hz : Donc si vous faites une boucle qui exécute 50 fois HALT sur un MSX PAL européen, vous créer un délai d'environ 1 seconde.

## IM0

**Rôle :** Sélectionner le mode d'interruption 0. (Voir le paragraphe sur [les interruptions](#) pour les détails.)

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
IM0	ED 46	8 (+2 pour M1)	2

## IM1

**Rôle :** Sélectionner le mode d'interruption 1. (Voir le paragraphe sur [les interruptions](#) pour les détails.)

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
IM1	ED 56	8 (+2 pour M1)	2

**Remarque :** Seul ce mode est utilisé sur MSX.

## IM2

**Rôle :** Sélectionner le mode d'interruption 2. (Voir le paragraphe sur [les interruptions](#) pour les détails.)

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
IM2	ED 5E	8 (+2 pour M1)	2

## RETI

**Rôle :** RETI a été créé afin de permettre à un dispositif d'interruption externe de détecter son exécution pendant une interruption grâce à son code présent sur le BUS de donnée afin de relancer ce dispositif. Pour que tout ce passe bien, en général, EI doit être exécuté avant ou peu après le RETI. Cependant, sur MSX, vous pouvez considérer que cette instruction fait la même chose que RET car ces dispositifs d'interruption ne sont pas utilisés. La seule petite différence est son code machine et son temps d'exécution.

**Opération :**  $PC = (SP)$  ;  $SP = SP + 2$  ;  $IFF1 = IFF2$

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
RETI	ED 4D	14 (+2 pour M1)	2

**Remarque :** Même si un dispositif d'interruption de ce genre est utilisable en pratique, la norme dit que seul le mode 1, géré par la routine d'interruption en Main-ROM, doit être utilisé.

## RETN

**Rôle :** Sur MSX, vous pouvez considérer que cette instruction fait la même chose que RET. La seule petite différence est son code machine et son temps d'exécution.

**Opération :**  $PC = (SP)$  ;  $SP = SP + 2$  ;  $IFF1 = IFF2$

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
RETN	ED 45	14 (+2 pour M1)	2

## NOP

**Rôle :** Aucun effet.

**Opération :** Aucune.

**Code machine et temps d'exécution :**

Instruction	Code machine	Cycles d'horloge (Z80)	Cycles d'horloge (R800)
NOP	00	4 (+1 pour M1)	1

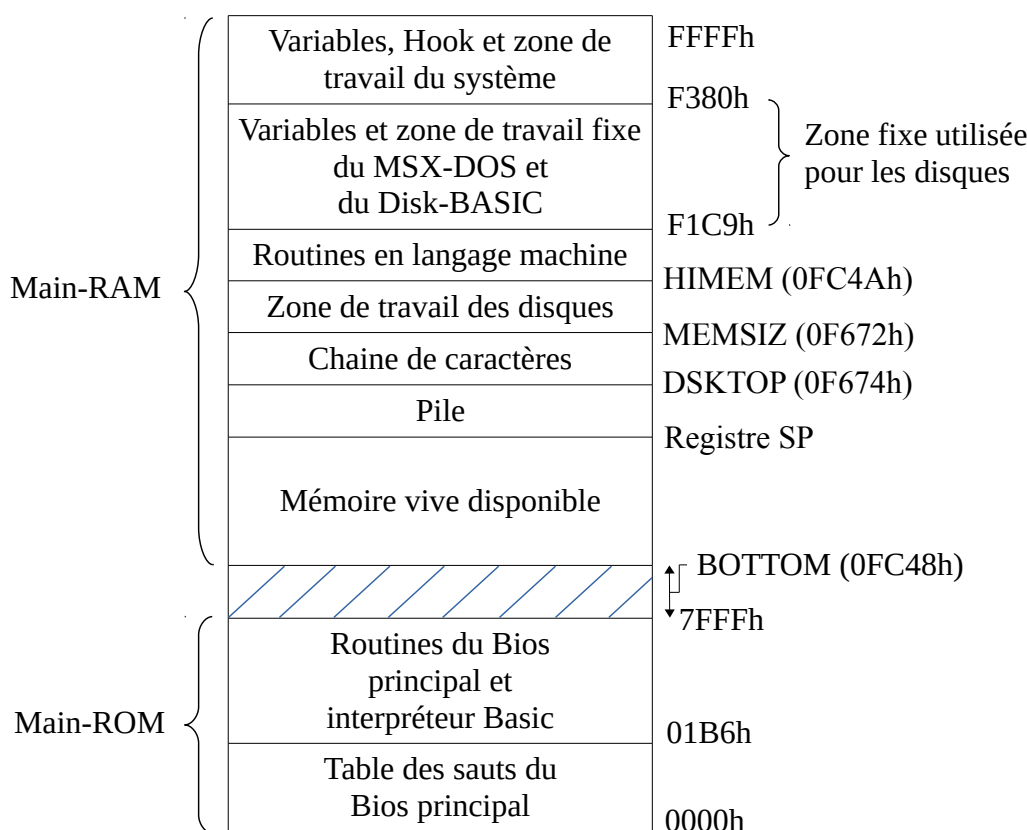
**Remarque :** Cette instruction est souvent utilisée pour produire de petits délais (NOP dure 4 cycles d'horloge, soit environ 1,11746 microseconde pour un z80 à 3,57Mhz).

# 3 Les Slot et le Memory Mapper

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

## 3.1 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 jusqu'à la moitié. 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. Cette adresse est indiquée par la variable système BOTTOM (0FC48h). Pour connaître la fin de mémoire vive disponible, il faudra lire la variable MEMSIZ (0F672h).

L'instruction CLEAR du basic permet de créer une zone « protégée » au dessus de celle de travail des disques afin d'y placer nos propres routines en langage machine. HIMEM (0FC4Ah) est spécifié par le deuxième paramètre de l'instruction CLEAR et la taille de la zone de la zone des variables qu'il y a entre MEMSIZ (0F672h) et DSKTOP (0F674h) est définie par le premier paramètre. A l'initialisation, la zone pour les routines en langage machine a une taille de 0 octet.

Sur les MSX ayant plus de 32Ko de mémoire vive, il est nécessaire de manipuler les Slot pour accéder au reste de la mémoire vive qui est inaccessible au Basic. Les trois paragraphes suivants expliquent comment manipuler les Slot.

### 3.2 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 16 bits, soit deux octets. Les adresses peuvent donc prendre n'importe quelle valeur entre 0 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 64 kilo octets (65536/1024).

Ce que « voit » le processeur



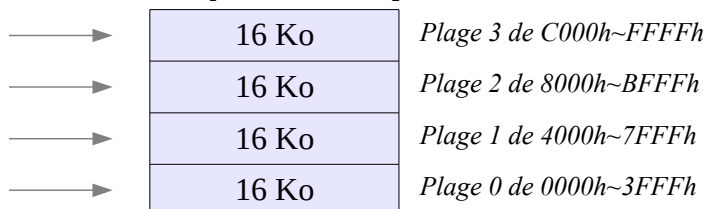
Mémoire centrale

Ainsi, quoiqu'il arrive, le Z80 ne pourra jamais adresser plus de 64 Ko de mémoire. Les concepteurs du MSX (ASCII) ont donc introduit un système, appelé « Slot », qui permet de multiplier la mémoire par jusqu'à 16, accessible par plage (Page en anglais) de 16 Ko sur lesquelles l'utilisateur peut y changer le contenu à volonté. Le dispositif de Slot se contrôle à l'aide de la puce PPI qui gère aussi le clavier.

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 Slot, 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).

### 3.3 Qu'est-ce qu'un Slot ?

Un Slot (mot anglais que l'on traduit par fente) est un emplacement contenant une ROM ou une RAM de 16 Ko. La sélection de ces emplacements est commandée par un registre qui commande la connexion vers le CPU via un dispositif comprenant en quelque sorte quatre « aiguillages » à quatre positions au lieu de les relier directement pour ne pas être limité à 64 Ko. Chacun de ses « aiguillages » peuvent être donc reliés à 16Ko de mémoire. Ces plages mémoire sont comprises entre les adresses 0000h et 3FFFh (plage 0), 4000h et 7FFFh (plage 1), 8000h et BFFFh (plage 2), et C000h et FFFFh (plage 3).

Il y a deux sortes de Slot, les Slot primaires et les Slot secondaires. Ces derniers sont une extension d'un Slot primaire qui fonctionnent de façon identique sauf que les registres sont accessibles différemment. Un MSX peut avoir jusqu'à 4 Slot primaires dont chacun peut être relié à 4 Slot secondaires au lieu d'une mémoire. Les Slot secondaires permettent de gérer 16 fois plus de mémoire.

Voici un exemple d'un MSX ayant tous ses Slot 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 qui a démarré sous basic en mode programmation (mode direct) :

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

Dès le démarrage d'un ordinateur MSX, le Slot de la Main-ROM est sélectionné sur toutes les plages mémoire. La Main-ROM est exécutée à l'adresse 0000h. Le système cherche la RAM dans les Slots puis s'installe (initialisation [des zones et des variables système](#) et des [Hook](#)). Une fois fait, il effectue les étapes suivantes :

- Appel aux Hook *H.STKE* (0FEDAh), *H.LOPD* (0FED5h) et *H.STKE* (0FEDAh).
- Recherche des ROMs et exécution si présentes
- Appel aux Hook *H.LOPD* (0FED5h) et *H.STKE* (0FEDAh).

Par exemple, voici la disposition juste après le démarrage d'un MSX1 Philips VG-8020/19 :

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

- Au démarrage sous Basic, les pages 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 Slot 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 le MSX2 VG-8235 de Philips :

Plage 3						MEM	
Plage 2						MEM	
Plage 1	MAIN					MEM	DISK
Plage 0	MAIN				SUB	MEM	
	0	1	2	3-0	3-1	3-2	3-3

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 Slot 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 E000h à FFFFh) dans n'importe quel Slot.
- Deux pages de mémoire morte (située de 0000h à 7FFFh) contenant le MSX-Basic version 1.0 et le Bios de la ROM principale (« Main-ROM ») dans le Slot 0 ou 0-0.
- 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.

MSX2+ :

- Quatre pages consécutives (64 Ko) de mémoire vive dans le même Slot et un Memory Mapper
- 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.



### 3.4 Utiliser les Slot

Toutes les routines du système code le numéro de Slot sur un octet de la manière suivante (au format F000SSPP) :

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
F	-	-	-	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).

F = Le bit 7 est un indicateur de type de Slot. Ce bit est à 1 lorsqu'il s'agit d'un Slot secondaire autrement, il s'agit d'un Slot primaire et les SS1 et SS0 ne doivent pas être pris en compte.

Les bits 6 à 4 peuvent varier suivant le contexte mais, lorsque c'est le cas, ils n'ont pas rapport direct avec les Slot.

Les registres des Slot primaires et secondaires sont codés ainsi :

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Slot sélectionné sur la plage 3		Slot sélectionné sur la plage 2		Slot sélectionné sur la plage 1		Slot sélectionné sur la plage 0	

La plage mémoire 0 va de 0000h à 3FFFh, la plage 1 de 4000h à 7FFFh, la plage 2 de 8000h à BFFFh et la plage 3 de C000h à FFFFh.

#### Variables systèmes liées aux Slot :

##### – **EXPTBL** (EXPanded slot TaBLe)

Adresse : 0FCC1h~0FCC4h en Main-RAM

Rôle : Le bit 7 de ces variables est un indicateur de Slot étendu pour chacun des Slot primaires. Il est à 1 lorsque le Slot primaire correspondant est étendu en Slot secondaires. Les autres bits sont à zéro. Comme la première variable à 0FCC1h (appelée aussi MNROM) sert à indiquer le Slot de la Main-ROM au format F000SSPP, ceci implique que la Main-ROM d'un MSX devrait toujours se trouver dans le Slot primaire 0, ou le Slot secondaire 0-0.

Note : Il existe une extension MSX2 pour MSX1 qui déplace la Main-ROM vers un autre Slot. Celle-ci modifie la variable MNROM (0FCC1h) dans la table EXPTBL pour indiquer le nouveau Slot de la Main-ROM au format F000SSPP. Cependant, ceci a pour effet qu'il n'est plus possible de déterminer si le Slot 0 est étendu ou pas. La variable MINROM (0FFF7h) est sensé remédier à ce problème mais cette extension modifie aussi MINROM.

– **SLTTBL** (SLoT TaBLe)

Adresse : 0FCC5h~0FCC8h en Main-RAM

Rôle : Le système y sauvegarde l'état des 4 registres des Slot secondaires de chaque Slot primaire (de 0 à 3). Attention, seuls les bits correspondants aux Slot étendus sont à prendre en compte (à l'aide des variables EXPTBL expliquées ci-dessus).

– **EXBRSA** (EXpanded Bios Rom slot Address)

Adresse : 0FAF8h en Main-RAM (MSX2 ou plus récent.)

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

– **MINROM** (MaIN-ROM slot)

Adresse : 0FFF7h en Main-RAM

Rôle : À partir du MSX2, cet octet contient le numéro de Slot qui contient la Main-ROM mais cette variable a été considérée obsolète par la suite.

– **SLTATR** (SLoT AttRibutes)

Adresse : 0FCC9h~0FD08h en Main-RAM

Rôle : 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 combinaisons. (Voir « [Développer un programme en cartouche](#) page 564 » au chapitre 14 pour la description)

– **SLTWRK** (SLoT WoRK)

Adresse : 0FD09h~0FD88h en Main-RAM

Rôle : SLTWRK est un tableau de 128 octets qui sert à réserver une zone travail en RAM pour les applications en ROM. (Voir « [Développer un programme en cartouche](#) page 564 » au chapitre 14 pour la description)

Routines du Bios liées aux Slot :

– **RDSLT** (ReaD SLoT)

Adresse : 0000Ch en Main-ROM

Rôle : Lecture d'une case mémoire dans un Slot.

– **WRSLT** (WRite SLoT)

Adresse : 00014h en Main-ROM

Rôle : Écriture dans une case mémoire dans un Slot.

– **CALSLT** (CALl SLoT)

Adresse : 0001Ch en Main-ROM

Rôle : Appel inter-Slot à une adresse.

– **ENASLT** (ENABle SLoT)

Adresse : 00024h en Main-ROM

Rôle : Sélection d'un Slot.

– **CALLF** (CALL Far)

Adresse : 00030h en Main-ROM

Rôle : Appel inter-Slot à une adresse.

– **RSLREG** (Read primary Slot REGister)

Adresse : 00138h en Main-ROM

Rôle : Lecture du registre des Slot primaires.

– **WSLREG** (Write Slot REGister)

Adresse : 0013Bh en Main-ROM

Rôle : Écriture dans le registre des Slot primaires.

– **SUBROM** (SUBROM call SUB-ROM)

Adresse : 0015Ch en Main-ROM

Rôle : Appel à une adresse dans la Sub-ROM.

– **EXTROM** (EXTROM call EXTernal ROM)

Adresse : 0015Fh en Main-ROM

Rôle : Appel à une adresse dans la Sub-ROM.

Ces routines n'appellent pas de commentaire particulier ([voir le Bios](#) page 130) 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 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 ».

### 3.5 Le Memory Mapper

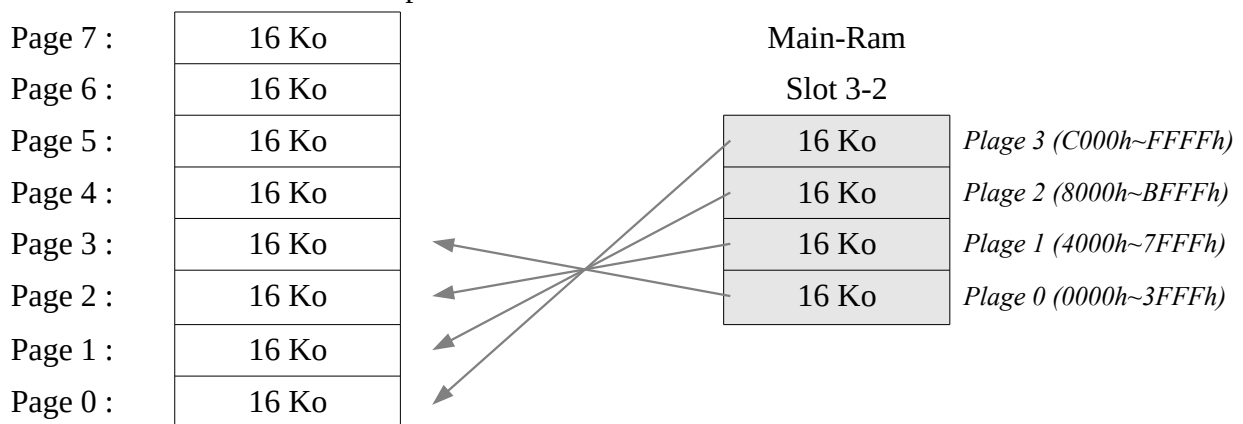
La plupart des MSX2 disponibles en France se trouvent équipés d'un dispositif perfectionné permettant de gérer des quantités assez importantes de mémoire vive pour l'époque (128 ou 256 Ko de RAM sur les modèles disponibles chez nous, mais pouvant aller en théorie jusqu'à 4 Mo par Slot).

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 d'un Slot.

Note : A l'initialisation du MSX, le système cherche la RAM dans les Slot primaires de 0 à 3 et ce sur les plages mémoire 8000h~BFFFh et C000h~FFFFh. Si un Slot primaire est étendu, il cherche dans les Slot secondaires avant de passer au Slot primaire suivant. Ainsi le système sélectionne la première RAM qu'il trouve pour chaque plage sans tenir compte du Memory Mapper, excepté le MSX turbo R qui prend sa RAM interne pour mémoire principale en mode R800. 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. Il est important de tenir compte de ces contraintes dues à l'évolution du standard.

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 Slot.

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 page 1, le port 0FEh pour (l'aviez-vous deviné ?) la page 2 et le port 0FFh pour la page 3.

0FCh = Page placée sur la plage 0 (0000h~3FFFh)

0FDh = Page placée sur la plage 1 (4000h~7FFFh)

0FEh = Page placée sur la plage 2 (8000h~BFFFh)

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

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 aux développeurs d'utiliser ces ports en lecture à cause du conflit provoqué par les bits non gérés. Certaines extensions ne gèrent pas du tout la lecture d'ailleurs.

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

mémoire centrale	Port E/S	Contenu du port	Page du Memory Mapper
Plage 0 (0000h~3FFFh)	0FCh	003h	page 3
Plage 1 (4000h~7FFFh)	0FDh	002h	page 2
Plage 2 (8000h~BFFFh)	0FEh	001h	page 1
Plage 3 (C000h~FFFFh)	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.

ASCII 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](#) page 524 pour les détails.)

## 4 La ROM principale et la ROM auxiliaire du système

Le système de tous les ordinateurs MSX se trouve dans une ROM de 32Ko. On l'appelle « Main-ROM » (ROM principale). Cette ROM contient le BIOS et l'interpréteur du Basic. Il y a aussi quelques octets qui donnent des informations sur le système. (Voir la liste ci-dessous.)

Ensuite, pour des MSX2 et les générations suivantes, une ROM auxiliaire de 16Ko appelée « Sub-ROM » (ROM auxiliaire) a été ajoutée. Celle-ci contient aussi un BIOS et des instructions supplémentaires du Basic. La ROM auxiliaire est décrite plus loin.)

### 4.1 Information système

Voici la liste et la description des informations fournies par la ROM de l'ordinateur MSX utilisé.

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

Rôle : Les octets 00004h et 00005h contiennent l'adresse du début de la table des caractères à partir du premier caractère au code ASCII en ROM. Sur un MSX arabe, ils correspondent à la police internationale. Pour connaître l'adresse de la police arabe, lire les octets 0003Ah à 0003Ch (Slot+adresse) de l'« Arabic ROM ».

#### 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](#) ».

## 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 »).

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0002Bh :	V <sub>Sync</sub>	Date format			Jeu de caractère			

Les bits 0 à 3 indiquent le types de Jeu de caractère en Main-ROM :

0 = Japonais ; 1 = international ; 2 = Pays soviétiques.

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 de rafraîchissement de l'écran par défaut :

0 = 60 Hertz ; 1 = 50 Hertz.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0002Ch :	SM	CDR	BRN	MEI	Type de clavier			

Type de clavier : 0 = Japonais ; 1 = Anglais (U.S.) ; 2 = Français,  
3 = Anglais (U.K.) ; 4 = Allemand ;  
5 = Pays soviétiques ; 6 = Espagnol.

MEI : Indique le mode d'écran par défaut au démarrage du MSX.

0 = SCREEN 0 ; 1 = SCREEN 1.

BRN : Indique le caractère utilisé comme borne dans l'instruction PRINT USING.

0 = « & » ; 1 = « \ ».

CDR : Indique le caractère de remplacement de l'instruction PRINT USING.

0 = « @ » ; 1 = « & ».

SM : Indique le quel symbole monétaire est utilisé.

0 = Autre que dollar ; 1 = Dollar.

Notes :

- Les ordinateurs MSX coréens et soviétiques ne respectent pas le standard pour le jeu de caractère et le type de clavier.
- Tableau des différences par pays :

Pays	Standard TV	Format de la date	Mode d'écran initial	Caractères spécifiques utilisés par le BASIC		
				Remplacement	Borne	Symbole monétaire
Allemagne	PAL (50Hz)	JJ/MM/AA	SCREEN0	@	&	\$
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	&	\	¥
Koweït, Egypt	PAL (50Hz)	JJ/MM/AA	SCREEN0/1	@	&	\$
Pays soviétiques	NTSC (60Hz)	MM/JJ/AA	SCREEN0	@	&	\$
U.K.	PAL (50Hz)	JJ/MM/AA	SCREEN0	@	&	£
USA	NTSC (60Hz)	MM/JJ/AA	SCREEN0	@	&	\$



#### 0002Dh (Main-ROM)    **MSXVER** (« MSX VERsion »)

Rôle :            Version du MSX. Cet octet contient 0 si il s'agit d'un MSX1, 1 pour un MSX2, 2 pour un MSX2+ ou 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 le MSX-MIDI est présent en interne.

Le bit 0 de cette adresse est à 1 si le MSX-MIDI est intégrée.

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

#### 0002Fh\* (Main-ROM)

Rôle :            Réservé.

## 4.2 Introduction au BIOS

Le Bios (Basic Input/Output System) est un ensemble de routines conçues afin de permettre au programmeur d'accéder à un matériel (« hardware ») sans que celui-ci soit toujours le même grâce à sa table des sauts. Le BIOS se compose donc de routines en ROM qui permettent d'accéder de la même façon à de plus ou moins différents matériels. Prenons l'exemple du clavier du MSX qui est géré par un circuit spécialisé, le PPI 8255. Un jour, il se peut qu'un constructeur MSX sorte un modèle avec clavier détaché à liaison infrarouge qui sera probablement contrôlé par une puce autre qu'un PPI 8255. Dans ce cas, si un programme, faisant des accès directs au PPI, ne fonctionnera pas correctement sur ce nouveau modèle. Au contraire, si le programme passe par la routine « lire l'état du clavier », il tournera très bien même avec le clavier à liaison infrarouge. Le BIOS concilie compatibilité des logiciels avec évolution du matériel.

Note : Avec le temps, grâce aux puces programmables, il a été possible par exemple de simuler un PPI tout en utilisant un clavier au fonctionnement différent. De ce fait, les ports E/S du PPI sont devenus standards.

## 4.3 Table des sauts du Bios de la ROM principale (Main-ROM)

Les routines du Bios pouvant avoir une position et une taille variable d'un MSX à l'autre, il a donc été nécessaire de créer une table de sauts vers chaque routine afin que tous les logiciels fonctionnent quelque soit le MSX utilisé. Sur la page suivante vous trouverez la liste de la table des sauts vers 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 des remarques sur la routine elle-même, sur son fonctionnement ou les différences entre chaque version de MSX lorsque nécessaire.

## 00000h (Main-ROM)     **STARUP** (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 mode d'écran et la couleur de bordure initialisées par défaut peuvent être différentes selon le pays.

Cette routine est la première à être exécutée à l'allumage du MSX. Voici un résumé de ce que fait la routine.

### MSX1 :

- La Main-ROM est placée sur la plage mémoire 0000h~3FFFh et 4000h~7FFFh.
- La routine cherche la RAM dans les Slot primaires de 0 à 3 sur les plages mémoire 8000h~BFFF et C000h~FFFFh. Si un Slot primaire est étendu, elle cherche dans les Slot secondaires de 0 à 3 avant de passer au Slot primaire suivant. Ainsi la routine sélectionne la première RAM qu'elle trouve sur chaque plage sans tenir compte du Memory Mapper.
- Initialise la zone de travail, les variables et les Hooks.
- Cherche une éventuelle Rom dans les Slot primaires de 0 à 3 sur les plages mémoire 4000h~7FFF et 8000h~BFFFh pour l'exécuter. (Lorsqu'une Rom est trouvée sur la plage 4000h~7FFFh, la plage 8000h~BFFFh contiendra de la RAM si le MSX a 32Ko ou plus. Lorsqu'une Rom est trouvée sur la plage 8000h~BFFFh, la plage 4000h~7FFFh contiendra la Main-ROM).
- Remplace la Main-ROM sur la plage 4000h~7FFFh et démarre le Basic.

### MSX2 :

Procède de la même façon que les MSX1 sauf que les registres du Memory Mapper sont initialisés comme indiqué au paragraphe sur [le Memory Mapper](#).

### MSX2+ :

Procède de la même façon que les MSX2 sauf que la routine prend en compte si il s'agit du démarrage de l'ordinateur ou d'un redémarrage. Dans le second cas, le logo MSX ne sera pas affiché. (Le bit 5 du port E/S F4h est utilisé pour cela.)

### MSX turbo R :

Procède de la même façon que les MSX2+ sauf que la RAM interne sera sélectionnée d'office en mode R800 parce que la RAM interne est cadencée à 7,159 MHz au lieu de 3,579545 pour les extensions.

## 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, exécute la routine appelée par CHRGT (00010h en Main-ROM) sinon, exécute la routine de traitement d'erreur de l'interpréteur du Basic.
- 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 CF passe à 1 si le caractère est un chiffre ;  
l'indicateur ZF passe à 1 si le caractère est 00h ou 3Ah (fin d'instruction).
- Modifie : AF, HL.
- Notes : - Cette routine est utilisée par l'interpréteur Basic. (RST 8)  
- Attention, la Main-Rom doit être sélectionnée sur la plage 4000h~7FFFh pour appeler cette routine.

## 0000Ch (Main-ROM)     **RDSL** (« 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.
- Notes : - Les interruptions sont désactivées par la routine.  
- Cette routine existe aussi sous MSX-DOS.

## 00010h (Main-ROM)     **CHRGTR** (« CHaracteR GeTteR »)

- Rôle :            Récupération d'un caractère ou d'un code dans un programme Basic.
- Entrée :        HL = Adresse actuelle. (Pointeur)
- Sortie :        HL = Adresse du caractère récupéré.  
                  A = Caractère ou chiffre récupéré.  
                  F = Indicateur ZF à 1 si il s'agit d'un code de fin de ligne (00h ou « : »).  
                  Indicateur CF 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).  
                  - Appel au Hook H.CHRG (0FF48h).  
                  - Cette routine est utilisée par l'interpréteur Basic. (RST 10)

## 00014h (Main-ROM)     **WRSLT** (« WRite SLot »)

- Rôle :            Ecriture d'un octet dans le Slot spécifié.
- Entrée :        HL = Adresse de l'octet à écrire (de 0000h à BFFFh).  
                  E = Donnée à écrire.  
                  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 :      AF, BC, D.
- Notes :        - Les interruptions sont désactivées puis rétablies par la routine.  
                  - Cette routine peut aussi être appelée sous MSX-DOS.

## 00018h (Main-ROM)     **OUTDO** (« OUT DO »)

- Rôle :            Sortie d'un caractère vers le fichier du périphérique actuel ou l'imprimante.
- Entrée :        A = Code ASCII du caractère à sortir.  
                  PRTFLG (0F416h) = Doit être différent de 0 pour envoyer à l'imprimante.  
                  PTRFIL (0F864h) = Adresse du fichier dans lequel le caractères doit être envoyé.  
                  Mettre 0000h pour envoyer le caractère à l'imprimante.
- 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 :        Dépend de la routine appelée.
- Modifie :      AF, IX, IY, registres auxiliaires.
- Notes :        - Les interruptions sont désactivées par la routine.  
                - L'appel à un autre Slot étendu du même Slot primaire que celui où se trouve le Bios n'est pas possible.  
                - Cette routine existe aussi sous MSX-DOS.

Exemple d'utilisation en assembleur :

```
                  ; Appeler la routine BEEP du Bios en Main-ROM sous MSX-DOS

CALSLT    equ 001ch                    ; Routine d'appel inter-Slot
EXPTBL    equ 0fcc1h                  ; Slot de la Main-ROM
BEEP      equ 000c0h                  ; Emission d'un bip sonore

BeepSound:
          ld iy, (EXPTBL-1)    ; Emplacement de la Main-ROM
          ld ix, BEEP           ; Adresse de la routine BEEP
          call CALSLT          ; Appel la routine BEEP
          ret
```

Second exemple d'utilisation en assembleur :

```
                  ; Passage du MSX-DOS au Basic

MASTERS    equ 0F348h                ; Slot de la Disk-ROM maitre
BASENT     equ 04022h                ; Adresse de la routine BASENT
CALSLT     equ 0001ch                ; Routine d'appel inter-Slot

GotoBasic:
          ld iy, (MASTERS-1)    ; Emplacement de la Disk-ROM
          ld ix, BASENT          ; Adresse de la routine BASENT
          jp CALSLT            ; Passe sous Basic
```

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

Rôle : Comparaison de HL avec DE.

Entrée : HL = Premier nombre.  
DE = Deuxième nombre.

Sortie : F = Indicateur CF passe à 1 si HL < DE ;  
indicateur ZF passe à 1 si HL = DE.

Modifie : AF.

Note : Pour appelez cette routine, assurez-vous que la Main-ROM soit présente sur la plage 04000h~05FFF et utilisez l'instruction `RST 20` du Z80. La routine est utilisée par l'interpréteur Basic.

## 00024h (Main-ROM)     **ENASLT** (« ENable Slot »)

Rôle : Sélection d'un Slot.

Entrée : H = 2 bits de poids fort indiquent la plage sur laquelle sélectionner le Slot.  
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.

## 00028h (Main-ROM)     **GETYPR** (« GET the TYPE of decimal acumulator »)

Rôle : Détermination du type du nombre à DAC (« Decimal Acumulator »).

Entrée : Rien.

Sortie : A = FFh si entier, 0 si chaine de caractère, 1 simple précision, 5 si double précision.  
F = SF\*, CF,  $PF/VF$  à 1 et ZF à 0 si chiffre entier ;  
SF, ZF,  $PF/VF$ \* à 0 et CF à 1 si chiffre simple précision ;  
SF, ZF, CF\* à 0 et  $PF/VF$  à 1 si chiffre double précision ;  
CF, ZF\*,  $PF/VF$  à 0 et SF à 1 si chaine de caractères.  
\* Indicateur sur lequel se baser pour connaître le type.

Modifie : AF.

Notes : - Cette routine est utilisé par l'interpréteur Basic. (RST 28)  
- Attention, la Main-Rom doit être sélectionnée sur la plage 4000h~7FFFh pour appeler cette routine.

### 00030h (Main-ROM)     **CALLF** (« CALL Far »)

- Rôle :            Appel inter-Slot à une adresse.
- Entrée :          Paramètres de la routine à appeler.
- Sortie :          Dépend de la routine appelée.
- Modifie :        AF, IX, IY, registres auxiliaires, et les registres modifiés par la routine appelée.
- Notes :          - CALLF diffère de CALSTL dans la manière d'appeler la routine. Pour appeler CALLF, il faut utiliser l'instruction RST 30 de la façon suivante.
- ```
rst 30
db Numéro du Slot (F000SSPP)
dw Adresse à appeler
```
- Les interruptions sont désactivées par la routine.
- L'appel à un autre Slot étendu du même Slot primaire que celui où se trouve le Bios n'est pas possible.
- Cette routine existe aussi sous MSX-DOS.

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

- Rôle :            Routine des interruptions masquables.
- Entrée :          Rien.
- Sortie :          Rien.
- Modifie :        Rien.
- Notes :          - Cette routine met tous les registres du CPU dans la pile dans l'ordre HL, DE, BC, AF, HL', DE', BC', AF', IY, IX. Juste après, elle appelle le Hook H.KEYI (0FD9Ah), puis elle lit le port 99h du VDP (registre 0 par défaut), met CF du CPU à 0 et appelle le Hook H.TIMI (0FD9Fh). Ensuite,
- Appel à la Sub-ROM pour les conversions Roma-Kana sur MSX2 ou plus récent.
- Cette routine est appelé à chaque interruption. (RST 38)

### 0003Bh (Main-ROM)     **INITIO** (« INITialize Input/Output »)

- Rôle :            Initialiser les périphériques.
- 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.  
Note :        Accès au VDP.

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

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

### 00044h (Main-ROM)    **ENASCR** (« ENable SCReen display »)

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

### 00047h (Main-ROM)    **WRTVDP** (« WRiTe VDP »)

Rôle :            Écriture dans un [registre de contrôle du VDP](#).  
Entrée :        C = Numéro du registre du VDP (0~7 sur MSX1 ; 0~23 et 32~46 sur MSX2 ; 0~25 et 32~46 sur MSX2+ et Turbo R).  
                  B = Valeur à écrire.  
Sortie :        La variable système correspondante au registre est actualisée.  
Modifie :      AF, B.  
Note :        Appel à la Sub-ROM sur MSX2 ou plus récent si le bit EV du registre 0 du VDP est modifié ou si le numéro de registre est supérieur à 7.

### 0004Ah (Main-ROM)    **RDVRM** (« ReaD VRaM »)

Rôle :            Lecture d'un octet en 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 (00174h en Main-ROM).

## 0004Dh (Main-ROM)    **WRTVRM** (« WRiTe VRaM »)

- Rôle :            Écriture d'un octet en 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 :            Définit l'adresse en VRAM où doit s'effectuer la lecture de données.
- Entrée :        HL = Adresse source en VRAM (00000h~03FFFh).
- Sortie :        Rien.
- Modifie :      AF.
- Notes :        - Une fois l'adresse définie, chaque lecture au port de lecture de donnée du VDP incrémentera automatiquement l'adresse source dans le VDP.  
                  - Routine MSX1. Pour accéder à une adresse supérieure à 03FFFh, utiliser la routine NSETRD (0016Eh en Main-ROM).

Exemple :

```
;Lire 4 octets en VRAM

VDP.DR equ 0006h
SETWRT equ 0050h

        ld    hl,0           ;Adresse en VRAM
        call SETRD

        ld    hl,DATA
        ld    b,4           ;Nombre de donnée à lire
        ld    a,(VDP.DR)    ;Port de lecture de donnée
        ld    c,a

LOOP:   ini                     ;Vous pouvez remplacer ces deux
        jr    nz,LOOP        ;lignes par OTIR sur MSX2/2+/turbo R
        ret

DATA:   db    0,0,0,0        ;Les données lues sont stockées ici
```

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

- Rôle :            Définit l'adresse en VRAM où doit s'effectuer l'écriture de données.
- Entrée :          HL = Adresse de destination en VRAM (00000h~03FFFh).
- Sortie :          Rien.
- Modifie :        AF.
- Notes :          - Une fois l'adresse définie, chaque écriture au port d'écriture de donnée du VDP incrémentera automatiquement l'adresse de destination dans le VDP.
- Routine MSX1. Pour accéder à une adresse supérieure à 03FFFh, utiliser la routine NSTWRT (00171h en Main-ROM).

### Exemple :

```
;Afficher "Hello!" en haut à gauche de l'écran (SCREEN1)

VDP.DW equ 0007h
SETWRT equ 0053h
T32NAM equ 0F3BDh      ;Table des caractères en SCREEN1

        ld    hl,(T32NAM)
        call  SETWRT

        ld    hl,TXT
        ld    b,7        ;Longueur du text
        ld    a,(VDP.DW) ;Port d'écriture de donnée
        ld    c,a

HELLO:
        outi                ;Vous pouvez remplacer ces deux
        jr    nz,HELLO      ;lignes par OTIR sur MSX2/2+/turbo R
        ret

TXT:
        db    "Hello!"
```

## 00056h (Main-ROM)     **FILVRM** (« FIL VRaM »)

- Rôle :            Remplissage de la mémoire vidéo avec une donnée.
- Entrée :          HL = Adresse de début (00000h~03FFFh).
- BC = Longueur.
- A = Donnée.
- Sortie :          Rien.
- Modifie :        BC.
- Notes :          - Routine MSX1. Pour accéder aux adresses supérieures à 03FFFh, utiliser la routine BIGFIL (0016Bh en Main-ROM).
- Dans les modes d'écrans autres que MSX1. La variable ACPAGE (0FAF6h) indique la page dans laquelle le remplissage doit s'effectuer.

### 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 (0000h~3FFFh ou FFFFh).  
                  DE = Adresse de destination en RAM centrale (0000h~FFFFh).  
                  BC = Longueur du bloc à transférer.

Sortie :        Rien.

Modifie :      Tout.

Note :         Dans les modes d'écrans autres que MSX1. La variable ACPAGE (0FAF6h) indique la page depuis laquelle le transfert doit s'effectuer.

### 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 (0000h~FFFFh).  
                  DE = Adresse de destination en VRAM (0000h~3FFFh ou FFFFh).  
                  BC = Longueur du bloc à transférer.

Sortie :        Rien.

Modifie :      Tout.

Note :         Dans les modes d'écrans autres que MSX1. La variable ACPAGE (0FAF6h) indique la page dans laquelle le transfert doit s'effectuer.

### 0005Fh (Main-ROM)     **CHGMOD** (« CHange screen MODe »)

Rôle :            Changement de mode d'écran (SCREEN X).

Entrée :        A = Mode d'écran désiré (0~3 ou 0~8 / 12, 9 seulement sur les MSX2 coréens).  
                  LINL32 (0F3AFh) = Nombre de caractères par lignes pour le SCREEN 1.  
                  LINL40 (0F3AEh) = Nombre de caractères par lignes pour le SCREEN 0.

Sortie :        SCRMOD (0FCAFh) = Nouveau mode d'écran.

Modifie :      Tout.

Notes :        - 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.  
                  - Voir le registre 25 à la page 279, pour mettre en mode d'écran 10/11 ou 12.

## 00062h (Main-ROM)     **CHGCLR** (« CHanGe CoLoR »)

- Rôle :            Change les couleurs de l'écran spécifié avec les valeurs par défaut.
- Entrée :          A = Mode d'écran.  
                    FORCLR (0F3E9h) = Couleur de texte.  
                    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 CHGCLRS (00111h) de la Sub-ROM.

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

- Rôle :            Réservé pour la 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)     **CLRSPR** (« CLeaR Sprite »)

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

## 0006Ch (Main-ROM)    **INITXT** (« INIitalize TeXT of 40/80 columms mode »)

- Rôle :            Initialisation du mode SCREEN 0.
- Entrée :        TXTNAM (0F3B3h) = Adresse de la table des positons de caractère.  
                  TXTCOL (0F3B5h) = Adresse de la table des couleurs de caractères. (80 colonnes)  
                  TXTCGP (0F3B7h) = Adresse de la table des formes de caractère.  
                  LINL40 (0F3AEh) = Nombre de caractères par lignes.  
                  FORCLR (0F3E9h) = Couleur de texte.  
                  BAKCLR (0F3EAh) = Couleur de fond.
- Sortie :        Rien.
- Modifie :      Tout.
- Notes :        - Dans la plupart des cas, il n'est pas nécessaire de définir TXTNAM, etc. Le MSX initialise leur valeur à l'allumage.  
                  - 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** (« INIialize Text of 32 columns mode »)

- Rôle :            Initialisation du mode SCREEN 1.
- Entrée :        T32NAM (0F3BDh) = Adresse de la table des positons de caractère.  
                  T32COL (0F3BFh) = Adresse de la table des couleurs de caractères.  
                  T32CGP (0F3C1h) = Adresse de la table des formes de caractères.  
                  T32ATR (0F3C3h) = Adresse de la table des attributs de Sprite.  
                  T32PAT (0F3C5h) = Adresse de la table des formes de Sprite.  
                  LINL32 (0F3AFh) = Nombre de caractères par lignes.  
                  FORCLR (0F3E9h) = Couleur de texte.  
                  BAKCLR (0F3EAh) = Couleur de fond.  
                  BDRCLR (0F3EBh) = Couleur de bordure.
- Sortie :        Rien.
- Modifie :      Tout.
- Notes :        - Dans la plupart des cas, il n'est pas nécessaire de définir T32NAM, etc. 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 positons de motif.  
                 GRPCOL (0F3C9h) = Adresse de la table des couleurs de motif.  
                 GRPCGP (0F3CBh) = Adresse de la table des formes de motif.  
                 GRPATR (0F3CDh) = Adresse de la table des attributs de Sprite.  
                 GRPPAT (0F3CFh) = Adresse de la table des formes de Sprite.  
                 FORCLR (0F3E9h) = Couleur de texte ou de tracé.  
                 BAKCLR (0F3EAh) = Couleur de fond.  
                 BDRCLR (0F3EBh) = Couleur de bordure.
- Sortie :        Rien.
- Modifie :      Tout.
- Notes :        - Dans la plupart des cas, il n'est pas nécessaire de définir GRPNAM, etc. 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 positons de motif.  
                 MLTCOL (0F3D3h) = Adresse de la table des couleurs de motif.  
                 MLTCGP (0F3D5h) = Adresse de la table des formes de motif.  
                 MLTATR (0F3D7h) = Adresse de la table des attributs de Sprite.  
                 MLTPAT (0F3D9h) = Adresse de la table des formes de Sprite.  
                 FORCLR (0F3E9h) = Couleur de texte ou de tracé.  
                 BAKCLR (0F3EAh) = Couleur de fond.  
                 BDRCLR (0F3EBh) = Couleur de bordure.
- Sortie :        Rien.
- Modifie :      Tout.
- Notes :        - Dans la plupart des cas, il n'est pas nécessaire de définir MLTNAM, etc. 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 positons de caractère.  
                    TXTCGP (0F3B7h) = Adresse de la table des formes de caractère.
- Sortie :          Rien.
- Modifie :        Tout.
- Notes :          - Dans la plupart des cas, il n'est pas nécessaire de définir TXTNAM et TXTCGP.  
                    Le MSX initialise leur valeur à l'allumage.

### 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 Sprite.  
                    T32PAT (0F3C5h) = Adresse de la table des formes de Sprite.
- Sortie :          Rien.
- Modifie :        Tout.
- Notes :          - Dans la plupart des cas, il n'est pas nécessaire de définir T32NAM, etc. Le MSX  
                    initialise leur valeur à l'allumage.

### 0005Eh (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 Sprite.  
                    GRPPAT (0F3CFh) = Adresse de la table des formes de Sprite.
- Sortie :          Rien.
- Modifie :        Tout.
- Notes :          - Dans la plupart des cas, il n'est pas nécessaire de définir GRPNAM, etc. Le MSX  
                    initialise leur valeur à l'allumage.



### 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 Sprite.  
                  MLTPAT (0F3D9h) = Adresse de la table des formes de Sprite.

Sortie :        Rien.

Modifie :      Tout.

Notes :        - Dans la plupart des cas, il n'est pas nécessaire de définir MLTNAM à MLTPAT.  
                  Le MSX initialise leur valeur à l'allumage.

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

Rôle :            Donne l'adresse de la table des formes (patrons) de Sprite en VRAM.

Entrée :        A = Numéro du Sprite.

Sortie :        HL = Adresse cherchée.

Modifie :      AF, DE, HL.

### 00087h (Main-ROM)     **CALATR** (« CALculate ATtRIBUTE table address »)

Rôle :            Donne l'adresse du début des attributs du Sprite indiqué.

Entrée :        A = Numéro du Sprite.

Sortie :        HL = Adresse cherchée.

Modifie :      AF, DE, HL.

### 0008Ah (Main-ROM)     **GSPSIZ** (« Get Sprite SIZE »)

Rôle :            Indique la taille actuelle de la matrice des Sprite.

Entrée :        Rien.

Sortie :        A = Largeur de la matrice (8 ou 16).  
                  F = L'indicateur CF est mis à 1 si la taille des Sprite est 16x16 ; 0 si 8x8.

Modifie :      AF.

## 0008Dh (Main-ROM)    **GRPPRT** (« GRaPhic PRinT »)

Rôle :            Affichage d'un caractère dans les mode SCREEN 2 à 12.

Entrée :           A = Code ASCII du caractère 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.

                  LOGOPR (0FB02h) = Code d'opération logique (pour les modes graphiques de  
                                         SCREEN 5 à 12).

Sortie :           Rien.

Modifie :        Rien.

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

## 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 :
  - Voir le [chapitre 8 sur le PSG](#) pour plus de détails sur les registres.
  - Les interruptions sont coupées puis réactivées par la routine.
  - Voici une petite descriptions des registres de contrôle 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) |       |                        |                                        |                                     |                      |        |        |

## 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.
- Notes :
  - Voir le [chapitre 8 sur le PSG](#) pour plus de détails sur les registres.
  - La routine ne coupe pas les interruptions.

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

Rôle :            Joue le MML pour PLAY en tâche de fond.  
Entrée :        Rien.  
Sortie :        Rien.  
Modifie :      Tout.  
Note :        Ne joue pas si un MML est déjà en cours de lecture.

### 0009Ch (Main-ROM)     **CHSNS** (« CHeck ScaN keyboard buffer Status »)

Rôle :            Vérification l'état du cache du clavier.  
Entrée :        Rien.  
Sortie :        F = L'indicateur ZF passe à 0 si il y a un caractère dans le cache du clavier.  
Modifie :      AF.

### 0009Fh (Main-ROM)     **CHGET** (« CHaracter GET »)

Rôle :            Attente d'un caractère tapé au clavier et retour avec son code.  
Entrée :        Rien.  
Sortie :        A = Code ASCII du caractère.  
Modifie :      AF.  
Notes :        - Appel au Hook H.CHGE (0FDC2h).  
                  - Les variables SCNCNT (0F3F7h) et REPCNT (0F3F7h) permettent de régler les intervalles de scrutations du clavier.  
                  - Cette routine prend en compte les [codes de contrôle](#) et [d'échappement](#).

### 000A2h (Main-ROM)     **CHPUT** (« CHaracter outPUT »)

Rôle :            Sortie d'un caractère sur écran en mode texte.  
Entrée :        A = Code du caractère.  
                  CSRX (0F3DDh) = Abscisse du curseur.  
                  CSRY (0F3DCh) = Ordonnée du curseur.  
Sortie :        CSRX et CSRY sont actualisés.  
Modifie :      Rien.  
Notes :        - Appel au Hook H.CHPU (0FDA4h).  
                  - Cette routine prend en compte les [codes de contrôle](#) et [d'échappement](#).

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

Rôle :            Sortie d'un caractère sur imprimante.  
Entrée :        A = Code du caractère.  
Sortie :        F = Indicateur CF 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 ZF 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 :            Permet de gérer les caractères obtenus avec la routine CHGET (0009Fh) pour une éventuelle conversion des caractères graphiques étendus.  
Entrée :        A = Code d'un caractère ou 1 pour annoncer un caractère graphique étendu.  
Sortie :        GRPHED (0FCA6h) est mis à 1 lorsque le code entrée est 1.  
                  F = Si l'indicateur CF est à 0, c'est que le code entrée est 1.  
                  Les indicateurs CF et ZF sont à 1 lorsque que le code est un caractère étendu.  
                  L'indicateur CF est à 1 et ZF à 0 lorsque que le code un caractère ordinaire.  
Modifie :      AF.  
Notes :        - GRPHED (0FCA6h) est pris en compte pour indiquer si il s'agit d'un caractère graphique étendu (compris entre 65 et 95) ou pas.  
                  - Voir les [jeux de caractères MSX](#) pour connaître les caractères graphiques correspondants.

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

- Rôle : Place le curseur à la ligne suivante puis stocke les caractères entrés au clavier dans le cache BUF jusqu'à ce que la touche STOP ou RET soit pressés.
- Entrée : Rien.
- Sortie : HL = 0F55Eh (BUF)  
F = CF est mis à 1 si c'est la touche STOP qui a été pressée.  
Le cache BUF (0F55Eh) contient les caractères entrés au clavier.  
La table LINTTB (0FBB2h) est actualisée.
- 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 cache BUF jusqu'à ce que la touche STOP ou RET soit pressée.
- Entrée : Rien.
- Sortie : HL = 0F55Eh (BUF)  
F = CF est mis à 1 si c'est la touche STOP qui a été pressée.  
Le cache BUF (0F55Eh) contient les caractères entrés au clavier .  
La table LINTTB (0FBB2h) est actualisée.
- Modifie : Tout.
- Notes : - Appel au Hook H.INLI (0FDE5h).  
- Par rapport à PINLIN, cette routine ne prend en compte que les caractères placés derrière de curseur. L'invite est utilisable.

### 000B4h (Main-ROM)    **QINLIN** (« Question 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 cache -1.  
F = Indicateur CF à 1 si un ^C a eu lieu.
- Modifie : Tout.
- Note : Appel au Hook H.QINL (0FDE0h).

### 000B7h (Main-ROM)    **BREAKX** (« BREAK X »)

Rôle :            Teste si les touches CTRL+STOP sont pressées.  
Entrée :        Rien.  
Sortie :        F = Indicateur CF sera à 1 si les touches CTRL+STOP sont pressées.  
Modifie :      AF

Exemple d'utilisation :

```
LOOP:  call BREAKX
        jr  nc, LOOP
        ret
```

### 000BAh (Main-ROM)    **ISCNTC** (« IS CoNTrol C ? »)

Rôle :            Vérifie si les touches CTRL+STOP ou STOP sont pressées ou non afin de stopper ou suspendre l'exécution en cours du programme Basic.  
Entrée :        Rien.  
Sortie :        Si les touches CTRL+STOP sont pressées et que BASROM (0FBB1h) = 0, l'exécution de l'interpréteur Basic est interrompu; si seulement STOP est pressé, l'exécution est suspendue jusqu'à ce que CTRL+STOP ou STOP soit à nouveau pressé. Si BASROM (0FBB1h) = 1, il ne se passe rien.  
Modifie :      Rien.  
Note :           Cette routine est utilisée par l'interpréteur Basic.

### 000BDh (Main-ROM)    **CKCNTC** (« ChecK CoNTrol C »)

Rôle :            Vérifie si les touches CTRL+STOP ou STOP sont pressées ou non afin de stopper ou suspendre l'exécution en cours du programme Basic.  
Entrée :        Rien.  
Sortie :        Si les touches CTRL+STOP sont pressées et que BASROM (0FBB1h) = 0, l'exécution de l'interpréteur Basic est interrompu; si seulement STOP est pressé, l'exécution est suspendue jusqu'à ce que CTRL+STOP ou STOP soit à nouveau pressé. Si BASROM (0FBB1h) = 1, il ne se passe rien.  
Modifie :      Rien.  
Notes :        - Cette routine est utilisé par l'interpréteur Basic.  
                 - Par rapport à ISCNTC, cette routine met le pointeur temporaire 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 :            Initialise l'instruction PLAY et émet un bref bip sonore.  
Entrée :          Rien.  
Sortie :          Rien.  
Modifie :        Tout.  
Notes :          - Appel à la Sub-ROM sur MSX2 ou plus récent.  
                    - Ce son est produit par le PSG.

### 000C3h (Main-ROM)    **CLS** (« CLear Screen »)

Rôle :            Effacement de l'écran dans tous les modes.  
Entrée :          F = Indicateur ZF à 1.  
Sortie :          Rien.  
Modifie :        AF, BC.  
Note :            Appel à la Sub-ROM sur MSX2 ou plus récent.

### 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 :            Active ou désactive les touches de fonction.  
Entrée :          CNSDFG (0F3DEh) = 0 pour ignorer les touches de fonction, 1 pour les activer.  
Sortie :          Rien.  
Modifie :        Tout.  
Notes :          - Appel au Hook H.DSPF (0FDB3h).  
                    - Appeler la routine DSPFNK (000CFh en Main-ROM) pour afficher le texte des touches de fonction.



### 000CCh (Main-ROM)    **ERAFNK** (« ERase FuNction Key »)

Rôle :            Efface l'affichage des touches de fonction (en mode texte).  
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.

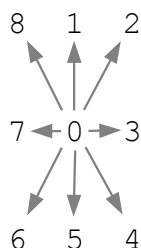
## 000D5h (Main-ROM) **GTSTCK** (« GeT joySTiCK status »)

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 : AF, B, DE, HL.

## 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, BC.

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

- Rôle :            Lecture du rayon optique, de la souris, de la tablette graphique ou du Track-Ball.
- Entrée :        A = Numéro de l'opération à effectuer (0~7 sur MSX1 ou 0~19 sur les autres). Pour chaque périphérique, la première opération (0, 4, 8, 12 ou 16) est indispensable car c'est elle qui lit les coordonnées fournies par le périphérique et les sauvegarde dans les variables systèmes XSAVE (0FAFEh) et YSAVE (0FB00h). Les opérations de lecture de coordonnées ne font que lire ces variables.
- Sortie :        A = Valeur résultante.
- Modifie :      Tout.
- Notes :        - La routine GTPAD fait appel à la routine NEWPAD de la Sub-ROM. (MSX2~)  
                  - Voici la liste des opérations possibles :

| Entrée           | Port              | Valeur résultante                                                                                                |
|------------------|-------------------|------------------------------------------------------------------------------------------------------------------|
| <b><u>0</u></b>  | Général<br>n°1    | A = 0FFh si le stylet est en contact avec la tablette graphique, les coordonnées sont donc lisible. Sinon A = 0. |
| 1                |                   | Abscisse de la tablette.                                                                                         |
| 2                |                   | Ordonnée de la tablette.                                                                                         |
| 3                |                   | 0FFh si le stylet appuie sur la tablette.                                                                        |
| <b><u>4</u></b>  | Général<br>n°2    | A = 0FFh si le stylet est en contact avec la tablette graphique, les coordonnées sont donc lisible. Sinon A = 0. |
| 5                |                   | Abscisse de la tablette.                                                                                         |
| 6                |                   | Ordonnée de la tablette.                                                                                         |
| 7                |                   | 0FFh si le stylet appuie sur la tablette.                                                                        |
|                  |                   |                                                                                                                  |
| <b><u>8</u></b>  | Crayon<br>optique | A = 0FFh si le crayon optique touche l'écran, les coordonnées sont donc lisible. Sinon A = 0.                    |
| 9                |                   | Abscisse du crayon optique.                                                                                      |
| 10               |                   | Ordonnée du crayon optique.                                                                                      |
| 11               |                   | 0FFh si le bouton du crayon optique est pressé.                                                                  |
| <b><u>12</u></b> | Général<br>n°1    | Toujours 0FFh. Lire les coordonnées juste après cet opération.                                                   |
| 13               |                   | Abscisse de la souris ou du Track-Ball.                                                                          |
| 14               |                   | Ordonnée de la souris ou du Track-Ball.                                                                          |
| 15               |                   | Toujours 0. (inutilisé)                                                                                          |
| <b><u>16</u></b> | Général<br>n°2    | Toujours 0FFh. Lire les coordonnées juste après cet opérations.                                                  |
| 17               |                   | Abscisse de la souris ou du Track-Ball.                                                                          |
| 18               |                   | Ordonnée de la souris ou du Track-Ball.                                                                          |
| 19               |                   | Toujours 0. (inutilisé)                                                                                          |

- Pour tester l'état des boutons de la souris ou du Track-Ball, veuillez utiliser la routine GTTRIG (000D8h en Main-ROM).
- Le crayon optique n'est pas géré par le MSX turbo R. Si on entre A = 8~11 en entrée, on aura A = 0 en sortie.
- La tablette graphique doit être une NEC PC-6051 ou compatible.

## 000DEh (Main-ROM)    **GTPDL** (« GeT PaDdLe »)

Rôle :            Lecture d'une molette (« Paddle Controler ») 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'entête ou d'un signal séparateur.
- Entrée :          Rien.
- Sortie :          F = L'indicateur CF passe à 1 si l'opération a été interrompue (par une erreur par exemple).
- Modifie :        Tout.
- Notes :          - Voir TAPoon (000EAh en Main-ROM) pour les explications du signal d'entête et du signal séparateur.  
- La vitesse (1200 ou 2400 bauds) est déterminée lors de la lecture du signal.  
- Les interruptions sont désactivées par la routine.  
- 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 = L'indicateur CF passe à 1 si l'opération a été interrompue (par une erreur par exemple) ou si le dernier octet du fichier a été lu.
- 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 et arrêt du moteur.
- Entrée :          Rien.
- Sortie :          Rien.
- Modifie :        Rien.
- Notes :          - Les interruptions sont réactivées en fin de routine.  
- Cette routine n'a pas d'effet sur MSX turbo R. (L'indicateur C = 1 en sortie)

## 000EAh (Main-ROM) **TAPOON** (« TAPe Out ON »)

- Rôle : Démarrage du moteur du magnétophone et écriture d'un signal d'entête ou d'un signal séparateur.
- Entrée : A = 0 pour écrire un signal un séparateur, ou autre valeur pour signal d'entête.
- Sortie : F = Indicateur CF à 1 si l'opération a été interrompue (par un CTRL+STOP ou une erreur).
- Modifie : Tout.
- Notes : - Le signal d'entête 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'entête (de 6.7 sec)    |      |                           |      |      |      |      |      |
| Entête 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)  |      |                           |      |      |      |      |      |
| .<br>.<br>.<br>.  | 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'entête suivi de dix octets 0D3h pour indiquer qu'il contient un programme Basic condensé et du nom du fichier. Après cette entête, il doit y avoir un séparateur qui annonce l'emplacement du programme. Le fichier se finit par une suite de sept 000h.

Format d'un fichier binaire :

|                   |                                               |      |                           |      |               |      |          |      |
|-------------------|-----------------------------------------------|------|---------------------------|------|---------------|------|----------|------|
| Début du fichier  | Signal d'entête (de 6.7 sec)                  |      |                           |      |               |      |          |      |
| Entête 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... |      |
| .<br>. .<br>.     | Données binaires (Programme machine ou autre) |      |                           |      |               |      |          |      |
| Fin du fichier    |                                               |      | ...fin                    |      |               |      |          |      |

Le fichier binaire commence par un signal d'entête suivi de dix octets 0D0h indiquant qu'il s'agit d'un programme machine et du nom du fichier. Après cette entête, 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'entête (de 6.7 sec)    |      |                           |      |        |      |      |      |
| Entête 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   |      |                           |      |        |      |      |      |
|                   | .                               |      |                           |      |        |      |      |      |
|                   | .                               |      |                           |      |        |      |      |      |
| Fin du fichier    | Signal séparateur (de 1.7 sec)  |      |                           |      |        |      |      |      |
|                   | Reste des caractères            |      |                           |      | ...fin |      |      |      |

Le fichier ASCII commence par un signal d'entête suivi de dix octets 0EAh pour indiquer qu'il contient de texte ASCII puis, du nom du fichier. Après cette entête, 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 CF 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) **TAPOOFF** (« TAPE Out OFF »)

Rôle : Fin d'écriture, arrêt du moteur.

Entrée : Rien.

Sortie : Rien.

Modifie : Rien.

Notes: - Rétabli les interruptions.

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

### 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. Elle modifie rien.

### 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 ZF à 1 si la queue est entièrement remplie.

Modifie :      AF, BC, HL.



## 000FCh (Main-ROM)    **RIGHTC** (« Move to the RIGHT from the Cursor »)

Rôle : Déplacement du curseur graphique d'un pixel vers la droite.  
Entrée : CLOC (0F92Ah) et CMASK (0F92Ch).  
Sortie : CLOC (0F92Ah) et CMASK (0F92Ch).  
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** (« move to the LEFT from the Cursor »)

Rôle : Déplacement du curseur graphique d'un pixel vers la gauche.  
Entrée : CLOC (0F92Ah) et CMASK (0F92Ch).  
Sortie : CLOC (0F92Ah) et CMASK (0F92Ch).  
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** (« move to the UP from the Cursor »)

Rôle : Déplacement du curseur graphique d'un pixel vers le haut.  
Entrée : CLOC (0F92Ah) et CMASK (0F92Ch).  
Sortie : CLOC (0F92Ah) et CMASK (0F92Ch).  
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 move to the UP from the Cursor »)

Rôle : Déplacement du curseur d'un pixel vers le haut avec test de dépassement d'écran.  
Entrée : CLOC (0F92Ah) et CMASK (0F92Ch).  
Sortie : F = L'indicateur CF passe à 1 si le curseur n'a pu être déplacé (sortie d'écran).  
CLOC (0F92Ah) et CMASK (0F92Ch).  
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** (« Move to the DOWN from the Cursor »)

Rôle : Déplacement du curseur graphique d'un pixel vers le bas.  
Entrée : CLOC (0F92Ah) et CMASK (0F92Ch).  
Sortie : CLOC (0F92Ah) et CMASK (0F92Ch).  
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 move to the DOWN from the Cursor »)

Rôle : Déplacement du curseur graphique d'un pixel vers le bas avec test de dépassement d'écran.  
Entrée : CLOC (0F92Ah) et CMASK (0F92Ch).  
Sortie : F = Indicateur CF à 1 si si le curseur n'a pu être déplacé (sortie d'écran).  
CLOC (0F92Ah) et CMASK (0F92Ch).  
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 ~ FFFFh).  
DE = Ordonnée (0 ~ FFFFh).  
Sortie : BC = Abscisse valide.  
DE = Ordonnée valide.  
F = Indicateur CF 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 Cursor »)

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

### 00114h (Main-ROM)     **FETCHC** (« FETCH current Cursor »)

- 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 :      AF, HL.

### 00117h (Main-ROM)     **STOREC** (« STORE current Cursor »)

- 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.

### 0011Ah (Main-ROM)    **SETATR** (« SET ATtribute byte »)

Rôle :            Modification de l'octet d'attribut ATRBYT (0F3F2h) au curseur graphique.  
Entrée :          A = Valeur du nouvel attribut (couleur).  
Sortie :          F = Indicateur CF à 1 si l'attribut n'est pas valide (>15).  
Modifie :        F.  
Note :            Cette routine n'est valable que dans les SCREEN 0 à 4.

### 0011Dh (Main-ROM)    **READC** (« READ current Cursor »)

Rôle :            Lecture de la couleur du pixel au curseur graphique actuel.  
Entrée :          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.  
Sortie :          A = Couleur du pixel (0~15 dans les SCREEN 5 à 7 ; 0 à 255 dans les SCREEN 8 à  
                         12)  
Modifie :        F.  
Note :            Appel à la Sub-ROM si le mode d'écran est supérieur au SCREEN 4.

### 00120h (Main-ROM)    **SETC** (« SET the Cursor »)

Rôle :            Placer un pixel à l'écran au curseur graphique actuel.  
Entrée :          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 :        AF  
Note :            Appel à la Sub-ROM si le mode d'écran est supérieur au SCREEN 4.

### 00123h (Main-ROM)     **NSETCX** (« Next SET the Cursor up to X pixel »)

Rôle : Tracer des pixels du curseur graphique actuel jusqu'à x pixels vers la droite.

Entrée : HL = Nombre de pixels à placer.  
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.  
ATRYT (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 l'instruction PAINT.

Entrée : Rien.

Sortie : Rien.

Modifie : AF.

### 0012Ch (Main-ROM)     **SCANR** (« SCAN to 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 = Valeur entrée moins le nombre de pixel parcourus jusqu'à la couleur rencontrée.

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 to 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 = Valeur entrée moins le nombre de pixel parcourus jusqu'à la couleur rencontrée.
- 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 : Activer 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 Slot 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<br>(C000h~FFFFh) |       | Slot de la plage 2<br>(8000h~BFFFh) |       | Slot de la plage 1<br>(4000h~7FFFh) |       | Slot de la plage 0<br>(0000h~3FFFh) |       |

### 0013Bh (Main-ROM)     **WSLREG** (« Write Slot REGister »)

Rôle :            Écriture dans le registre des Slot 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<br>(C000h~FFFFh) |       | Slot de la plage 2<br>(8000h~BFFFh) |       | Slot de la plage 1<br>(4000h~7FFFh) |       | Slot de la plage 0<br>(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 : Appeler la routine de lecture / écriture de secteur(s) sur un disque logique du la Disk-ROM.
- Entrée : A = Numéro du disque logique (0~7).  
B = Nombre de secteur.  
C = Identifiant du type de média utilisé. (0F8h~0FFh).  
0F8h pour une disquette 360K, 3,5", simple-face, 9 secteur ;  
0F9h pour une disquette 720K, 3,5", double-face, 9 secteur ;  
0FAh pour une disquette 320K, 3,5", simple-face, 8 secteur ;  
0FBh pour une disquette 640K, 3,5", double-face, 8 secteur ;  
0FCh pour une disquette 180K, 5,25", simple-face, 9 secteur ;  
0FDh pour une disquette 360K, 5,25", double-face, 9 secteur ;  
0FEh pour une disquette 160K, 5,25", simple-face, 8 secteur ;  
0FFh pour une disquette 320K, 5,25", double-face, 8 secteur.  
DE = Numéro du premier secteur.  
HL = Adresse de destination en mémoire vive.  
F = Indicateur CF à 1 pour écrire, 0 pour lire.
- Sortie : F = Indicateur CF à 1 si erreur rencontrée.  
A = Code d'erreur.  
B = Nombre de secteur non lus.
- Modifie : Tout.
- Notes : - Appel au Hook H.PHYD (0FFA7h).  
- Voici les codes d'erreur possible.  
0 = Disque protégé contre l'écriture  
2 = Disque pas prêt  
4 = Erreur de donnée  
6 = Erreur de recherche  
8 = Enregistrement pas trouvé  
10 = Erreur à l'écriture  
12 = Paramètre erroné  
14 = Pas assez de mémoire  
16 = Autre  
- La compatibilité avec les lecteurs 3,5" ou 5,25" dépend du contrôleur utilisé.  
- Aucun effet si il n'a y pas de Disk-ROM (sauf l'appel au Hook H.PHYD).

Exemple en assembleur :

```
; Lire le secteur 0 d'une disquette double face 3.5".
```

```
PHYDIO     equ     0114h
```

```
org       0D000h
```

```
RDSECT:
```

```
ld     a,0                    ; Lecteur "A:"  
ld     b,1                    ; Nombre de secteur à lire = 1  
ld     c,0F9h                ; Identifiant disquette double face 3.5"  
ld     de,0                   ; Premier secteur  
ld     hl,(0F351h)            ; Vers la cache de secteur du Disk-Basic  
or     a                      ; Lecture (mettre SCF pour écriture)  
call   PHYDIO                ; Lecture du secteur 0
```



#### 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 tabulations en espaces. Lors de l'utilisation d'une imprimante non-standard MSX, les caractères graphiques changent et les caractères japonais Hiragana sont convertis en Katakana sur un MSX japonais. De plus, CTRL+STOP provoque un « device I/O error ».

#### 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 cache du clavier.  
Entrée :          Rien.  
Sortie :          Rien.  
Modifie :        HL.

## 00159h (Main-ROM)     **CALBAS** (« CALl BASic interpreter »)

Rôle :            Appel à une routine de l'interpréteur Basic (appel inter-Slot) lorsque la Main-ROM n'est pas sélectionnée sur la plage 4000h~7FFFh.  
Entrée :          IX = Adresse à appeler.  
Sortie :          Rien.  
Modifie :        AF, IX, IY, registres auxiliaires.  
Note :            Utilisée par l'instruction CALL du Basic.

## **Routines ajoutées aux MSX2 :**

## 0015Ch (Main-ROM)     **SUBROM** (« call SUB-ROM »)

Rôle :            Appel à une adresse dans la Sub-ROM.  
Entrée :          Mettre IX dans la pile puis, IX = Adresse à appeler.  
Sortie :          Rien.  
Modifie :        Tout sauf 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 la Sub-ROM.  
Entrée :          IX = Adresse à appeler.  
Sortie :          Rien.  
Modifie :        Tout sauf IY et les registres auxiliaires du Z80.  
Notes :          - Routine absente sur MSX1.  
                  - Attention, cette routine ne fonctionne pas sous MSX-DOS ! Pour cela, vous pouvez utiliser la méthode suivante. Les conditions requises pour appeler une routine de la Sub-ROM sous MSX-DOS sont :  
                  1/ Charger l'adresse de la routine à appeler dans le registre IX.

2/ Assurez-vous que la pile ne trouve pas sur plage 0000h~3FFFh)

3/ Il n'est pas nécessaire de désactiver les interruptions.

4/ Le Hook H.NMI peut rester tel quel après un appel.

Comme il n'est pas possible de transmettre directement l'adresse avec IX et que la routine d'appel inter-Slot ne peut appeler de routine en Sub-ROM, utilisez la routine suivante. Celle-ci a été fournie par ASCII. Elle fonctionne sous toutes les versions de MSX-DOS.

```
; Entrée : IX = Adresse à appeler dans la Sub-ROM
;          AF, HL, DE, BC = Paramètres de la routine
;
; Sortie : AF, HL, DE, BC = Dépendent de la routine

CALSLT    equ    001ch
EXTROM     equ    015fh
NMI        equ    0066h          ; Interruption Non-masquable (NMI)
EXPTBL     equ    0fcc1h
H.NMI      equ    0fdd6h          ; Hook de la routine NMI

CALSUB:
    exx                    ; Préserve les paramètres
    ex    af,af'           ; des registers

    ld     hl,EXTROM        ; Place la routine suivante
    push   hl               ; dans la pile (longueur = 10)
    ld     hl,0c300h        ;
    push   hl               ; +0    inc sp
    push   ix               ; +1    inc sp
    ld     hl,021ddh        ; +2    ld ix,<adresse à appeler>
    push   hl               ; +6    nop
    ld     hl,03333h        ; +7    jp EXTROM
    push   hl
    ld     hl,0
    add    hl,sp
    ld     a,0c3h
    ld     (H.NMI),a
    ld     (H.NMI+1),hl
    ex     af,af'           ; Restore les paramètres
    exx                    ; des registers

    ld     ix,NMI
    ld     iy,(EXPTBL-1)
    call   CALSLT
    ei

    ld     hl,10 ; Jeter la routine qui sert d'interface
    add    hl,sp
    ld     sp,hl
    ret
```

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

Rôle :            Recherche du Slot de la Sub-ROM. (Routine du système appelée pendant l'initialisation du MSX.)

Entrée :          Rien.

Sortie :          EXBRSA (0FAF8h) = Numéro du slot de la Sub-ROM.

Modifie :        Tout.

Note :            Routine absente sur MSX1.

### 00165h (Main-ROM)     **CHKNEW** (« CHeCK NEW screen mode »)

Rôle :            Vérification si le mode d'écran est supérieur à SCREEN 4.  
Entrée :          Rien.  
Sortie :          F = Indicateur CF 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 de caractères d'un point donné jusqu'à la fin de la ligne (équivalent au CTRL+E du Basic).  
Entrée :          H = Numéro de la colonne à partir de laquelle il faut effacer.  
                    L = Numéro de la ligne à effacer.  
Sortie :          Rien.  
Modifie :        Tout.  
Note :            Routine absente sur MSX1.

### 0016Bh (Main-ROM)     **BIGFIL** (« BIG FILl »)

Rôle :            Remplissage de la mémoire vidéo.  
Entrée :          HL = Adresse de début (0000h~FFFFh).  
                    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 (0000h~FFFFh)  
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 (0000h~FFFFh).  
Sortie :          Rien.  
Modifie :        AF.  
Note :            Routine absente 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 (0000h~FFFFh).  
Sortie :        A = Contenu de la case mémoire lue.  
Modifie :      F.  
Note :        Routine absente sur MSX1.

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

Rôle :            Écriture dans une case mémoire de VRAM.  
Entrée :        HL = Adresse de la case mémoire (0000h~FFFFh).  
                  A = Donnée à écrire.  
Sortie :        Rien.  
Modifie :      AF.  
Note :        Routine absente sur MSX1.

### **Routines ajoutées aux MSX2+ :**

#### 0017Ah (Main-ROM)    **RDRES** (« ReaD RESet »)

Rôle :            Indique le type initialisation du MSX à faire.  
Entrée :        Rien.  
Sortie :        A = Le bit 7 à 1 indique si la routine STARUP (0000h) effectuera une initialisation logicielle sinon elle sera matérielle. Le bit 5 indique, sur MSX turbo R, si le R800 doit être initialisé ou pas.  
Modifie :      AF.  
Note :        Routine existante qu'à partir du MSX2+ (autrement cette adresse contient 00h). Elle n'est utile que pour le système.

#### 0017Dh (Main-ROM)    **WRRES** (« WRite RESet »)

Rôle :            Choix du type initialisation du MSX à faire.  
Entrée :        A = permet de choisir la façon d'initialiser le MSX. Le bit 7 doit être à 1 pour effectuer une initialisation logicielle. Sur MSX turbo R, mettre le bit 5 à 1 pour ne pas initialiser le R800.  
Sortie :        Rien.  
Modifie :      AF.  
Note :        Routine existante qu'à partir du MSX2+ (autrement cette adresse contient 00h). Elle n'est utile que pour le système.

## 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.

Notes :

- 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 poids 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 CF à 1 si la lecture s'est déroulée normalement.

E = 3 bits de poids 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.

## 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 poids 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 CF à 1 si la lecture s'est déroulée normalement.

E = 3 bits de poids 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.

- Un enregistrement à 15,75kHz en mode R800 ROM provoque une erreur (indicateur C à 0).

- Presser la touche STOP pour interrompre un enregistrement.



## 4.4 *Routines de l'interpréteur Basic*

En plus des routines du Bios, voici des routines internes de l'interpréteur Basic appelées. Celles-ci se trouvent en Main-ROM et ont été garanties de rester aux mêmes adresses par ASCII.

La routine CALBAS (0159h) du Bios de la Main-ROM permet d'appeler ces routines lorsque la ROM principale n'est pas sélectionnée sur la plage 4000h~7FFFh.

### **Routines « Math-pack »**

Les routines internes du Basic appelées « Math-pack » servent à gérer les variables et les valeurs spécifiées d'un programme en Basic. Elles permettent aussi d'effectuer des opérations mathématiques, de comparaison et de conversion de nombres codés sur 2 octets pour un entier, 3 pour une chaîne de caractères, 4 pour une valeur en précision simple ou 8 pour une valeur en double précision dans vos programmes en langage machine. Ces nombres sont codés comme ci-dessous.

#### **Nombre entier :**

3<sup>e</sup> octet : 8 bits les moins significatifs du nombre

4<sup>e</sup> octet : Signe (bit 7) et les 7 bits les plus significatifs du nombre

Les nombres codés en simple et double précision sont stockés au format BCD (norme [IEEE 754-1985](#)).

#### **Nombre codés en simple/double précision :**

1<sup>er</sup> octet : Le signe (bit 7) et exposant

2<sup>e</sup> octet : Le chiffre avant et après la virgule

3<sup>e</sup> octet : 2<sup>e</sup> et 3<sup>e</sup> chiffre après la virgule

4<sup>e</sup> octet : 4<sup>e</sup> et 5<sup>e</sup> chiffre après la virgule ← Jusqu'ici en simple précision

5<sup>e</sup> octet : 6<sup>e</sup> et 7<sup>e</sup> chiffre après la virgule

6<sup>e</sup> octet : 8<sup>e</sup> et 9<sup>e</sup> chiffre après la virgule

7<sup>e</sup> octet : 10<sup>e</sup> et 11<sup>e</sup> chiffre après la virgule

8<sup>e</sup> octet : 12<sup>e</sup> et 13<sup>e</sup> chiffre après la virgule

#### **Pour une chaîne de caractères :**

3<sup>e</sup> octet : 8 bits les moins significatifs de l'adresse de chaîne

4<sup>e</sup> octet : 8 bits les plus significatifs de l'adresse de chaîne

La variable VALTYP (0F663h) spécifie le type de codage des nombres.

2 = Nombre entier

3 = Nombre dans une chaîne de caractères

4 = Nombre en simple précision

8 = Nombre en double précision

## Routines d'opérations de base :

### 0268Ch (Main-ROM)    **DECSUB** (« DECimal SUBstraction »)

Rôle : Effectuer l'opération  $DAC \leftarrow DAC - ARG$   
Entrée : DAC (0F7F6h) = Opérande 1.  
ARG (0F847h) = Opérande 2. (Nombre à soustraire.)  
VALTYP (0F663h) à 8  
Sortie : DAC (0F7F6h) = Résultat  
Modifie : AF, BC, DE, HL.  
Note : DAC et ARG doivent être en double précision.

### 0269Ah (Main-ROM)    **DECADD** (« DECimal ADDition »)

Rôle : Effectuer l'opération  $DAC \leftarrow DAC + ARG$   
Entrée : DAC (0F7F6h) = Opérande 1.  
ARG (0F847h) = Opérande 2. (Nombre à additionner.)  
VALTYP (0F663h) à 8  
Sortie : DAC (0F7F6h) = Résultat  
Modifie : AF, BC, DE, HL.  
Note : DAC et ARG doivent être en double précision.

### 026FAh (Main-ROM)    **DECNRM** (« DECimal NoRMalisation »)

Rôle : Normaliser le format du nombre à DAC.  
Entrée : DAC (0F7F6h) = Nombre en double précision.  
VALTYP (0F663h) à 8  
Sortie : DAC (0F7F6h) = Résultat  
Modifie : AF, BC, DE, HL.  
Note : Cette routine retire les zéros inutiles après la virgule. (ex : 0.00123 -> 0.123E-2)

### 0273Ch (Main-ROM)    **DECROU** (« DECimal ROUnd »)

Rôle : Arrondir le nombre à DAC  
Entrée : DAC (0F7F6h) = Nombre en double précision.  
VALTYP (0F663h) à 8  
Sortie : DAC (0F7F6h) = Résultat  
Modifie : AF, BC, DE, HL.

### 027E6H (Main-ROM)    **DECMUL** (« DECimal MULTiplication »)

Rôle : Effectuer l'opération  $DAC \leftarrow DAC * ARG$   
Entrée : DAC (0F7F6h) = Opérande 1.  
ARG (0F847h) = Opérande 2. (Nombre à multiplier.)  
VALTYP (0F663h) à 8  
Sortie : DAC (0F7F6h) = Résultat  
Modifie : AF, BC, DE, HL.  
Note : DAC et ARG doivent être en double précision.

### 0289FH (Main-ROM)    **DECDIV** (« DECimal DIVision »)

Rôle :            Effectuer l'opération  $DAC \leftarrow DAC / ARG$   
Entrée :        DAC (0F7F6h) = Opérande 1.  
                  ARG (0F847h) = Opérande 2. (Nombre à diviser.)  
                  VALTYP (0F663h) à 8  
Sortie :        DAC (0F7F6h) = Résultat  
Modifie :      AF, BC, DE, HL.  
Note :        DAC et ARG doivent être en double précision.

### 0314Ah (Main-ROM)    **UMULT** (« Unsigned MULTiplication »)

Rôle :            Effectuer l'opération  $DE \leftarrow BC * DE$   
Entrée :        BC = Opérande 1. (Nombre à multiplier.)  
                  DE = Opérande 2.  
Sortie :        DE = Résultat  
                  DAC+2 (0F7F8h) = Résultat  
                  VALTYP (0F663h) = 2  
Modifie :      AF, BC, DE.

### 03167h (Main-ROM)    **ISUB** (« Integer SUBstraction »)

Rôle :            Effectuer l'opération  $HL \leftarrow DE - HL$   
Entrée :        DE = Opérande 1.  
                  HL = Opérande 2. (Nombre à soustraire.)  
Sortie :        HL = Résultat.  
                  DAC+2 (0F7F8h) = Résultat  
                  VALTYP (0F663h) = 2  
Modifie :      AF, BC, DE, HL.

### 03172h (Main-ROM)    **IADD** (« Integer ADDition »)

Rôle :            Effectuer l'opération  $HL \leftarrow DE + HL$   
Entrée :        DE = Opérande 1.  
                  HL = Opérande 2. (Nombre à additionner.)  
Sortie :        HL = Résultat.  
                  DAC+2 (0F7F8h) = Résultat  
                  VALTYP (0F663h) = 2  
Modifie :      AF, BC, DE, HL.

### 03193h (Main-ROM)     **IMULT** (« Integer MULTiplication »)

Rôle :            Effectuer l'opération  $HL \leftarrow DE * HL$   
Entrée :          DE = Opérande 1. (Nombre à multiplier.)  
                    HL = Opérande 2.  
Sortie :          HL = Résultat.  
                    DAC+2 (0F7F8h) = Résultat  
                    VALTYP (0F663h) = 2  
Modifie :        AF, BC, DE, HL.

### 031E6h (Main-ROM)     **IDIV** (« Integer DIVision »)

Rôle :            Effectuer l'opération  $HL \leftarrow DE / HL$   
Entrée :          DE = Opérande 1. (Nombre à diviser.)  
                    HL = Opérande 2.  
Sortie :          HL = Résultat.  
                    DAC+2 (0F7F8h) = Résultat  
                    VALTYP (0F663h) = 2  
Modifie :        AF, BC, DE, HL.

### 0323Ah (Main-ROM)     **IMOD** (« Integer MOD »)

Rôle :            Effectuer l'opération  $HL \leftarrow DE \bmod HL$   
Entrée :          DE = Opérande 1. (Nombre à diviser.)  
                    HL = Opérande 2.  
Sortie :          HL = Résultat.  
                    DAC+2 (0F7F8h) = Reste de la division  
                    VALTYP (0F663h) = 2  
Modifie :        AF, BC, DE, HL.

## **Puissance :**

### 037C8h (Main-ROM)     **SNGEXP** (« SiNGle presision EXPonentiation »)

Rôle :            Effectuer l'opération  $DAC \leftarrow DAC ^ ARG$  en simple précision  
Entrée :          DAC (0F7F6h) = Opérande 1.  
                    ARG (0F847h) = Opérande 2. (Puissance.)  
                    VALTYP (0F663h) à 4  
Sortie :          DAC (0F7F6h) = Résultat  
Modifie :        AF, BC, DE, HL.

### 037D7h (Main-ROM)    **DBLEXP** (« DouBLLe presision EXPonentiation »)

Rôle :            Effectuer l'opération  $DAC \leftarrow DAC \wedge ARG$  en double précision  
Entrée :        DAC (0F7F6h) = Opérande 1.  
                  ARG (0F847h) = Opérande 2. (Puissance.)  
                  VALTYP (0F663h) à 8  
Sortie :        DAC (0F7F6h) = Résultat  
Modifie :      AF, BC, DE, HL.

### 0383Fh (Main-ROM)    **INTEXP** (« INTeger EXPonentiation »)

Rôle :            Effectuer l'opération  $DAC \leftarrow DE \wedge HL$  en entier  
Entrée :        DAC (0F7F6h) = Opérande 1.  
                  ARG (0F847h) = Opérande 2. (Puissance.)  
                  VALTYP (0F663h) à 2  
Sortie :        DAC (0F7F6h) = Résultat  
Modifie :      AF, BC, DE, HL.

## **Fonctions trigonométriques :**

### 02993h (Main-ROM)    **COS** (« COSine »)

Rôle :            Effectuer la fonction  $DAC \leftarrow \text{COS}(DAC)$   
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Modifie :      AF, BC, DE, HL.

### 029ACh (Main-ROM)    **SIN** (« SINE »)

Rôle :            Effectuer la fonction  $DAC \leftarrow \text{SIN}(DAC)$   
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Modifie :      AF, BC, DE, HL.

### 029FBh (Main-ROM)    **TAN** (« TANGent »)

Rôle :            Effectuer la fonction  $DAC \leftarrow \text{TAN}(DAC)$   
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Modifie :      AF, BC, DE, HL.

### 02A14h (Main-ROM)    **ATN** (« ArcTaNgent »)

Rôle :            Effectuer la fonction  $DAC \leftarrow \text{ATN}(DAC)$   
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Modifie :      AF, BC, DE, HL.

## Autres fonctions :

### 02A72h (Main-ROM)    **LOG** (« LOGarithmic »)

Rôle :            Effectuer la fonction  $DAC \leftarrow \text{LOG}(DAC)$   
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Modifie :       AF, BC, DE, HL.

### 02AFFh (Main-ROM)    **SQR** (« SQuare Root »)

Rôle :            Effectuer la fonction  $DAC \leftarrow \text{SQR}(DAC)$   
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Modifie :       AF, BC, DE, HL.

### 02A72h (Main-ROM)    **EXP** (« EXPonential »)

Rôle :            Effectuer la fonction  $DAC \leftarrow \text{EXP}(DAC)$   
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Modifie :       AF, BC, DE, HL.

### 02AFFh (Main-ROM)    **RND** (« RaNDom »)

Rôle :            Effectuer la fonction  $DAC \leftarrow \text{RND}(DAC)$   
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Modifie :       AF, BC, DE, HL.

### 02E71h (Main-ROM)    **SIGN** (« SIGNe »)

Rôle :            Indique le signe du nombre à DAC  
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Sortie :        Si négatif A = FFh, l'indicateur ZF = 0 et CF = 1 ;  
                  Si égal à zéro : A = 0, l'indicateur ZF = 1 et ZF = 0 ;  
                  Si positif : A = 1, l'indicateur ZF = 0 et CF = 0.  
Modifie :       A

### 02E82h (Main-ROM)    **ABSFN** (« ABSolute FoNction »)

Rôle :            Effectuer la fonction  $DAC \leftarrow \text{ABS}(DAC)$   
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Sortie :        DAC (0F7F6h) = Résultat  
Modifie :       AF, BC, DE, HL.

### 02E8Dh (Main-ROM)    **NEG** (« NEGative base »)

Rôle :            Effectuer la fonction  $DAC \leftarrow \text{NEG}(DAC)$   
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Sortie :        DAC (0F7F6h) = Résultat  
Modifie :       A, HL.

### 02E97h (Main-ROM)    **SGN** (« SiGne Number »)

Rôle :            Effectuer la fonction  $DAC \leftarrow SNG(DAC)$   
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Sortie :        DAC (0F7F6h) = Résultat en simple précision  
Modifie :      A, HL.  
Note :        Le résultat est retourné sous la forme d'un entier de 2 octets.

### Transfert de nombre :

### 02C4Dh (Main-ROM)    **MAF** (« Move to Argument a Floating-point number »)

Rôle :             $ARG \leftarrow DAC$ . Placer le nombre de DAC à ARG.  
Entrée :        VALTYP (0F663h) à 8  
Modifie :      AF, BC, DE, HL.

### 02C50h (Main-ROM)    **MAM**

Rôle :             $ARG \leftarrow (HL)$ . Placer le nombre pointé par HL à ARG.  
Entrée :        VALTYP (0F663h) à 8  
Modifie :      AF, BC, DE, HL.

### 02C53h (Main-ROM)    **MOV8DH** (« MOVE 8 type number to (De) from (Hl) »)

Rôle :             $(DE) \leftarrow (HL)$ . Placer le nombre pointé par HL à l'adresse pointée par DE.  
Entrée :        VALTYP (0F663h) à 8  
Modifie :      AF, BC, DE, HL.

### 02C59h (Main-ROM)    **MFA** (« Move Floating-point Argument »)

Rôle :             $DAC \leftarrow ARG$ . Placer le nombre de ARG à DAC.  
Entrée :        VALTYP (0F663h) à 8  
Modifie :      AF, BC, DE, HL.

### 02C5Ch (Main-ROM)    **MFm** (« Move Floating-point nuMber »)

Rôle :             $DAC \leftarrow (HL)$ . Placer le nombre pointé par HL à DAC.  
Entrée :        VALTYP (0F663h) à 8  
Modifie :      AF, BC, DE, HL.

### 02C67h (Main-ROM)    **MMF**

Rôle :             $(HL) \leftarrow DAC$ . Placer le nombre de DAC à l'adresse pointée par HL.  
Entrée :        VALTYP (0F663h) à 8  
Modifie :      AF, BC, DE, HL.

**02C6Ah (Main-ROM)    MOV8HD** (« MOVe 8 type number to (Hl) from (De)  
Rôle :            (HL)  $\leftarrow$  (DE). Placer le nombre pointé par DE à l'adresse pointée par HL.  
Entrée :        VALTYP (0F663h) à 8  
Modifie :       AF, BC, DE, HL.

**02C6Fh (Main-ROM)    XTF** (« eXchange sTack with Floating-point number »)  
Rôle :            (SP)  $\leftrightarrow$  DAC. Inverser le nombre placé sur la pile avec celui de DAC.  
Entrée :        VALTYP (0F663h) à 8  
Modifie :       AF, BC, DE, HL

**02CC7h (Main-ROM)    PHA** (« PusH the Argument »)  
Rôle :            ARG  $\rightarrow$  (SP). Placer l'argument sur la pile.  
Entrée :        VALTYP (0F663h) à 8  
Modifie :       AF, BC, DE, HL.

**02CCCh (Main-ROM)    PHF** (« PusH the Floating-point number »)  
Rôle :            DAC  $\rightarrow$  (SP). Placer le nombre de DAC sur la pile.  
Entrée :        VALTYP (0F663h) à 8  
Modifie :       AF, BC, DE, HL.

**02CDCh (Main-ROM)    PPA** (« PoP the Argument »)  
Rôle :            (SP)  $\rightarrow$  ARG. Placer le nombre qu'il y a sur la pile à l'argument.  
Entrée :        VALTYP (0F663h) à 8  
Modifie :       AF, BC, DE, HL.

**02CE1h (Main-ROM)    PPF** (« PoP the Floating-point number »)  
Rôle :            (SP)  $\rightarrow$  DAC. Placer le nombre qu'il y a sur la pile à l'accumulateur décimal.  
Entrée :        VALTYP (0F663h) à 8  
Modifie :       AF, BC, DE, HL.

**02EB1h (Main-ROM)    PUSHF** (« PUSH the Floating-point number »)  
Rôle :            DAC  $\rightarrow$  (SP). Placer le nombre de DAC sur la pile.  
Entrée :        VALTYP (0F663h) à 4  
Modifie :       DE.

**02EBEh (Main-ROM)    MOVFM**  
Rôle :            DAC  $\leftarrow$  (HL). Placer le nombre pointé par HL à DAC.  
Entrée :        VALTYP (0F663h) à 4  
Modifie :       BC, DE, HL.



#### 02EC1h (Main-ROM)    **MOVFR**

Rôle :            DAC → (CBED). Placer le nombre contenu dans les registres C, B, E et D à DAC.  
Entrée :        VALTYP (0F663h) à 4  
Modifie :       DE.

#### 02ECCh (Main-ROM)    **MOVRF**

Rôle :            (CBED) ← DAC. Placer le nombre de DAC dans les registres C, B, E et D.  
Entrée :        VALTYP (0F663h) à 4  
Modifie :       BC, DE, HL.

#### 02ED6h (Main-ROM)    **MOVRFM**

Rôle :            (CBED) ← (HL). Placer le nombre pointé par HL dans les registres C, B, E et D.  
Entrée :        VALTYP (0F663h) à 4  
Modifie :       BC, DE, HL.

#### 02EDFh (Main-ROM)    **MOVRFM**

Rôle :            (BCDE) ← (HL) Placer le nombre pointé par HL dans les registres B, C, D et E.  
Entrée :        VALTYP (0F663h) à 4  
Modifie :       BC, DE, HL.

#### 02EE8h (Main-ROM)    **MOVFM**

Rôle :            (HL) ← DAC. Placer le nombre de DAC à l'adresse pointée par HL.  
Entrée :        VALTYP (0F663h) à 4  
Modifie :       BC, DE, HL.

#### 02EEBh (Main-ROM)    **MOVE**

Rôle :            (HL) ← (DE). Placer le nombre pointée par DE à l'adresse pointée par HL.  
Entrée :        VALTYP (0F663h) à 4  
Modifie :       BC, DE, HL.

#### 02EEFh (Main-ROM)    **VMOVAM**

Rôle :            ARG ← (HL). Placer le nombre de ARG à l'adresse pointée par HL.  
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Modifie :       BC, DE, HL.

#### 02EF2h (Main-ROM)    **MOVVFM**

Rôle :            (DE) ← (HL). Placer le nombre pointée par HL à l'adresse pointée par DE.  
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Modifie :       BC, DE, HL.

### 02EF3h (Main-ROM)    **VMOVE**

Rôle :            (HL)  $\leftarrow$  (DE). Placer le nombre pointée par DE à l'adresse pointée par HL.  
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Modifie :       BC, DE, HL.

### 02F05h (Main-ROM)    **VMOVFA**

Rôle :            DAC  $\leftarrow$  ARG. Placer le nombre de ARG à DAC.  
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Modifie :       BC, DE, HL.

### 02F08h (Main-ROM)    **VMOVFM**

Rôle :            DAC  $\leftarrow$  (HL). Placer le nombre de DAC à l'adresse pointée par HL.  
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Modifie :       BC, DE, HL.

### 02F0Dh (Main-ROM)    **VMOVAF**

Rôle :            ARG  $\leftarrow$  DAC. Placer le nombre de DAC à ARG.  
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Modifie :       BC, DE, HL.

### 02F10h (Main-ROM)    **VMOVMF**

Rôle :            (HL)  $\leftarrow$  DAC. Placer le nombre de DAC à l'adresse pointée par HL.  
Entrée :        VALTYP (0F663h) à 2, 4 ou 8  
Modifie :       BC, DE, HL.

## Comparaison de nombre :

### 02F21h (Main-ROM)    **FCOMP**

Rôle : Comparer le nombre placé dans les registres C, B, E et D avec celui de DAC.  
Entrée : C, B, E et D = Premier octet, second octet, troisième octet et quatrième octet du nombre à comparer avec celui à DAC.  
DAC (0F7F6h) = Nombre à comparer avec celui dans C, B, E et D.  
VALTYP (0F663h) à 4.  
Sortie : Si CBED > DAC alors A=FFh, l'indicateur ZF = 0 et CF = 1 ;  
Si CBED = DAC alors A=0, l'indicateur ZF = 1 et CF = 0 ;  
Si CBED < DAC alors A=1, l'indicateur ZF = 0 et CF = 0.  
Modifie : AF, HL.

### 02F4Dh (Main-ROM)    **ICOMP**

Rôle : Comparer le nombre entier placé dans DE avec celui dans HL.  
Entrée : VALTYP (0F663h) à 2. (entier)  
Sortie : Si DE > HL alors A=FFh, l'indicateur ZF = 0 et CF = 1 ;  
Si DE = HL alors A=0, l'indicateur ZF = 1 et CF = 0 ;  
Si DE < HL alors A=1, l'indicateur ZF = 0 et CF = 0.  
Modifie : AF, HL.

### 02F5Ch (Main-ROM)    **XDCOMP**

Rôle : Comparer le nombre à ARG avec celui à DAC.  
Entrée : VALTYP (0F663h) à 8. (double précision)  
Sortie : Si ARG > DAC alors A=FFh, l'indicateur ZF = 0 et CF = 1 ;  
Si ARG = DAC alors A=0, l'indicateur ZF = 1 et CF = 0 ;  
Si ARG < DAC alors A=1, l'indicateur ZF = 0 et CF = 0.  
Modifie : AF, BC, DE, HL.

## **Conversion de nombre :**

### **02F8Ah (Main-ROM)    FRCINT**

- Rôle : Convertir le nombre à DAC en entier sur 2 octets à DAC+2.
- Entrée : DAC (0F7F6h) = Nombre entier, en simple ou double précision.  
VALTYP (0F663h) = 4 ou 8.
- Sortie : DAC+2 (0F7F8h) = Nombre entier.  
VALTYP (0F663h) = 2.
- Modifie : AF, BC, DE, HL.
- Note : Une erreur « Type mismatch » est générée si il y a une chaîne caractère à DAC.  
Une erreur « Overflow » est générée si le nombre est trop grand.

### **02FB2h (Main-ROM)    FRCSNG**

- Rôle : Convertir le nombre à DAC en un nombre de simple précision.
- Entrée : DAC (0F7F6h) = Nombre entier, en simple ou double précision.  
VALTYP (0F663h) = 2 ou 8.
- Sortie : DAC (0F7F6h) = Nombre en simple précision.  
VALTYP (0F663h) = 4.
- Modifie : AF, BC, DE, HL.
- Note : Une erreur « Type mismatch » est générée si il y a une chaîne caractère à DAC. Si le nombre double précision dépasse il est arrondi.

### **0303Ah (Main-ROM)    FRCDBL**

- Rôle : Convertit le nombre à DAC en un nombre de double précision.
- Entrée : DAC (0F7F6h) = Nombre entier ou en simple précision.  
VALTYP (0F663h) = 2 ou 4. (Sans effet si 8.)
- Sortie : DAC (0F7F6h) = Résultat.  
VALTYP (0F663h) = 8.
- Modifie : AF, BC, DE, HL.
- Note : Une erreur « Type mismatch » est générée si il y a une chaîne caractère à DAC.

### **030BEh (Main-ROM)    FIXER**

- Rôle : Effectuer :  $DAC \leftarrow SGN(DAC) * INT(ABS(DAC))$ .
- Entrée : DAC (0F7F6h) = Nombre entier, en simple ou double précision.  
VALTYP (0F663h) = 2, 4 ou 8.
- Sortie : DAC (0F7F6h) = Résultat.  
VALTYP (0F663h) = 2, 4 ou 8.
- Modifie : AF, BC, DE, HL.
- Note : Une erreur « Type mismatch » est générée si il y a une chaîne caractère à DAC.

### 030CFh (Main-ROM)    **INT** (« convert to INTegeter »)

- Rôle : Convertir le nombre à DAC en un nombre entier.
- Entrée : DAC (0F7F6h) = Nombre en simple ou double précision  
VALTYP (0F663h) à 4 ou 8.
- Sortie : DAC (0F7F6h) = Nombre entier  
VALTYP (0F663h) = 2.
- Modifie : AF, BC, DE, HL.
- Note : Une erreur « Type mismatch » est générée si il y a une chaîne caractère à DAC.  
Une erreur « Overflow » est générée si le nombre est trop grand.

### Entrée/sortie de nombre à virgule flottante :

#### 03299h (Main-ROM)    **FIN** (« Floating-point number IN »)

- Rôle : Convertir un nombre à virgule flottante dans une chaîne de caractère ASCII vers un nombre à DAC.
- Entrée : HL = Adresse de la chaîne.  
A = Premier caractère de la chaîne.
- Sortie : DAC (0F7F6h) = 2, 4 ou 8.  
C = 0 si le nombre est à virgule, sinon 255.  
B = Nombre de chiffres après le virgule.  
D = Nombre de chiffres total.
- Modifie : AF, BC, DE, HL.

#### 03425h (Main-ROM)    **FOUT** (« Floating-point number OUT »)

- Rôle : Convertir un nombre à virgule flottante placé à DAC en un nombre à virgule flottante dans une chaîne de caractère ASCII (au format non spécifié).
- Entrée : DAC (0F7F6h) = Nombre.  
VALTYP (0F663h) = Type de format du nombre.  
B = Nombre de chiffres avant la virgule.  
C = Nombre de chiffres après la virgule.
- Sortie : HL = Adresse de la chaîne résultante.  
DE = Adresse de fin de la chaîne.
- Modifie : AF, BC, DE, HL.

### 03426h (Main-ROM)     **PUFOUT**

Rôle : Convertir un nombre stocké à DAC en une chaîne au format spécifié.  
Entrée : DAC (0F7F6h) = Nombre  
VALTYP (0F663h) = Type de format du nombre.  
A = Format désiré  
    Bit 7 à 0 pour aucun format spécifié, 1 pour spécifier le format ;  
    Bit 6 à 1 pour séparer chaque 3 chiffres avec une virgule ;  
    Bit 5 à 1 pour remplacer l'espace en tête par \* ;  
    Bit 4 à 1 pour mettre \$ en tête ;  
    Bit 3 à 1 pour mettre le signe + si la valeur est positive ;  
    Bit 2 à 1 pour placer le signe derrière ;  
    Bit 1 inutilisé ;  
    Bit 0 à 0 pour virgule fixe, 1 virgule flottante.  
B = Nombre de chiffres avant la virgule  
C = Nombre de chiffres après la virgule  
Sortie : HL = Adresse de la chaîne résultante.  
        DE = Adresse de fin de la chaîne.  
Modifie : AF, BC, DE, HL.

### 0371Ah (Main-ROM)     **FOUTB**

Rôle : Convertir un entier sur 2 octets à DAC+2 en une chaîne d'expression binaire.  
Entrée : DAC+2 (0F7F8h) = Nombre entier  
VALTYP (0F663h) à 2.  
Sortie : HL = Adresse de la chaîne résultante.  
Modifie : AF, BC, DE, HL.

### 0371Eh (Main-ROM)     **FOUTO**

Rôle : Convertir un entier sur 2 octets à DAC+2 en une chaîne d'expression octale.  
Entrée : DAC+2 (0F7F8h) = Nombre entier  
VALTYP (0F663h) à 2.  
Sortie : HL = Adresse de la chaîne résultante.  
Modifie : AF, BC, DE, HL.

### 03722h (Main-ROM)     **FOUTH**

Rôle : Convertir un entier sur 2 octets à DAC+2 en une chaîne d'expression hexadécimale.  
Entrée : DAC+2 (0F7F8h) = Nombre entier  
VALTYP (0F663h) à 2.  
Sortie : HL = Adresse de la chaîne résultante.  
Modifie : AF, BC, DE, HL.

## Routines générales de l'interpréteur

En plus des routines du Bios et celles appelées « Math-pack », voici des routines internes du Basic garanties de rester aux mêmes adresses par ASCII. Ces routines et celles de « Math-pack » vous permettra d'ajouter des instructions au Basic avec des paramètres avec CALL par exemple. Elles pourront aussi vous être utiles dans vos programmes en langage machine même pour MSX-DOS.

### 0406Fh (Main-ROM)    **ERRHAND**

Rôle :            Retour au mode direct du Basic en affichant le message d'erreur correspondant.  
Entrée :        E = Code d'erreur. (Voir [les codes d'erreur du Basic](#) page 600.)  
Sortie :        ERRFLG (0F414h) = Code d'erreur.  
Modifie :      AF, BC, DE, HL  
Note :        Routine sans retour.

### 0409Bh (Main-ROM)    **READYR**

Rôle :            Retour au mode direct du Basic.  
Entrée :        Rien.  
Modifie :      AF, BC, DE, HL.  
Note :        Routine sans retour.

### 042B2h (Main-ROM)    **CRUNCH**

Rôle :            Tokénise les instructions d'une ligne de programme Basic au format texte.  
Entrée :        HL = Pointer vers le texte ASCII avec 00h à la fin.  
Sortie :        La ligne tokénisée est placée à KBUF (0F41Fh).  
                 HL = KBUF  
                 BC = Longueur de la ligne au format texte, le 00h de fin inclu.  
                 DE = Adresse de fin de la ligne au format tokénisée, les trois de fin 00h inclus.  
Modifie :      AF, BC, DE, HL.

### 04295h (Main-ROM)    **LINEADRS**

Rôle :            Retourne l'adresse d'une ligne de programme Basic.  
Entrée :        DE = Numéro de ligne  
Sortie :        BC = Adresse de la ligne correspondante (à partir de l'adresse de la ligne suivante).  
                 F = ZF est mis à 0 si le numéro de ligne spécifié n'existe pas. Et dans ce cas, l'adresse de la ligne qui précède ce numéro est placées dans BC et celle de la suivante dans HL.  
Modifie :      AF, BC, DE, HL.

#### 04601h (Main-ROM)     **NEWSTT**

Rôle :            Exécuter une ligne de programme Basic.  
Entrée :          HL = Adresse de la ligne de programme à exécuter. La ligne doit commencer par le caractère deux-points. Cette ligne peut être placée à n'importe quelle adresse de la mémoire utilisateur.  
Modifie :        AF, BC, DE, HL.

#### 04666h (Main-ROM)     **CHRGTR**

Rôle :            Saute les espaces jusqu'au prochain caractère.  
Entrée :          HL = Pointeur actuel.  
Sortie :          A = Caractère trouvé.  
                    HL = adresse du caractère.  
                    F = ZF est mis à 1 s'il n'y a plus de caractère dans la phrase.  
Modifie :        AF, BC, DE, HL.

#### 04C64h (Main-ROM)     **FRMEVL**

Rôle :            Évaluer une expression dans le programme et la retourner.  
Entrée :          HL = Adresse du début de l'expression.  
Sortie :          HL = Adresse de l'expression suivante.  
                    DAC (F7F6h) = Résultat.  
                    VALTYP (F663h) = 2, 3, 4 ou 8 selon le type d'expression.  
Modifie :        AF, BC, DE, HL.  
Note :            L'opération est effectuée quelque soit le type d'expression utilisé. Le type du résultat dépendra du résultat. Les routines 521Ch et 542Fh sont plus pratiques si vous voulez un type de résultat attendu.

#### 04DC7h (Main-ROM)     **EVAL**

Rôle :            Évaluer une expression dans le programme non tokenisé et la retourner.  
Entrée :          HL = Adresse du début de l'expression.  
Sortie :          HL = Adresse de l'expression suivante.  
                    DAC (F7F6h) = Nombre trouvé.  
                    VALTYP (F663h) = 2, 4 ou 8 si nombre trouvé.  
Modifie :        AF, BC, DE, HL.



## 04EB8h (Main-ROM)

Rôle : Convertir un nombre au format alphanumérique préfixé par le caractère esperluette ("&") au format entier, simple/double précision.

Entrée : DAC (F7F6h) = Nombre au format alphanumérique préfixé par "&B", "&H" ou "&O".  
VALTYP (0F663h) = 3

Sortie : DAC (0F7F6h) = Résultat.  
VALTYP (0F663h) = 2, 4 ou 8 selon le type de format du résultat.

Modifie : AF, BC, DE, HL.

## 0521Ch (Main-ROM) **GETBYT**

Rôle : Évaluer une expression et la retourner en type entier de 1 octet.

Entrée : HL = Adresse du début de l'expression.

Sortie : HL = Adresse de la prochaine expression.  
A = E = Résultat.

Modifie : AF, BC, DE, HL.

Note : Cette routine est pratique car elle effectue également une vérification du type, une vérification de débordement et stocke les valeurs dans des registres. En cas de débordement, l'erreur « Illegal function call » se produit.

## 0542Fh (Main-ROM) **FRMQNT** (« Formula Quantificator »)

Rôle : Évaluer une expression dans le programme et la retourner au format d'un entier.

Entrée : HL = Adresse du début de l'expression.

Sortie : HL = Adresse de l'expression suivante, DE = Entier résultant.

Modifie : AF, BC, DE, HL.

## 05439h (Main-ROM)

Rôle : Évaluer le nombre à DAC et le retourner dans la paire de registres HL.

Entrée : DAC (F7F6h) = Nombre  
VALTYP (0F663h) = 2, 4, 8.

Sortie : HL = Nombre entier.

Modifie : AF, BC, DE, HL.

Note : Utilisé par POKE, PEEK et BLOAD.

### 05597h (Main-ROM)     **GETYPR** (« Get Type into Register A »)

Rôle : Détermination du type du nombre à DAC (« Decimal Acumulator »).

Entrée : Rien

sortie : A = FFh si entier, 0 si chaîne de caractère, 1 simple précision, 5 si double précision.  
F = SF\*, CF, <sup>PF</sup>/VF à 1 et ZF à 0 si chiffre entier ;  
SF, ZF, <sup>PF</sup>/VF\* à 0 et CF à 1 si chiffre simple précision ;  
SF, ZF, CF\* à 0 et <sup>PF</sup>/VF à 1 si chiffre double précision ;  
CF, ZF\*, <sup>PF</sup>/VF à 0 et SF à 1 si chaîne de caractères.  
\* Indicateur sur lequel se baser pour connaître le type.

Modifie : AF, BC, DE, HL.

Note : Pareil que GETYPR (0028h)

### 05EA4h (Main-ROM)     **PTRGET** (« Pointer Get »)

Rôle : Recherche d'une variable.

Entrée : HL = Adresse du programme, SUBFLG = 0 variable unique sinon tableau.

Sortie : HL = Adresse suivante, DE = Pointeur vers la première variable.

Modifie : AF, BC, DE, HL.

### 066A7h (Main-ROM)     **PPSWRT**

Rôle : Cette routine contient un POP AF et un RET. (voir la note)

Entrée : Adresse placée en haut de la pile.

Modifie : AF.

Note : Cette routine contient ce qui suit, Donc, mieux vaut appeler sa propre routine sous MSX-DOS si besoin.

PPSWRT:

```
pop    af
ret
```

### 067D0h (Main-ROM)     **FRESTR** (« FREe STRing »)

Rôle : Cette routine libère la mémoire occupée par la chaîne temporaire dont l'adresse est contenue dans DAC.

Entrée : DAC = Descripteur de chaîne  
VALTYP (0F663h) = 3

Sortie : HL = Pointeur vers le descripteur de chaîne

Modifie : AF, BC, DE, HL.

Note : L'adresse de la chaîne est comparée à celle de FRETOP pour voir s'il s'agit de la chaîne la plus basse dans la zone de stockage de chaînes, si ce n'est pas le cas la routine se termine. Autrement, la longueur de la chaîne est ajoutée à FRETOP, qui est ensuite mise à jour avec cette nouvelle valeur, libérant ainsi l'espace de stockage occupé par la chaîne.

## 4.5 Table des sauts du Bios de la Sub-ROM

Avec l'évolution du standard vers le MSX2, la ROM contenant le Bios étant trop petite, il a fallu ajouter une ROM auxiliaire afin d'étendre le Bios. Cette ROM s'appelle « Sub-ROM ». Celle-ci contient aussi une table des sauts qui permet d'appeler les routines supplémentaires spécifiques au MSX2 et aux générations supérieures plus efficacement qu'en passant par la Main-ROM. Pour appeler une de ces routines, vous pouvez passer la routine EXTROM (0015Fh) ou SUBROM (0015Ch) de la Main-ROM ou sélectionner vous-même la Sub-ROM et passer par la table des sauts. Vous pouvez connaître le Slot de la Sub-ROM en lisant la variable système EXBRSA (0FAF8h). Contrairement aux routines du BIOS, dont la table des sauts contient que des instructions de saut (JP), la table de la Sub-ROM contient des instructions de saut précédés de l'instruction pour activer les interruptions (EI), excepté pour la routine NMI..

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

### 00066h (Sub-ROM)      **NMI** (« NMI »)

Rôle :            Réservé pour la routine d'interruptions non-masquables.  
Note :            Appel au Hook (0FDD6h).

### 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 :

```
EXTROM equ 0015Fh
PAINT   equ 00069h

        org 0c000h
DEBUT:
        ld hl, DATA
        ld ix, PAINT
        call EXTROM
        ret

DATA:   db '(100,100),5,8',0
```

### 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.

### 00051h (Sub-ROMp)      **ATRSCN** (« ATRibute SCaN »)

- Rôle :            Recherche d'une couleur.
- Entrée :        HL = Pointeur sur le texte Basic.  
                  GXPOS (0FCB3h) = Abscisse du pixel de la couleur trouvée.  
                  GYPOS (0FCB5h) = Ordonnée du pixel de la couleur trouvée.
- Sortie :        HL = Pointeur réactualisé.
- Modifie :      Tout.
- Notes :        - Valide dans les SCREEN 5 à 8 / 12.  
                  - Utilisé par l'interpréteur du Basic.

### 00075h (Sub-ROM)      **GLINE** (« Graphic LINE »)

- Rôle :            Tracé d'une ligne.
- Entrée :        HL = Pointeur de texte du Basic.
- Sortie :        HL = Pointeur réactualisé.  
                  GXPOS (0FCB3h) = Abscisse du pixel d'arrivée.  
                  GYPOS (0FCB5h) = Ordonnée du pixel d'arrivée.
- Modifie :      Tout.
- Notes :        - Valide dans les SCREEN 5 à 8 / 12.  
                  - Utilisé par l'interpréteur du Basic.

### 00079h (Sub-ROM)      **DOBOXF** (« DO BOX Fill »)

Rôle :            Tracé d'un rectangle plein.

Entrée :        HL = Pointeur vers le texte Basic.

Sortie :         HL = Pointeur réactualisé.  
                  GXPOS (0FCB3h) = Abscisse du pixel d'arrivée.  
                  GYPOS (0FCB5h) = Ordonnée du pixel d'arrivée.

Modifie :       Tout.

Notes :        - Valide dans les SCREEN 5 à 8 / 12.  
                  - Utilisé par l'interpréteur du Basic.

### 0007Dh (Sub-ROM)      **DOLINE** (« DO LINE »)

Rôle :            Tracé de ligne.

Entrée :        HL = Pointeur de texte du Basic.

Sortie :         HL = Pointeur réactualisé.  
                  GXPOS (0FCB3h) = Abscisse du pixel d'arrivée.  
                  GYPOS (0FCB5h) = Ordonnée du pixel d'arrivée.

Modifie :       Tout.

Notes :        - Valide dans les SCREEN 5 à 8 / 12.  
                  - Utilisé par l'interpréteur du Basic.

### 00081h (Sub-ROM)      **BOXLIN** (« BOX LINE »)

Rôle :            Tracé d'un rectangle.

Entrée :        HL = Pointeur de texte du Basic.

Sortie :         HL = Pointeur réactualisé.  
                  GXPOS (0FCB3h) = Abscisse du pixel d'arrivée.  
                  GYPOS (0FCB5h) = Ordonnée du pixel d'arrivée.

Modifie :       Tout.

Notes :        - Valide dans les SCREEN 5 à 8 / 12.  
                  - Utilisé par l'interpréteur du Basic.

## 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 :

```
EXTROM    equ    0015fh
DOGRPH     equ    00085h
GXPOS      equ    0fcb3h
GYPOS      equ    0fcb5h
ATRBYT     equ    0f3f2h
LOGOPR     equ    0fb02h

           org    0c000h

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
```

00089h (Sub-ROM)      **GRPPRTS** (« GRaPhic PRinT »)

Rôle : Affichage d'un caractère sur écran graphique.

Entrée :      A = Code ASCII du caractère.

GRPACX (0FCB7h) = Abscisse du pixel où doit s'afficher le caractère.

GRPACY (0FCB9h) = Ordonnée du pixel où doit s'afficher le caractère.

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

LOGOPR (0FB02h) = Opérateur logique.

Sortie : Rien.

Modifie : Rien.

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

Voir DOGRPH (00085h en Sub-ROM) pour l'explication de LOGOPR.

0008Dh (Sub-ROM)      **SCALXYS** (« SCALe X & Y »)

|        |                                                                             |
|--------|-----------------------------------------------------------------------------|
| Rôle : | Vérification et éventuellement ajustement des valeurs du curseur graphique. |
|--------|-----------------------------------------------------------------------------|

Entrée : BC = Abscisse (0 ~ FFFFh).

DE = Ordonnée (0 ~ FFFFh).

Sortie : BC = Abscisse valide.

DE = Ordonnée valide.

F = Indicateur CF à 1 si une des coordonnées était hors de l'écran.

Modifie : AF.

Note : Lorsque l'une des deux coordonnées est négative, cette routine prend 0 comme nouvelle coordonnée. De même, si une coordonnée dépasse la valeur maximale autorisée (255 ou 511 pour l'abscisse, 191 ou 211 pour l'ordonnée), la routine prend la valeur maximale comme nouvelle coordonnée.

00091h (Sub-ROM)      **MAPXYCS** (« 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)      **READCS** (« 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)      **SETATRS** (« SET ATtribute »)

Rôle :            Modification de l'octet d'attribut ATRBYT (0F3F2h).

Entrée :        A = Valeur du nouvel attribut. (Couleur de tracé des routines graphiques.)

Sortie :        F = Indicateur CF à 1 si attribut non valide (>3 en SCREEN 6 ; >15 en SCREEN 5 et 7).

Modifie :      F.

### 0009Dh (Sub-ROM)      **SETCS** (« SET the Cursor »)

Rôle :            Placer un pixel à l'écran au curseur graphique 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 and move to the RIGHT the graphic Cursor »)

Rôle :            Déplacement de la coordonnée actuelle d'un pixel vers la droite tant que celle-ci reste dans la partie visible de l'écran.

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)      RIGHTCS (« move to the RIGHT the graphic Cursor »)**

Rôle : Déplacement de la coordonnée actuelle 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 and move to the LEFT the graphic Cursor »)**

Rôle : Déplacement de la coordonnée actuelle d'un pixel vers la gauche tant que celle-ci reste dans la partie visible de l'écran.  
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)      LEFTCS (« move to the LEFT the graphic Cursor »)**

Rôle : Déplacement de la coordonnée actuelle 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)      TDOWNCS (« Test and move to the DOWN the graphic Cursor »)**

Rôle : Déplacement de la coordonnée actuelle du curseur d'un pixel vers le bas tant que celle-ci reste dans la partie visible de l'écran.  
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)      DOWNCS (« move to the DOWN the graphic Cursor »)**

Rôle : Déplacement de la coordonnée actuelle du curseur 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)      **TUPCS** (« Test and move to the UP the graphic Cursor »)

Rôle : Déplacement du curseur graphique d'un pixel vers le haut tant que celle-ci reste dans la partie visible de l'écran.

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)      **UPCS** (« move to the UP the graphic Cursor »)

Rôle : Déplacement du curseur graphique 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)      **SCANRS** (« SCAN to 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)      **SCANLS** (« SCAN to 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)      **NVXFL**

- 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)      **CHGMODS** (« CHanGe MODE »)

- Rôle :            Changement de mode d'écran (SCREEN X).
- Entrée :        A = SCREEN désiré. (0 ~ 8 / 12)  
                 LINL32 (0F3AFh) = Nombre de caractères par lignes pour le SCREEN 1.  
                 LINL40 (0F3AEh) = Nombre de caractères par lignes pour le SCREEN 0.
- Sortie :        SCRMOD (0FCAFh) = Nouveau mode d'écran.
- Modifie :      Tout.
- Notes :        - 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.  
                 - Voir le registre 25 à la page 279, pour mettre en mode d'écran 10/11 ou 12.

## 000D5h (Sub-ROM)      **INITXTS** (« INItialize TeXT mode »)

- Rôle :            Initialisation du mode SCREEN 0.
- Entrée :        TXTNAM (0F3B3h) = Adresse de la table des positons de caractère.  
                  TXTCOL (0F3B5h) = Adresse de la table des couleurs de caractère.  
                  TXTCGP (0F3B7h) = Adresse de la table des formes de caractère .  
                  TXTCOL (0F3B5h) = Adresse de la table des couleurs.  
                  LINL40 (0F3AEh) = Nombre de caractères par lignes.  
                  FORCLR (0F3E9h) = Couleur de texte.  
                  BAKCLR (0F3EAh) = Couleur de fond.
- Sortie :        Rien.
- Modifie :      Tout.
- Note :        Il n'est pas obligatoire de définir TXTNAM, etc car le MSX initialise leur valeur à l'allumage.

## 000D9h (Sub-ROM)      **INIT32S** (« INItialize Text 32 mode »)

- Rôle :            Initialisation du mode SCREEN 1.
- Entrée :        T32NAM (0F3BDh) = Adresse de la table des positons de caractère.  
                  T32COL (0F3BFh) = Adresse de la table des couleurs de caractère.  
                  T32CGP (0F3C1h) = Adresse de la table des formes de caractère.  
                  T32ATR (0F3C3h) = Adresse de la table des attributs de Sprite.  
                  T32PAT (0F3C5h) = Adresse de la table des formes de Sprite.  
                  LINL32 (0F3AFh) = Nombre de caractères par lignes.  
                  FORCLR (0F3E9h) = Couleur de texte.  
                  BAKCLR (0F3EAh) = Couleur de fond.  
                  BDRCLR (0F3EBh) = Couleur de bordure.
- Sortie :        Rien.
- Modifie :      Tout.
- Note :        Il n'est pas obligatoire de définir T32NAM, etc car le MSX initialise leur valeur à l'allumage.

## 000DDh (Sub-ROM)      **INIGRPS** (« INItialize GRaPhic mode »)

- Rôle :            Initialisation du mode graphique 256x192 (SCREEN 2).
- Entrée :        GRPNAM (0F3C7h) = Adresse de la table des positions de motif.  
                 GRPCOL (0F3C9h) = Adresse de la table des couleurs de motif.  
                 GRPCGP (0F3CBh) = Adresse de la table des formes de motif.  
                 GRPATR (0F3CDh) = Adresse de la table des attributs de Sprite.  
                 GRPPAT (0F3CFh) = Adresse de la table des formes de Sprite.  
                 FORCLR (0F3E9h) = Couleur de texte.  
                 BAKCLR (0F3EAh) = Couleur de fond.  
                 BDRCLR (0F3EBh) = Couleur de bordure.
- Sortie :        Rien.
- Modifie :      Tout.
- Note :        Il n'est pas obligatoire de définir GRPNAM, etc car le MSX initialise leur valeur à l'allumage.

## 000E1h (Sub-ROM)      **INIMLTS** (« INItialize MuLTicolor mode »)

- Rôle :            Initialisation du mode SCREEN 3.
- Entrée :        MLTNAM (0F3D1h) = Adresse de la table des positions de motif.  
                 MLTCOL (0F3D3h) = Adresse de la table des couleurs de motif.  
                 MLTCGP (0F3D5h) = Adresse de la table des formes de motif.  
                 MLTATR (0F3D7h) = Adresse de la table des attributs de Sprite.  
                 MLTPAT (0F3D9h) = Adresse de la table des formes de Sprite.  
                 FORCLR (0F3E9h) = Couleur de texte.  
                 BAKCLR (0F3EAh) = Couleur de fond.  
                 BDRCLR (0F3EBh) = Couleur de bordure.
- Sortie :        Rien.
- Modifie :      Tout.
- Note :        Il n'est pas obligatoire de définir MLTNAM, etc car le MSX initialise leur valeur à l'allumage.

## 000E5h (Sub-ROM)      **SETTXTS** (« SET TeXT mode »)

- Rôle :            Passage direct en mode texte 40x24.
- Entrée :        TXTNAM (0F3B3h) = Adresse de la table des positions de caractère.  
                 TXTCGP (0F3B7h) = Adresse de la table des formes de caractère.
- 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)      **SETT32S** (« SET Text 32 mode »)

- Rôle :            Passage direct en mode texte 32x24.
- Entrée :        T32NAM (0F3BDh) = Adresse de la table des positions de caractère.  
                  T32COL (0F3BFh) = Adresse de la table des couleurs de caractère.  
                  T32CGP (0F3C1h) = Adresse de la table des formes de caractère.  
                  T32ATR (0F3C3h) = Adresse de la table des attributs de Sprite.  
                  T32PAT (0F3C5h) = Adresse de la table des formes de Sprite.
- 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)      **SETGRPS** (« SET GRaPhic mode »)

- Rôle :            Passage direct en mode SCREEN 2.
- Entrée :        GRPNAM (0F3C7h) = Adresse de la table des positions de modif.  
                  GRPCOL (0F3C9h) = Adresse de la table des couleurs de modif.  
                  GRPCGP (0F3CBh) = Adresse de la table des formes de modif.  
                  GRPATR (0F3CDh) = Adresse de la table des attributs de Sprite.  
                  GRPPAT (0F3CFh) = Adresse de la table des formes de Sprite.
- 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)      **SETMLTS** (« SET MuLTicolor mode »)

- Rôle :            Passage direct en mode SCREEN 3.
- Entrée :        MLTNAM (0F3D1h) = Adresse de la table des positions de modif.  
                  MLTCOL (0F3D3h) = Adresse de la table des couleurs de modif.  
                  MLTCGP (0F3D5h) = Adresse de la table des formes de modif.  
                  MLTATR (0F3D7h) = Adresse de la table des attributs de Sprite.  
                  MLTPAT (0F3D9h) = Adresse de la table des formes de Sprite.
- 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)      **CLRSPRS** (« CLear Sprite »)

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

### 000F9h (Sub-ROM)      **CALPATS** (« CALculate address of PATtern table »)

Rôle :            Donne l'adresse de la table des formes (patrons) de Sprite en VRAM.  
Entrée :          A = Numéro de plan du Sprite.  
Sortie :          HL = Adresse cherchée.  
Modifie :        AF, DE, HL.  
Note :            Idem CALPAT en Main-ROM.

### 000FDh (Sub-ROM)      **CALATRS** (« CALculate address of ATRibute table »)

Rôle :            Donne l'adresse de la table des attributs de 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)      **GSPSIZS** (« Get Sprite SIZE »)

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

### 00105h (Sub-ROM)      **GETPAT** (« GET PATtern »)

Rôle :            Obtenir les huit octets qui définissent la forme d'un caractère.  
Entrée :          A = Code ASCII du caractère.  
Sortie :          PATWRK (0FC40h) = Adresse des huit octets qui définissent la forme 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 ~ FFFFh)  
                    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)      **RDVRMS** (« ReaD VRaM »)

Rôle :            Lecture d'une case mémoire de la VRAM  
Entrée :        HL = Adresse de case mémoire (0 ~ FFFFh)  
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)      **CHGCLRS** (« CHanGe CoLoR »)

Rôle :            Change les couleurs de l'écran spécifié avec les valeurs par défaut.  
Entrée :        A = Mode d'écran.  
                    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)      **CLSS** (« 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)      DSPFNKS (« 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 ZF 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)      **BASE** (« BASE »)

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)      **BEEPS** (« BEEP buzzer »)

Rôle :            Initialise l'instruction PLAY et émet d'un bref « bip » sonore.  
Entrée :        Rien.  
Sortie :        Rien.  
Modifie :      Tout.  
Note :        Ce son est produit par le PSG.

#### 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** (« Set DeFault SCReen »)

Rôle :            Restaure l'écran initial avec les paramètres sauvegardés dans la SRAM de l'horloge (largeur d'écran, couleurs, etc).  
Entrée :        Mettez l'indicateur Carry à 1 pour afficher le texte des touches de fonction.  
Sortie :        Rien.  
Modifie :      Tout.

#### 00189h (Sub-ROM)      **SETSCR**    SET start SCReen »)

Rôle :            Similaire à 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.

```
EXTROM equ 0015fh
BLTVV   equ 00191h
SX      equ 0f562h      ; abscisse du pixel de départ
SY      equ SX+2        ; ordonnée du 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

      org 0c000h
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

      ld hl,SX
      ld ix,BLTVV       ; adresse de la routine à appeler
      call EXTROM
      ret
```

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)   | 05FFFh | (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.

```
EXTROM equ 0015fh
BLTVM  equ 00195h
DPTR   equ 0f562h
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

      org 0c000h
DEBUT:
      ld hl,0d000h
      ld (DPTR),hl      ; Adresse du début des données
      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

      ld hl,DPTR
      ld ix,BLTVM       ; Adresse de la routine à appeler
      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
```

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.

```
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

org 0c000h
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

ld hl,SX
ld ix,BLTMV ; Adresse de la routine à appeler
call EXTROM
ret
```

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 :        F = CF est mis à 0.

Modifie :      Tout.

Exemple d'utilisation en assembleur :

```
; Faite l'équivalent de COPY "A:ESSAI.SC7" to (16,32)
```

```
EXTROM equ 0015fh
BLTVD   equ 0019dh
FNPTR   equ 0f562h      ; Nom du fichier dans BUF
DX       equ FNPTR+4     ; Abscisse de destination
DY       equ DX+2        ; Ordonnée de destination
ARG      equ DY+7
LOGOP    equ ARG+1       ; Opérateur logique

                db 0feh          ; Entête pour les
                dw DEBUT,END,DEBUT ; fichiers binaires

                org 0c000h
DEBUT:
                ld hl,NOM
                ld (FNPTR),hl
                ld hl,10h
                ld (DX),hl
                ld hl,20h
                ld (DY),hl
                xor a           ; Mise 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,0
END:
```

Assemblez la routine sous le nom "BLTVD.BIN", puis lancer la avec le programme Basic suivant :

```
10 CLEAR,&HC000: SCREEN7
20 BLOAD"BLTVD.BIN": DEFUSR=&HC000: A$=USR(0)
```

Vous aurez le nom du fichier en sortie dans la variable A\$.

Pour plus de précisions, voir les routines BLTVV et BLTDV.

## 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.

```
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

db 0feh ; Entête pour les
dw DEBUT,END,DEBUT ; fichiers binaires

org 0c000h
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 !!!

ld hl,SX
ld ix,BLTDV ; Adresse de la routine à appeler
call EXTROM
ret

NOM: db 022h,'A:ESSAI.SC7',022h,0
END:
```

Assemblez la routine sous le nom "BLTDV.BIN", puis lancer la avec le programme Basic suivant :

```
10 CLEAR,&HC000: SCREEN7: CIRCLE(128,106),105,6
20 BLOAD"BLTDV.BIN": DEFUSR=&HC000: A$=USR(0)
```

Vous aurez le nom du fichier en sortie dans la variable A\$.

Pour plus de précisions, voir les routines BLTVV et BLTVD.

## 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.

```
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

org 0c000h
DEBUT:
    ld hl,NOM
    ld (FNPTR),hl
    ld hl,0d000h
    ld (SPTR),hl
    ld hl,0d020h
    ld (EPTR),hl

    ld hl,FNPTR
    ld ix,BLTMD ; Adresse de la routine à appeler
    call EXTROM
    ret
;
NOM: db 022h,'A:ESSAI.SC7',022h,0
END:
```

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 un COPY <0D000h> TO "A:ESSAI.SC7" en assembleur

```
EXTROM equ 0015fh
BLTDM  equ 001a9h
SPTR   equ 0f562h      ; Adresse du début des données
EPTR   equ SPTR+2      ; Adresse de fin des données en mémoire
FNPTR  equ SPTR+4      ; Pointeur sur le nom de fichier

DEBUT:  org 0c000h
        ld hl,NOM
        ld (FNPTR),hl
        ld hl,0d000h
        ld (SPTR),hl
        ld hl,0d020h
        ld (EPTR),hl

        ld hl,SPTR
        ld ix,BLTDM     ; Adresse de la routine à appeler
        call EXTROM
        ret

NOM:
        db 022h,'A:ESSAI.SC7',022h,0

END:
```

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 ou plus récent, il est possible d'appeler la routine **GTPAD** en Main-ROM à place. C'est juste un peu plus rapide d'appeler **NEWPAD** directement.

                  - Voir la routine **GTPAD** (000DBh) en Main-ROM pour connaître les opérations possibles.

### 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 CHGMODS (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 :      F.

Note :          Routine présente sur les MSX japonais ayant une Kanji-ROM.

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

Rôle :            Lecture d'un registre de l'horloge interne (RTC). L'état des registres sont conservés grâce à une pile ou une batterie selon le MSX.

Entrée :        C = Numéro du bloc et du registre à lire.

| bit 7 | bit 6 | bit 5 | bit 4 | bit 3             | bit 2 | bit 1 | bit 0 |
|-------|-------|-------|-------|-------------------|-------|-------|-------|
| -     | -     | Bloc  |       | Registre (0 ~ 12) |       |       |       |

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

Modifie :      F.

Note :        L'adresse mémoire se décompose de la façon. Les cases grises sont inutilisées.

Bloc 0 :

Ce bloc sert a régler ou donner l’heure et la date.

| Registre | bit 3                                  | bit 2                               | bit 1                                     | bit 0            |
|----------|----------------------------------------|-------------------------------------|-------------------------------------------|------------------|
| 0        | Compteur de l'unité des secondes       |                                     |                                           |                  |
| 1        | -                                      | Compteur de la dizaine des secondes |                                           |                  |
| 2        | Compteur de l'unité des minutes        |                                     |                                           |                  |
| 3        | -                                      | Compteur de la dizaine des minutes  |                                           |                  |
| 4        | Compteur de l'unité des heures         |                                     |                                           |                  |
| 5        | -                                      | -                                   | Compteur de la dizaine des heures         |                  |
| 6        |                                        | Compteur du jour des semaines       |                                           |                  |
| 7        | Compteur de l'unité des jours des mois |                                     |                                           |                  |
| 8        | -                                      | -                                   | Compteur de la dizaine des jours des mois |                  |
| 9        | Compteur de l'unité des mois           |                                     |                                           |                  |
| 10 (Ah)  | -                                      | -                                   | -                                         | Dizaine des mois |
| 11 (Bh)  | Compteur de l'unité des années         |                                     |                                           |                  |
| 12 (Ch)  | Compteur de la dizaine des années      |                                     |                                           |                  |



## Bloc 1 (01 en binaire) :

Ce bloc sert à régler une alarme ou un timer, ainsi que le type d'affichage de l'heure et indique si l'année est bissextile ou non.

| Registre | bit 3                             | bit 2                           | bit 1                               | bit 0         |
|----------|-----------------------------------|---------------------------------|-------------------------------------|---------------|
| 0        | -                                 |                                 |                                     |               |
| 1        | -                                 |                                 |                                     |               |
| 2        | Unité des minutes de l'alarme     |                                 |                                     |               |
| 3        | -                                 | Dizaine des minutes de l'alarme |                                     |               |
| 4        | Unité de l'heure de l'alarme      |                                 |                                     |               |
| 5        | -                                 | -                               | Dizaine de l'heure de l'alarme      |               |
| 6        |                                   | Jour de semaine de l'alarme     |                                     |               |
| 7        | Unité du jour du mois de l'alarme |                                 |                                     |               |
| 8        | -                                 | -                               | Dizaine du jour du mois de l'alarme |               |
| 9        | -                                 |                                 |                                     |               |
| 10 (Ah)  | -                                 | -                               | -                                   | 12 / 24 heure |
| 11 (Bh)  | -                                 | -                               | Compteur pour année bissextile      |               |
| 12 (Ch)  | -                                 |                                 |                                     |               |

Note : L'alarme et le timer n'est pas utilisé par le système. L'interface SCSI Gouda (Novaxis) utilise certains de ces registres pour stocker quelques paramètres comme l'ID hôte et l'ID cible.

Bloc 2 (10 en binaire) :

Ce bloc sert à stocker des paramètres du système.

| Registre | bit 3                                                                                                                                                                                                                                  | bit 2                                        | bit 1                            | bit 0        |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------|----------------------------------|--------------|
| 0        | Identificateur (1010 si la RAM est initialisée)                                                                                                                                                                                        |                                              |                                  |              |
| 1        | Décalage horizontale du SET ADJUST (-8 à 7)                                                                                                                                                                                            |                                              |                                  |              |
| 2        | Décalage verticale du SET ADJUST (-8 à 7)                                                                                                                                                                                              |                                              |                                  |              |
| 3        | -                                                                                                                                                                                                                                      | -                                            | -                                | SCREEN 0 / 1 |
| 4        | 4 bits de poids faible de la valeur de WIDTH                                                                                                                                                                                           |                                              |                                  |              |
| 5        | -                                                                                                                                                                                                                                      | 3 bits de poids fort de la valeur de WIDTH   |                                  |              |
| 6        | Couleur du texte par défaut (0 ~ 15)                                                                                                                                                                                                   |                                              |                                  |              |
| 7        | Couleur de fond par défaut (0 ~ 15)                                                                                                                                                                                                    |                                              |                                  |              |
| 8        | Couleur de bordure par défaut (0 ~ 15)                                                                                                                                                                                                 |                                              |                                  |              |
| 9        | Vitesse de transfert<br>0 = 1200 baud<br>1 = 2400 baud                                                                                                                                                                                 | 0 = Imprimante MSX<br>1 = Imprimante Non MSX | Cliquetis des touches du clavier | KEY OFF / ON |
| 10 (Ah)  | Type de son BEEP                                                                                                                                                                                                                       |                                              | Volume du BEEP                   |              |
| 11 (Bh)  | -                                                                                                                                                                                                                                      | -                                            | Couleur de l'écran de démarrage  |              |
| 12 (Ch)  | Code de la zone :<br>0 = Japon ; 1 = USA ; 2 = International ; 3 = Grande Bretagne ;<br>4 = France ; 5 = Allemagne ; 6 = Italie ; 7 = Espagne ;<br>8 = Emirats arabes ; 9 = Corée ; 10 = URSS ;<br>11 ~ 15 = Indéfinis (au 05/02/1986) |                                              |                                  |              |

Note : Le code de la zone n'est pas toujours le bon par défaut. Par exemple, le Sony HB-F9S qui était vendu en Espagne indique 5 au lieu de 7.

### Bloc 3 (11 en binaire) :

Ce bloc sert à stocker un titre, un mot de passe ou l'invite du BASIC. L'invite est « OK » par défaut.

| Registre | bit 3                                                                                                                        | bit 2 | bit 1 | bit 0 |
|----------|------------------------------------------------------------------------------------------------------------------------------|-------|-------|-------|
| 0        | identificateur (0 ~ 15) :<br>0 = Titre ; 1 = Mot de passe ; 2 = Invite (Prompt) ;<br>3 à 15 = <i>Indéfinis</i> au 05/02/1986 |       |       |       |
| 1        | 4 bits de poids faible du premier caractère                                                                                  |       |       |       |
| 2        | 4 bits de poids fort du premier caractère                                                                                    |       |       |       |
| 3        | 4 bits de poids faible du second caractère                                                                                   |       |       |       |
| 4        | 4 bits de poids fort du second caractère                                                                                     |       |       |       |
| 5        | 4 bits de poids faible du troisième caractère                                                                                |       |       |       |
| 6        | 4 bits de poids fort du troisième caractère                                                                                  |       |       |       |
| 7        | 4 bits de poids faible du quatrième caractère                                                                                |       |       |       |
| 8        | 4 bits de poids fort du quatrième caractère                                                                                  |       |       |       |
| 9        | 4 bits de poids faible du cinquième caractère                                                                                |       |       |       |
| 10 (Ah)  | 4 bits de poids fort du cinquième caractère                                                                                  |       |       |       |
| 11 (Bh)  | 4 bits de poids faible du sixième caractère                                                                                  |       |       |       |
| 12 (Ch)  | 4 bits de poids fort du sixième caractère                                                                                    |       |       |       |

### Exemple d'utilisation :

```
EXTROM    equ    0015fh
REDCLK    equ    001f5h

          org    0c000h
DEBUT:
          ld     c,02ch          ; bloc 2 = registre 12
          ld     ix,REDCLK
          call   EXTROM
          ret                    ; lors du retour A contient 4
```

### 001F9h (Sub-ROM)      **WRTCLK** (« WRiTe CLoCK »)

Rôle :            Écriture dans un registre de l'horloge interne (RTC). L'état des registres sont conservés grâce à une pile ou une batterie selon le MSX.

Entrée :        C = Numéro du bloc et du registre  
                 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.

## 5 Les variables et zones de travail du système

### 5.1 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 routines du 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 et ne pas écraser n'importe quoi.

### 5.2 Liste des variables et des zones de travail système

| Adresse | Nom    | Long. | Fonction                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------|--------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0F380h  | RDPRIM | 5     | <p>Sous-routine de RDSLT (Main-ROM) qui sert à lire un octet en mémoire à l'adresse indiquée dans les Slot primaire spécifié, puis replacer les Slot dans l'état au moment de l'appel.</p> <p>Pour appeler cette routine, le registre A doit contenir l'état actuel du registre des Slot primaires sauf les 2 bits qui doivent indiquer le Slot dans lequel vous voulez lire l'octet. Le registre D doit contenir l'état actuel du registre des Slot primaires et HL l'adresse de l'octet à lire. L'adresse ne doit pas être en dehors de la plage du Slot choisi.</p> <p>En sortie, le registre E contiendra l'octet lu.</p>                                                                        |
| 0F385h  | WRPRIM | 7     | <p>Sous-routine de WRS�T (Main-ROM) qui sert à écrire un octet en mémoire à une adresse dans le Slot primaire, puis replacer les Slot dans l'état au moment de l'appel.</p> <p>Pour appeler cette routine, le registre A doit contenir l'état actuel du registre des Slot primaires sauf les 2 bits qui doivent indiquer le Slot dans lequel vous voulez écrire l'octet. Le registre D doit contenir l'état actuel du registre des Slot primaires, le registre E l'octet à écrire et HL l'adresse de l'octet à écrire. L'adresse ne doit pas être en dehors de la plage du Slot choisi.</p>                                                                                                          |
| 0F38Ch  | CLPRIM | 14    | <p>Sous-routine de CALSLT (Main-ROM) qui sert à appeler une routine dans un Slot primaire, puis replacer les Slot dans l'état au moment de l'appel.</p> <p>En entrée, le registre A doit contenir l'état actuel du registre des Slot primaires sauf les 2 bits qui doivent indiquer le Slot à sélectionner sur la plage dans laquelle la routine à appeler se trouve. Le registre A' doit contenir la valeur du registre A dont la routine à appeler aurait besoin. L'octet de point fort au dessus de la pile doit contenir l'état actuel du registre des Slots primaires. IX doit contenir l'adresse de la routine à appeler. L'adresse ne doit pas être en dehors de la plage du Slot choisi.</p> |

|        |        |    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------|--------|----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0F39Ah | USRTAB | 20 | Adresses des routines définies par l'instruction DEFUSR X=.<br>USRTAB = Adresse de la routine définie par DEFUSR0=<br>USRTAB+2 = Adresse de la routine définies par DEFUSR1=<br>USRTAB+4 = Adresse de la routine définies par DEFUSR2=<br>USRTAB+6 = Adresse de la routine définies par DEFUSR3=<br>USRTAB+8 = Adresse de la routine définies par DEFUSR4=<br>USRTAB+10 = Adresse de la routine définies par DEFUSR5=<br>USRTAB+12 = Adresse de la routine définies par DEFUSR6=<br>USRTAB+14 = Adresse de la routine définies par DEFUSR7=<br>USRTAB+16 = Adresse de la routine définies par DEFUSR8=<br>USRTAB+18 = Adresse de la routine définies par DEFUSR9=<br>Note :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 Sprite en mode SCREEN 1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 0F3C5h | T32PAT | 2  | Adresse de la table des formes de Sprite 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 Sprite en mode SCREEN 2.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 0F3CFh | GRPPAT | 2  | Adresse de la table des formes de Sprite 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 | MLTATR | 2 | Adresse de la table des attributs de Sprite en mode SCREEN 3.                                                                                                                                                                                                             |
| 0F3D9h | MLTPAT | 2 | Adresse de la table des formes de Sprite en mode SCREEN 3.                                                                                                                                                                                                                |
| 0F3DBh | CLIKSW | 1 | Cliquetis des touches. 0 pour désactiver ; autre pour activer.                                                                                                                                                                                                            |
| 0F3DCh | CSRY   | 1 | Ordonnée du curseur.                                                                                                                                                                                                                                                      |
| 0F3DDh | CSRX   | 1 | Abscisse du curseur.                                                                                                                                                                                                                                                      |
| 0F3DEh | CNSDFG | 1 | Indicateur pour savoir si les touches de fonction sont affichées (255) ou pas (0).                                                                                                                                                                                        |
| 0F3DFh | RG0SAV | 1 | Contenu du <a href="#">registre 0</a> du VDP.                                                                                                                                                                                                                             |
| 0F3E0h | RG1SAV | 1 | Contenu du <a href="#">registre 1</a> du VDP.                                                                                                                                                                                                                             |
| 0F3E1h | RG2SAV | 1 | Contenu du <a href="#">registre 2</a> du VDP.                                                                                                                                                                                                                             |
| 0F3E2h | RG3SAV | 1 | Contenu du <a href="#">registre 3</a> du VDP.                                                                                                                                                                                                                             |
| 0F3E3h | RG4SAV | 1 | Contenu du <a href="#">registre 4</a> du VDP.                                                                                                                                                                                                                             |
| 0F3E4h | RG5SAV | 1 | Contenu du <a href="#">registre 5</a> du VDP.                                                                                                                                                                                                                             |
| 0F3E5h | RG6SAV | 1 | Contenu du <a href="#">registre 6</a> du VDP.                                                                                                                                                                                                                             |
| 0F3E6h | RG7SAV | 1 | Contenu du <a href="#">registre 7</a> du VDP.                                                                                                                                                                                                                             |
| 0F3E7h | STATFL | 1 | Contenu du <a href="#">registre de statut 0</a> du VDP.                                                                                                                                                                                                                   |
| 0F3E8h | TRGFLG | 1 | État des boutons des manettes (0 = Bouton pressé)<br>bit 0 = Barre d'espacement ;<br>bit 1~3 = <i>Inutilisés</i> ;<br>bit 4 = Bouton 1 de la manette 1 ;<br>bit 5 = Bouton 2 de la manette 1 ;<br>bit 6 = Bouton 1 de la manette 2 ;<br>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 fils d'attente 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 | Compteur pour l'intervalle entre deux scrutations du clavier.                                                                                                                                                                                                                                                                                                                                                                       |
| 0F3F7h | REPCNT | 1 | Compteur pour l'intervalle avant de commencer la fonction de répétition automatique. (1 par défaut)                                                                                                                                                                                                                                                                                                                                 |
| 0F3F8h | PUTPNT | 2 | Adresse du premier emplacement libre dans le cache du clavier. (0FBF0h à l'initialisation)                                                                                                                                                                                                                                                                                                                                          |
| 0F3FAh | GETPNT | 2 | Adresse du premier caractère/code de la ligne actuelle (derrière le dernier code 0Dh entré) dans le cache du clavier. (0FBF0h à l'initialisation)                                                                                                                                                                                                                                                                                   |
| 0F3FCh | CS120  | 5 | Zone de travail du lecteur de cassette. (Jusqu'au MSX2+)<br>CS120 = Longueur du signal LOW du bit 0. (53h par défaut)<br>CS120+1 = Longueur du signal HIGH du bit 0. (5Ch par défaut)<br>CS120+2 = Longueur du signal LOW du bit 1. (26h par défaut)<br>CS120+3 = Longueur du signal HIGH du bit 1. (2Dh par défaut)<br>CS120+4 = Longueur du signal d'entête calculé par<br>$\text{HEADLEN} * 2 / 256$ (HEADLEN = 2000 par défaut) |
| 0F401h | CS240  | 5 | Zone de travail du lecteur de cassette. (Jusqu'au MSX2+)<br>CS240 = Longueur du signal LOW du bit 0. (25h par défaut)<br>CS240+1 = Longueur du signal HIGH du bit 0. (2Dh par défaut)<br>CS240+2 = Longueur du signal LOW du bit 1. (0Eh par défaut)<br>CS240+3 = Longueur du signal HIGH du bit 1. (16h par défaut)<br>CS240+3 = Longueur du signal d'entête calculé par<br>$\text{HEADLEN} * 4 / 256$ (HEADLEN = 2000 par défaut) |
| 0F406h | LOW    | 2 | Paramètres pour le lecteur de cassette. (Jusqu'au MSX2+)<br>LOW = Longueur du signal LOW qui représente le bit 0 à la vitesse de transmission actuelle. (53h par défaut)<br>LOW+1 = Longueur du signal HIGH qui représente le bit 0 à la vitesse de transmission actuelle. (5Ch par défaut)                                                                                                                                         |
| 0F408h | HIGH   | 2 | Paramètres pour le lecteur de cassette. (Jusqu'au MSX2+)<br>HIGH = Longueur du signal LOW qui représente le bit 1 à la vitesse de transmission actuelle. (26h par défaut)<br>HIGH+1 = Longueur du signal HIGH qui représente le bit 1 à la vitesse de transmission actuelle. (2Dh par défaut)                                                                                                                                       |
| 0F40Ah | HEADER | 1 | Paramètres pour le lecteur de cassette. (Jusqu'au MSX2+)<br>Longueur du signal d'entête actuel calculé par $\text{HEADLEN} * 2 / 256$ ou $\text{HEADLEN} * 4 / 256$ (HEADLEN = 2000 par défaut)                                                                                                                                                                                                                                     |
| 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. (« : » suivi de 4 zéro par défaut.)                                                                                                                                                                                                                                                                                                                                      |
| 0F414h | ERRFLG | 1 | Numéro de la dernière erreur en Basic.                                                                                                                                                                                                                                                                                                                                                                                              |
| 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.<br>0 = Imprimante MSX (les Hiragana sont convertis en Katakana sur les MSX japonais.) ;<br>Autre valeur = Pas de conversion Hiragana vers Katakana.                           |
| 0F418h | RAWPRT  | 1   | Indicateur d'impression brute.<br>0 = Conversion des tabulations en espaces ;<br>Autre valeur = Imprimer tel quel.                                                                                             |
| 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.                                                                                                                                                                       |
| 0F41Eh | KBFMIN  | 1   | Utilisé lorsqu'une erreur de déclaration directe survient.                                                                                                                                                     |
| 0F41Fh | KBUF    | 318 | « Crunch Buffer » utilisé par le Basic.                                                                                                                                                                        |
| 0F55Dh | BUFMIN  | 1   | Petit cache intermédiaire.                                                                                                                                                                                     |
| 0F55Eh | BUF     | 258 | Cache du mode direct, de plusieurs instructions d'un programme Basic en cours d'exécution et des routines du BIOS : PINLIN, INLIN et QINLIN. 0F562h (FNPTR) = Paramètres de l'instruction traitée.             |
| 0F660h | ENDBUF  | 1   | Indicateur de dépassement du cache BUF.                                                                                                                                                                        |
| 0F661h | TTYPOS  | 1   | Dernière position du curseur sur la ligne actuelle de texte.                                                                                                                                                   |
| 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 = Chaîne de caractères ; 4 = Simple précision ; 8 = Double précision. (voir <a href="#">les routines « Math-pack »</a> page 177 pour les formats.) |
| 0F664h | OPRTYP  | 1   | Lorsqu'une instruction utilise un opérateur, le type d'opérateur est stocké momentanément ici.                                                                                                                 |
| "      | DORES   | 1   | Indicateur temporaire de l'interpréteur Basic pour indiquer si il peut comprimer le mot-clé 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. (mise à la norme IEEE 754-1985)                                                                                                                            |
| 0F666h | CONXTXT | 2   | Sauvegarde temporaire du pointeur de texte par la routine CHRGT (00010h en Main-ROM).                                                                                                                          |
| 0F668h | CONSAV  | 1   | Sauvegarde temporaire du code de l'instruction en cours par la routine CHRGT (00010h en Main-ROM)                                                                                                              |
| 0F669h | CONTP   | 1   | Type de constante trouvé par la routine CHRGT (00010h en Main-ROM).                                                                                                                                            |
| 0F66Ah | CONLO   | 8   | Valeur de la constante trouvée par la routine CHRGT (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 réservée pour la pile. Modifié par CLEAR du Basic.                                                                                                                                |



|        |        |    |                                                                                                                                                                                                                                                                 |
|--------|--------|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0F676h | TXTTAB | 2  | Adresse de début du programme Basic à charger.<br>Contient initialement l'adresse indiquée à BOTTOM (0FC48h) +1.                                                                                                                                                |
| 0F678h | TEMPPT | 2  | Pointeur descripteur temporaire du Basic.                                                                                                                                                                                                                       |
| 0F67Ah | TEMPST | 30 | Stockage provisoire pour NUMTEMP de taille de trois UMTMP.                                                                                                                                                                                                      |
| 0F698h | DSCTMP | 3  | Contient la longueur et la position du message « Ok » du Basic.<br>(L'instruction SET PROMPT modifie ces paramètres.)<br>DSCTMP = Nombre de caractère.<br>DSCTMP+1 = Adresse du premier octet de la chaîne de caractères.<br>(Variable appelé aussi « DSCPTR ») |
| 0F69Bh | FRETOP | 2  | Adresse de fin de la table de chaînes de caractères.                                                                                                                                                                                                            |
| 0F69Dh | TEMP3  | 2  | Données temporaires stockées pour le récupérateur de mémoire ou la fonction USR.                                                                                                                                                                                |
| 0F69Fh | TEMP8  | 2  | Données temporaires stockées pour le récupérateur de mémoire.                                                                                                                                                                                                   |
| 0F6A1h | ENDFOR | 2  | Sauvegarde de la position derrière l'instruction FOR 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 adresse pour l'interpréteur Basic. 0 = Ne pas convertir ; 1 = Convertir.                                                                                                                                         |
| 0F6AAh | AUTFLG | 1  | Indicateur pour l'instruction AUTO du Basic.<br>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 du message erreur ou, adresse par défaut de la ligne de programme où le pointeur doit reprendre avec RESUME.                                                                                                                    |
| 0F6B1h | SAVSTK | 2  | Sauvegarde de l'adresse de la pile stockée par les routines de récupération d'erreur afin de restaurer la pile après la gestion d'une erreur.                                                                                                                   |
| 0F6B3h | ERRLIN | 2  | Numéro de la ligne à laquelle la dernière erreur s'est produite.                                                                                                                                                                                                |
| 0F6B5h | DOT    | 2  | Numéro de ligne actuelle affichée par l'instruction LIST du Basic.                                                                                                                                                                                              |
| 0F6B7h | ERRTXT | 2  | Pointeur de texte de l'erreur à afficher.                                                                                                                                                                                                                       |
| 0F6B9h | ONELIN | 2  | Numéro de ligne définie par l'instruction ON ERROR GOTO.                                                                                                                                                                                                        |
| 0F6BBh | ONEFLG | 1  | Indicateur d'erreur pour l'instruction ON ERROR GOTO.<br>1 si traitement d'erreur en cours, autrement 0.                                                                                                                                                        |
| 0F6BCh | TEMP2  | 2  | Mémoire de travail pour le programme d'évaluation de formules.                                                                                                                                                                                                  |
| 0F6BEh | OLDLIN | 2  | Le numéro de ligne y est placé après l'exécution de l'instruction STOP ou END, ou en pressant CTRL+STOP.                                                                                                                                                        |
| 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 de la zone des variables du Basic. L'instruction NEW initialise cette variable avec le contenu de TXTTAB + 2.                                                                                                              |
| 0F6C4h | ARYTAB | 2   | Pointeur sur le début de la zone des variables alphanumériques.                                                                                                                                                                                  |
| 0F6C6h | STREND | 2   | Adresse de la fin de la zone des variables du Basic.                                                                                                                                                                                             |
| 0F6C8h | DATPTR | 2   | Pointeur de la donnée à lire de l'instruction DATA. Modifié par l'instruction RESTORE.                                                                                                                                                           |
| 0F6CAh | DEFTBL | 26  | Table de déclaration des variables définies par les instructions DEFSTR, DEFINT, DEFSNG et DEFDBL du Basic pour chacune des 26 lettres de l'alphabet. 2 = Entier ; 3 = Chaîne ; 4 = Simple précision ; 8 = Double précision (valeur par défaut). |
| 0F6E4h | PRMSTK | 2   | Pointeur vers le bloc défini précédemment. (Afin de nettoyer les déchets.)                                                                                                                                                                       |
| 0F6E6h | PRMLEN | 2   | Nombre d'octets actuellement utilisés dans PARM1.                                                                                                                                                                                                |
| 0F6E8h | PARM1  | 100 | Définition des paramètres.                                                                                                                                                                                                                       |
| 0F74Ch | PRMPRV | 2   | Pointeur vers 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 une recherche dans PARM1 est en cours.                                                                                                                                                                                 |
| 0F7B5h | ARYTA2 | 2   | Point d'arrêt pour une recherche simple.                                                                                                                                                                                                         |
| 0F7B7h | NOFUNS | 1   | Indicateur pour savoir des fonctions ont été définis ou non. 0 si aucune fonction n'est active.                                                                                                                                                  |
| 0F7B8h | TEMP9  | 2   | Pointeur temporaire du récupérateur de mémoire.                                                                                                                                                                                                  |
| 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  | Cache utilisé par les routines mathématiques.                                                                                                                                                                                                    |
| 0F7F0h | DECTMP | 2   | Mémoire de travail utilisé pour transformer un nombre entier décimal en nombre à virgule flottante.                                                                                                                                              |
| 0F7F2h | DECTM2 | 2   | Mémoire de travail pour les divisions.                                                                                                                                                                                                           |
| 0F7F4h | DECCNT | 1   | Mémoire de travail pour les divisions.                                                                                                                                                                                                           |
| 0F7F6h | DAC    | 16  | Accumulateur décimal. Le contenu varie selon que le nombre soit un entier, un simple précision ou un double précision. (voir <a href="#">les routines « Math-pack »</a> page 177 pour les formats.)<br>Note : DAC+2 est aussi appelé FACLO.      |
| 0F806h | HOLD8  | 48  | Sauvegarde temporaire pendant les multiplications décimales.                                                                                                                                                                                     |
| 0F836h | HOLD2  | 8   | Zone de travail pour l'exécution d'opérateurs numériques.                                                                                                                                                                                        |
| 0F83Eh | HOLD   | 8   | Idem que ci-dessus.                                                                                                                                                                                                                              |
| 0F847h | ARG    | 16  | Argument mathématique. Le contenu varie selon que le nombre soit un entier, un simple précision ou un double précision.                                                                                                                          |
| 0F857h | RNDX   | 8   | Dernier nombre aléatoire généré.                                                                                                                                                                                                                 |

|        |        |     |                                                                                                              |
|--------|--------|-----|--------------------------------------------------------------------------------------------------------------|
| 0F85Fh | MAXFIL | 1   | Nombre maximum de fichiers autorisés. Modifié par l'instruction MAXFILES du Disk-Basic.                      |
| 0F860h | FILTAB | 2   | Pointeur sur l'adresse des données du fichier.                                                               |
| 0F862h | NULBUF | 2   | Pointeur du cache 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 pour cassette ou disque.                                                    |
| 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.                                                         |
| 0F87Fh | FNKSTR | 160 | Contient les caractères des touches de fonction à afficher. (16 octets par touche.)                          |
| 0F91Fh | CGPNT  | 3   | Emplacement de la forme des caractères pour initialiser l'écran.<br>CGPNT = Slot ;<br>CGPNT+1 = Adresse.     |
| 0F922h | NAMBAS | 2   | Adresse de la table des caractères ou Bitmap actuelle en VRAM.                                               |
| 0F924h | CGPBAS | 2   | Adresse de la table des formes actuelle en VRAM.                                                             |
| 0F926h | PATBAS | 2   | Adresse de la table des formes de Sprite actuelle en VRAM.                                                   |
| 0F928h | ATRBAS | 2   | Adresse de la table des attributs de Sprite actuelle en VRAM.                                                |
| 0F92Ah | CLOC   | 2   | Adresse en VRAM du curseur graphique (SCREEN 2 à 4) ou, abscisse du curseur graphique (SCREEN 5 à 8)         |
| 0F92Ch | CMASK  | 1   | Masque du curseur graphique (SCREEN 2 à 4) ou, ordonnée du curseur graphique (SCREEN 5 à 12).                |
| 0F92Dh | MINDEL | 2   | Mémoire de travail pour les routines graphiques du Bios.                                                     |
| 0F92Fh | MAXDEL | 2   | Fin de la mémoire de travail pour les routines graphiques du Bios.                                           |
| 0F931h | ASPECT | 2   | Aspect du cercle à tracer. Défini par l'angle de l'instruction CIRCLE du Basic.                              |
| 0F933h | CENCNT | 2   | Compteur utilisé par 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 de contrôle de redondance cyclique 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 | Position la plus à gauche d'une excursion de LH. Utilisé par l'instruction PAINT du Basic.                          |
| 0F94Ah | LOHDIR | 1 | Nouvelle direction de peinture requise par une excursion de LH. Utilisé par l'instruction PAINT du Basic.           |
| 0F94Bh | LOHADR | 2 | Position la plus à gauche d'un LH. Utilisé par l'instruction PAINT du Basic.                                        |
| 0F94Dh | LOHCNT | 2 | Taille d'une excursion de LH. Utilisé par l'instruction PAINT du Basic.                                             |
| 0F94Fh | SKPCNT | 2 | Nombre de saut retournés par la routine de remplissage de l'instruction PAINT du Basic.                             |
| 0F951h | MOVCNT | 2 | Garde le nombre de déplacement retourné par la routine de remplissage de l'instruction PAINT du Basic.              |
| 0F953h | PDIREC | 1 | Direction dans laquelle l'instruction PAINT remplit la forme. 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 table de conversion du Macro langage de l'instruction PLAY ou DRAW du Basic.                       |
| 0F958h | MCLFLG | 1 | Indicateur indiquant si la table de conversion du langage macro actuelle est celle de PLAY (>0) ou de DRAW (0).     |

0F959h    QUETAB    24    Table des fils d'attente de PLAY et de l'interface RS-232C pour MSX1.

QUETAB:

;Queue voix A

db 0                    ; Position de départ

db 0                    ; Indicateur de position

db 0                    ; Indicateur de remplacement

db 7Fh                 ; Taille

dw VOICAQ             ; Adresse (0F975h par défaut)

;Queue voix B

db 0                    ; Position de départ

db 0                    ; Indicateur de position

db 0                    ; Indicateur de remplacement

db 7Fh                 ; Taille

db VOICBQ             ; Adresse (0F9F5h par défaut)

; Queue voix C

db 0                    ; Position de départ

db 0                    ; Indicateur de position

db 0                    ; Indicateur de remplacement

db 7Fh                 ; Taille

dw VOICCQ             ; Adresse (0FA75h par défaut)

; Queue de la RS-232C

db 0                    ; Position de départ

db 0                    ; Indicateur de position

db 0                    ; Indicateur de remplacement

db 0                    ; Taille (3Fh si utilisée)

dw 0000h              ; Adresse (0FAF5h si utilisée)

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). Le standard stipule que **cette zone ne doit plus être utilisée pour l'RS-232C à partir du MSX2.**

0F971h    QUEBAK    4    Table des fils d'attente de caractères de remplacement.

QUEBAK:

db 0    ; Caractère de remplacement (voix A)

db 0    ; Caractère de remplacement (voix B)

db 0    ; Caractère de remplacement (voix C)

db 0    ; Caractère de remplacement (RS-232)

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-232C. (Retiré du standard à partir du MSX2)

0FAF5h    DPPAGE    1    Page affichée. (MSX2~)

0FAF6h    ACPAGE    1    Page active. (MSX2~)

0FAF7h    AVCSAV    1    Sauvegarde du port (F7h) de contrôle AV. (MSX2+~)

0FAF8h    EXBRSA    1    Slot dans lequel se trouve la Sub-ROM. (MSX2~)

0FAF9h    CHRCNT    1    Compteur de caractère dans le cache de conversion Roma-Kana pour le système. (MSX2~)

0FAFAh    ROMA    2    Adresse de sauvegarde le caractères par le système lors d'une conversion Roma-Kana. (MSX2~)

|        |        |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------|--------|---|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0FAFCh | MODE   | 1 | Indicateurs du type de masque à appliquer pour tracer à l'écran en Screen 5 ~ 12, de la taille de la VRAM disponible et pour la conversion Romaji vers Katakana/ Hiragana.<br>bit 0 = 1 si la conversion des Romaji vers Kana est possible.<br>bit 1~2 = Taille de la VRAM.<br>00 pour 16Ko ;<br>01 pour 64Ko ;<br>10 pour 128Ko ;<br>11 pour 192Ko (taille possible mais non standardisé).<br>bit 3 = 1 pour masquer la VRAM au dessus des 16Ko en SCREEN 0~3. (MSX2+~)<br>bit 4 = 0 pour limiter la coordonnée Y à 211/255. (MSX2+~)<br>bit 5 = 0/1 pour tracer en mode RGB/YJK en SCREEN 10/11.<br>bit 6 = 1 si la ROM Kanji est de niveau 2. (MSX2+~)<br>bit 7 = 1 conversion vers Katakana ; 0 conversion vers Hiragana. |
| 0FAFDh | NORUSE | 1 | Utilisé par le pilote de Kanji (Kanji Driver).<br>bit 0~2 = Opérateur logique utilisé pour l'affichage des Kanji.<br>000 pour PSET ;<br>001 pour AND ;<br>010 pour OR ;<br>011 pour XOR ;<br>100 pour NOT.<br>bit 3 = 1 pour couleur 0 transparente.<br>bit 4 = Inutilisé.<br>bit 5 = Désactive<br>bit 6 = Permet le défilement avec SHIFT+Haut/Bas (en mode Kanji).<br>bit 7 = 1 pour interdire le retour au mode texte.                                                                                                                                                                                                                                                                                                     |
| 0FAFEh | XSAVE  | 2 | Sauvegarde de l'abscisse du curseur graphique lue avec la routine GTPAD ou NEWPAD lors de l'utilisation du crayon optique, de la souris, de la tablette graphique, etc. Le bit 15 indique une demande d'interruption du crayon optique. Les bits 14~8 restent à zéro sauf avec le crayon optique car ces bits servent au décalage de la calibration.                                                                                                                                                                                                                                                                                                                                                                          |
| 0FB00h | YSAVE  | 2 | Sauvegarde de l'ordonnée du curseur graphique lue avec la routine GTPAD ou NEWPAD lors de l'utilisation du crayon optique, de la souris, de la tablette graphique, etc. Le bit 15 n'est pas utilisé. Les bits 14~8 restent à zéro sauf avec le crayon optique car ces bits servent au décalage de la calibration.                                                                                                                                                                                                                                                                                                                                                                                                             |
| 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 | Compteur utilisée par l'interface RS-232C.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 0FB04h | RSFCB  | 2 | Adresse du FCB (« File Control Block ») de la RS-232C.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 0FB06h | RSIQLN | 1 | Donnée temporaire utilisée par l'interface RS-232C.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 0FB07h | MEXBIH | 5 | Hook appelé par l'interface RS-232C.<br>MEXBIH : RST 30h (0F7h)<br>MEXBIH+1 : numéro du Slot<br>MEXBIH+2 : adresse<br>MEXBIH+4 : RET (0C9h)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

|        |        |   |                                                                                                                                                             |
|--------|--------|---|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0FB0Ch | OLDSTT | 5 | Hook appelé par l'interface RS-232C.<br>OLDSTT : RST 30h (0F7h)<br>OLDSTT+1 : numéro du Slot<br>OLDSTT+2 : adresse<br>OLDSTT+4 : RET (0C9h)                 |
| 0FB11h | OLDINT | 5 | Hook appelé par l'interface RS-232C.<br>OLDINT : RST 30h (0F7h)<br>OLDINT+1 : numéro du Slot<br>OLDINT+2 : adresse<br>OLDINT+4 : RET (0C9h)                 |
| 0FB16h | DEVNUM | 1 | Utilisée par l'interface RS-232C.                                                                                                                           |
| 0FB17h | DATCNT | 3 | RS-232C.<br>DATCNT : numéro du Slot<br>DATCNT+1 : pointeur (adresse)                                                                                        |
| 0FB1Ah | ERRORS | 1 | Code d'erreur utilisée par l'interface RS-232C.                                                                                                             |
| 0FB1Bh | FLAGS  | 1 | Indicateurs utilisés par l'interface RS-232C.                                                                                                               |
| 0FB1Ch | ESTBLS | 1 | Utilisée par l'interface RS-232C.                                                                                                                           |
| 0FB1Dh | COMMSK | 1 | Utilisée par l'interface RS-232C.                                                                                                                           |
| 0FB1Eh | LSTCOM | 1 | Utilisée par l'interface RS-232C.                                                                                                                           |
| 0FB1Fh | LSTMOD | 1 | Utilisée par l'interface RS-232C.                                                                                                                           |
| 0FB20h | HOKVLD | 1 | Le bit 0 de cet octet indique la présence d'un Bios étendu. 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 sonore des 3 voix du PSG lors d'un stop.                                                                                               |
| 0FB3Bh | MCLLEN | 1 | Longueur de la macro en cours d'analyse.                                                                                                                    |
| 0FB3Ch | MCLPTR | 2 | Adresse de la macro en cours d'analyse.                                                                                                                     |
| 0FB3Eh | QUEUEN | 1 | Numéro de la fil d'attente actuelle.                                                                                                                        |
| 0FB3Fh | MUSICF | 1 | Indicateur d'interruption musicale.                                                                                                                         |
| 0FB40h | PLYCNT | 1 | Nombre de chaines macro dans la file d'attente de PLAY.                                                                                                     |



|        |        |    |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------|--------|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0FB41h | VCBA   | 37 | Zone utilisée par la routine STRTMS pour la voix 0 du PSG.<br>VCBA = Compteur de durée<br>VCBA+2 = Longueur de chaîne<br>VCBA+3 = Adresse de chaîne<br>VCBA+5 = Adresse de données de la pile<br>VCBA+7 = Longueur du paquet de musique<br>VCBA+8 = Paquet de musique<br>VCBA+9 = Octave<br>VCBA+10 = Longueur<br>VCBA+11 = Tempo<br>VCBA+12 = Volume<br>VCBA+13 = Période enveloppe<br>VCBA+15 = Espace pour les données de la pile |
| 0FB66h | VCBB   | 37 | Zone utilisée par la routine STRTMS pour la voix 1 du PSG.<br>VCBB = Compteur de durée<br>VCBB+2 = Longueur de chaîne<br>VCBB+3 = Adresse de chaîne<br>VCBB+5 = Adresse de données de la pile<br>VCBB+7 = Longueur du paquet de musique<br>VCBB+8 = Paquet de musique<br>VCBB+9 = Octave<br>VCBB+10 = Longueur<br>VCBB+11 = Tempo<br>VCBB+12 = Volume<br>VCBB+13 = Période enveloppe<br>VCBB+15 = Espace pour les données de la pile |
| 0FB8Bh | VCBC   | 37 | Zone utilisée par la routine STRTMS pour la voix 2 du PSG.<br>VCBC = Compteur de durée<br>VCBC+2 = Longueur de chaîne<br>VCBC+3 = Adresse de chaîne<br>VCBC+5 = Adresse de données de la pile<br>VCBC+7 = Longueur du paquet de musique<br>VCBC+8 = Paquet de musique<br>VCBC+9 = Octave<br>VCBC+10 = Longueur<br>VCBC+11 = Tempo<br>VCBC+12 = Volume<br>VCBC+13 = Période enveloppe<br>VCBC+15 = Espace pour les données de la pile |
| 0FBB0h | ENSTOP | 1  | Indicateur différent de 0 lorsqu'il est possible de reprendre l'exécution d'un programme Basic.                                                                                                                                                                                                                                                                                                                                      |
| 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 pour chaque ligne de texte affichée à l'écran. (Utilisé par l'interpréteur Basic.)<br>AFh = Ligne qui a été entrée manuellement ou par le système ;<br>Autre valeur = Nombre de ligne restante pour l'affichage à partir de la ligne actuelle (24~1).                                                                                                                                                        |
| 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.<br>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.<br>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 cliquetis de touche 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.<br>SFTKEY (NEWKEY+6) donne l'état de GRAPH, CTRL,SHIFT.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 0FBF0h | KEYBUF | 40 | Cache du clavier par défaut. Contient les codes des dernières touches pressées.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 0FC18h | LINWRK | 40 | Zone de travail pour afficher la ligne des touches de fonction.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 0FC40h | PATWRK | 8  | Zone de travail utilisé par la routine GETPAT (00105h) de la Sub-ROM pour y mettre les données qui forment le caractère spécifié.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 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. Sous Basic, il est possible de créer une zone entre l'adresse indiquée par HIMEM et F1C9h pour y mettre des données ou des routines en langage machine. Pour cela, il faut utiliser l'instruction CLEAR (deuxième paramètre).                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 0FC4Ch | TRPTBL | 78 | Tables pour chacune des instructions suivantes :<br>TRPTBL (longueur 3 × 10 octets) = ON KEY GOSUB<br>TRPTBL+30 (longueur 1 × 3 octet) = ON STOP GOSUB<br>TRPTBL+33 (longueur 1 × 3 octet) = ON SPRITE GOSUB<br>TRPTBL+36 (longueur 5 × 3 octets) = ON STRIG GOSUB<br>TRPTBL+51 (longueur 1 × 3 octet) = ON INTERVAL GOSUB<br>TRPTBL+54 (longueur 8 × 3 octet) = Réservé<br>TRPTBL+72 (longueur 2 × 3 octet) = Réservé pour le système<br>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.<br><i>Note : TRPTBL+30 est aussi nommé REQSTP (0FC6Ah)</i> |
| 0FC9Ah | RTYCNT | 1  | Utilisé par l'interpréteur du Basic pour le contrôle d'interruption.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 0FC9Bh | INTFLG | 1  | Indicateur d'interruption. 4 si STOP est pressé, 3 si CTRL+STOP, autrement 0.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 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 à chaque interruption <a href="#">Vblank</a> utilisé par 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  | Entête 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 | Indicateur pour afficher, ou pas, le curseur pendant l'exécution d'un programme Basic. 0 = Ne pas afficher.                                                                                                                                                                                    |
| 0FCAAh | CSTYLE | 1 | Taille du curseur. 0 = Curseur entier ; Autre valeur = Petit curseur (pour le 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 pour connaître le type de clavier. (MSX Japonais)<br>Contient 0 si l'agencement des touches du clavier est « Kana » sinon, il est « JIS ».                                                                                                                                          |
| 0FCADh |        | 1 | Mettre le bit 0 à 1 pour afficher les Hangul un par un, 0 pour regrouper les Hangul. (MSX coréen seulement)<br>Le bit 2 sert à activer/désactiver (0/1) les instructions CALL HANON et CALL HANOFF sur les MSX2 coréens. (1 par défaut)                                                        |
| 0FCAEh | FLBMEM | 1 | Indicateur de chargement : 0 si un programme Basic se charge.                                                                                                                                                                                                                                  |
| 0FCAFh | SCRMOD | 1 | Mode d'écran actuel.                                                                                                                                                                                                                                                                           |
| 0FCB0h | OLDSCR | 1 | Dernier mode d'écran de texte utilisé. Permet au système de savoir dans quel mode texte il faut revenir après un mode graphique.                                                                                                                                                               |
| 0FCB1h | CASPRV | 1 | Mémoire de travail pour la cassette. (MSX1~MSX2+)<br>Sur les MSX turbo R, le système y sauvegarde le contenu du port A7h à chaque écriture.<br>bit 0 = 1 lorsque la LED Pause est allumée ;<br>bit 1 = 1 lorsque le mode Z80 est actif ;<br>bit 7 = 1 lorsque la LED du mode R800 est 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 pour l'instruction BLOAD. Mis à 52 lorsqu'un fichier binaire a été exécuté. Mis à 53 pendant le chargement de données en VRAM.                                                                                                                                                      |
| 0FCC1h | EXPTBL | 4 | Indicateur de Slot étendu pour chaque Slot primaire. Le bit 7 de ces variables est à 1 lorsque le Slot correspondant est étendu en Slot secondaires. (Voir le chapitre « <a href="#">Utiliser les Slot</a> » page 121 pour plus de précision.)                                                 |

|        |         |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------|---------|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0FCC5h | SLTTBL  | 4   | <p>Contenu des 4 registres qui commandent les Slot secondaires de chacun de Slot primaires.</p> <p>SLTTBL+0 = Contenu du registre des Slot secondaires du Slot primaire 0</p> <p>SLTTBL+1 = Contenu du registre des Slot secondaires du Slot primaire 1</p> <p>SLTTBL+2 = Contenu du registre des Slot secondaires du Slot primaire 2</p> <p>SLTTBL+3 = Contenu du registre des Slot secondaires du Slot primaire 3</p> <p>Format du contenu d'un registre :</p> <p>bit 0 et 1 = Slot secondaire sélectionné sur la plage 0000h~3FFFh</p> <p>bit 2 et 3 = Slot secondaire sélectionné sur la plage 4000h~7FFFh</p> <p>bit 4 et 5 = Slot secondaire sélectionné sur la plage 8000h~BFFFh</p> <p>bit 6 et 7 = Slot secondaire sélectionné sur la plage C000h~FFFFh</p> <p>Le système actualise la variable correspondante au registre à chaque écriture.</p> |
| 0FCC9h | SLTATR  | 64  | Attributs de chaque Slot. (Voir « <a href="#">Développer un programme en cartouche</a> » au chapitre 14 page 564)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 0FD09h | SLTWRK  | 128 | Tableau pour réserver une zone de travail en RAM pour les application en ROM. (Voir « <a href="#">Développer un programme en cartouche</a> » au chapitre 14 page 564 pour la description)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 0FD0Bh |         | 1   | Mettre à 1 pour ne pas afficher de Hangul incomplet tant qu'il n'est pas terminé lors de l'utilisation des instructions INKEY\$ et INPUT\$. (MSX coréens seulement)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 0FD89h | PROCNM  | 16  | Contient le nom de l'instruction étendue en cours d'exécution (exécutée par CALL) ou le nom du périphérique utilisé par OPEN. Le nom se termine par 00h.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 0FFE7h | RG8SAV  | 1   | Contenu du <a href="#">registre 8</a> du VDP. (MSX2~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 0FFE8h | RG9SAV  | 1   | Contenu du <a href="#">registre 9</a> du VDP. (MSX2~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 0FFE9h | RG10SAV | 1   | Contenu du <a href="#">registre 10</a> du VDP. (MSX2~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0FFEAh | RG11SAV | 1   | Contenu du registre 11 du VDP. (MSX2~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0FFEBh | RG12SAV | 1   | Contenu du <a href="#">registre 12</a> du VDP. (MSX2~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0FFEC  | RG13SAV | 1   | Contenu du <a href="#">registre 13</a> du VDP. (MSX2~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0FFEDh | RG14SAV | 1   | Contenu du <a href="#">registre 14</a> du VDP. (MSX2~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0FFEEh | RG15SAV | 1   | Contenu du <a href="#">registre 15</a> du VDP. (MSX2~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0FFEFh | RG16SAV | 1   | Contenu du <a href="#">registre 16</a> du VDP. (MSX2~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0FFF0h | RG17SAV | 1   | Contenu du <a href="#">registre 17</a> du VDP. (MSX2~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0FFF1h | RG18SAV | 1   | Contenu du <a href="#">registre 18</a> du VDP. (MSX2~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0FFF2h | RG19SAV | 1   | Contenu du <a href="#">registre 19</a> du VDP. (MSX2~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0FFF3h | RG20SAV | 1   | Contenu du <a href="#">registre 20</a> du VDP. (MSX2~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0FFF4h | RG21SAV | 1   | Contenu du <a href="#">registre 21</a> du VDP. (MSX2~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0FFF5h | RG22SAV | 1   | Contenu du <a href="#">registre 22</a> du VDP. (MSX2~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0FFF6h | RG23SAV | 1   | Contenu du <a href="#">registre 23</a> du VDP. (MSX2~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

|                  |                  |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                  |                  |                  |                  |
|------------------|------------------|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|------------------|------------------|------------------|
| 0FFF7h           | MINROM           | 1                | Contient le numéro de Slot de la Main-ROM. Préférez utiliser MNROM (0FCC1h) dans vos programme pour savoir si le Slot 0 primaire est étendu, ou non, et MINROM (0FFF7h) pour connaître le Slot de la Main-ROM sauf lorsque l'extension MA-20 est utilisée. (MSX2~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                  |                  |                  |                  |
| 0FFF8h           |                  | 2                | Réservé.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                  |                  |                  |                  |
| 0FFFAh           | RG25SAV          | 1                | Contenu du <a href="#">registre 25</a> du VDP. (MSX2+~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                  |                  |                  |                  |
| 0FFFBh           | RG26SAV          | 1                | Contenu du <a href="#">registre 26</a> du VDP. (MSX2+~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                  |                  |                  |                  |
| 0FFFC            | RG27SAV          | 1                | Contenu du <a href="#">registre 27</a> du VDP. (MSX2+~)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                  |                  |                  |                  |
| 0FFFDh           |                  | 2                | Réservé.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                  |                  |                  |                  |
| 0FFFFh           | SSLTSL           | 1                | Adresse d'accès au registre de sélection des Slot secondaires du Slot primaire dans lequel se trouve la Main-RAM de la plage 0C000h. La valeur lue correspond aux Slot secondaires dont les bits sont inversés afin de pouvoir déterminer si le Slot est étendu ou pas.<br><div style="display: flex; justify-content: space-around; margin-top: 5px;"> <span>bit 7</span><span>bit 6</span><span>bit 5</span><span>bit 4</span><span>bit 3</span><span>bit 2</span><span>bit 1</span><span>bit 0</span> </div> <table border="1" style="margin-top: 5px; width: 100%; text-align: center;"> <tr> <td><math>\overline{SS3}</math></td><td><math>\overline{SS2}</math></td><td><math>\overline{SS1}</math></td><td><math>\overline{SS0}</math></td></tr> </table> | $\overline{SS3}$ | $\overline{SS2}$ | $\overline{SS1}$ | $\overline{SS0}$ |
| $\overline{SS3}$ | $\overline{SS2}$ | $\overline{SS1}$ | $\overline{SS0}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                  |                  |                  |                  |

Chaque Slot primaire étendus ont leur propre registre. Il est donc nécessaire de sélectionner le Slot primaire sur cette page pour pouvoir accéder au registre des Slot secondaires correspondants. 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. Faites attention aussi à la pile qui se trouve probablement sur cette plage.

### 5.3 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 cache du clavier qui renvoie des caractères au mauvais moment. Effacer ce cache est souvent utile :

En Basic :

```
POKE &HF3FA,PEEK(&HF3F8)
POKE &HF3FB,PEEK(&HF3F9)
```

En assembleur :

```
ld    hl,(PUTPNT)      ; PUTPNT = 0F3F8h
ld    (GETPNT),hl      ; GETPNT = 0F3FAh
```

- 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 le contenu d'une touche de fonction, ce ne peut être qu'aux dépens de la suivante.  
Le petit programme Basic suivant illustre cet exemple :

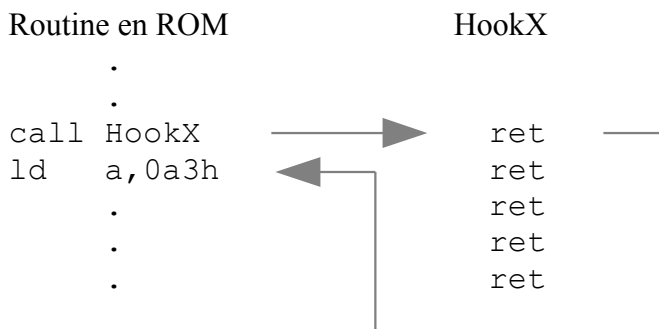
```

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 ..."

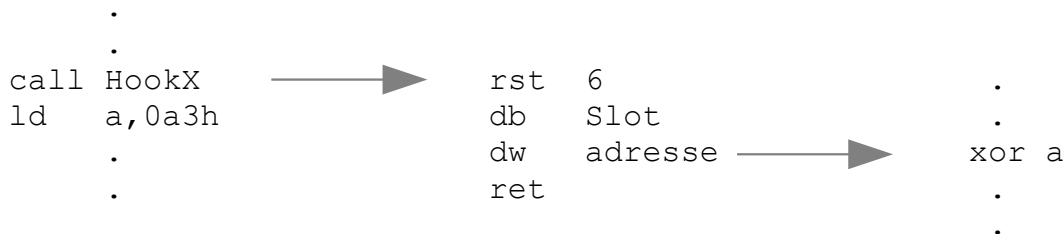
```

## 6 Les Hooks du système

Un Hook est une zone mémoire de cinq octets en mémoire vive (RAM) qui permet d'étendre ou dérouter les routines en mémoire morte, ainsi que les instructions au Basic. De base, les cinq octets contiennent tous 0C9h (RET), comme dans l'exemple suivant :



Il est facile de modifier les 5 octets du Hook afin de détourner une fonction en ROM vers une autre dans un Slot différent, 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 ' 0FF89h: POP HL
30 POKE &HFF8A,&HC3 ' 0FF8Ah: 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.

Note : Dans la pratique avant de modifier un Hook votre programme devra déplacer le contenu du Hook si il est déjà utilisé pour que votre routine y fasse un saut avant ou après exécution.

## 6.1 Liste des Hooks

Voici la liste des Hooks pour tous les MSX. Elle comprend l'adresse, le nom du Hook, le nom de la routine qui l'appelle, ainsi que son utilité :

|         |                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0FD9AH  | <b>H.KEYI</b> (« KEY Interruption »)                                                                                                                                                                                                                                                                                                                                                                     |
| Appel : | Au début de la routine des interruptions KEYINT (Main-ROM à 0038h).                                                                                                                                                                                                                                                                                                                                      |
| Rôle :  | Permet de gérer les interruptions ligne du VDP ou de tester si l'interruption a été provoquée par un dispositif autre que le VDP (La RS-232C, le MSX-MIDI externe, etc.).                                                                                                                                                                                                                                |
| 0FD9Fh  | <b>H.TIMI</b> (« TIMer Interruption »)                                                                                                                                                                                                                                                                                                                                                                   |
| Appel : | Appelé par la routine des interruptions KEYINT (Main-ROM à 0038h).                                                                                                                                                                                                                                                                                                                                       |
| Rôle :  | Permet d'ajouter un Timer, de gérer la collision de Sprite, etc. L'appel à ce Hook se fait juste après la lecture du port de lecture du registre de statut du VDP. Donc lorsqu'elle est appelée, le registre A du CPU contient normalement le contenu du <a href="#">registre de statut 0 du VDP</a> . (Attention à partir du MSX2, le registre de contrôle 15 du VDP peut modifier le registre à lire). |
| 0FDA4h  | <b>H.CHPU</b> (« Character outPUt »)                                                                                                                                                                                                                                                                                                                                                                     |
| Appel : | Au début de la routine CHPUT du Bios (Main-ROM à 00A2h) qui sert à afficher un caractère à l'écran en mode texte.                                                                                                                                                                                                                                                                                        |
| Rôle :  | Permet de sortir le caractère sur un périphérique autre que l'écran.                                                                                                                                                                                                                                                                                                                                     |
| 0FDA9h  | <b>H.DSPC</b> (« DiSPlay Cursor »)                                                                                                                                                                                                                                                                                                                                                                       |
| Appel : | Au début de la routine interne DSPCSR qui sert à afficher le curseur.                                                                                                                                                                                                                                                                                                                                    |
| Rôle :  | Permettre l'accès à d'autres périphériques que l'écran.                                                                                                                                                                                                                                                                                                                                                  |
| 0FDAEh  | <b>H.ERAC</b> (« ERAsE Cursor »)                                                                                                                                                                                                                                                                                                                                                                         |
| Appel : | Au début de la routine interne ERACSR d'effacement du curseur.                                                                                                                                                                                                                                                                                                                                           |
| Rôle :  | Permettre l'accès à d'autres périphériques que l'écran.                                                                                                                                                                                                                                                                                                                                                  |
| 0FDB3h  | <b>H.DSPF</b> (« DiSplay Fonctions »)                                                                                                                                                                                                                                                                                                                                                                    |
| Appel : | Au début de la routine du Bios DSPFNK (Main-ROM à 00CFh) qui sert à afficher le contenu des touches de fonction.                                                                                                                                                                                                                                                                                         |
| Rôle :  | Permettre l'accès à d'autres périphériques que l'écran.                                                                                                                                                                                                                                                                                                                                                  |

|         |                                                                                                                                                                                                                             |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0FDB8h  | <b>H.ERAF</b> (« ERAse Fonctions »)                                                                                                                                                                                         |
| Appel : | Au début de la routine ERAFNK du Bios (Main-ROM à 00CCh) qui sert à effacer le contenu des touches de fonction.                                                                                                             |
| Rôle :  | Permettre l'accès à d'autres périphériques que l'écran.                                                                                                                                                                     |
| 0FDBDh  | <b>H.TOTE</b> (« TO TExt »)                                                                                                                                                                                                 |
| Appel : | Au début de la routine du Bios TOTEXT (Main-ROM à 00D2h) de passage en mode texte.                                                                                                                                          |
| Rôle :  | Permettre l'accès à d'autres périphériques que l'écran.                                                                                                                                                                     |
| 0FDC2h  | <b>H.CHGE</b> (« CHaracter GEt »)                                                                                                                                                                                           |
| Appel : | Au début de la routine du Bios CHGET (Main-ROM à 009Fh) de lecture d'un caractère au clavier.                                                                                                                               |
| Rôle :  | Permettre l'accès à d'autres périphériques d'entrée que le clavier du MSX.                                                                                                                                                  |
| 0FDC7h  | <b>H.INIP</b> (« INItialize Patterns »)                                                                                                                                                                                     |
| Appel : | Au début de la routine interne d'initialisation des caractères INIPAT qui copient la police des caractères vers la VRAM.                                                                                                    |
| Rôle :  | Permet de modifier le jeu de caractères lorsque l'on revient en mode texte.                                                                                                                                                 |
| 0FDCCh  | <b>H.KEYC</b> (« KEY Code »)                                                                                                                                                                                                |
| Appel : | Au début de la routine interne KEYCOD de lecture de clavier.                                                                                                                                                                |
| Rôle :  | Permet d'intercepter la lecture du clavier. Lorsque le Hook est appelé, le registre A contient le numéro de ligne de la matrice du clavier sur les bits 4 à 7 et le numéro du bit de la touche enfoncée sur les bits 0 à 3. |
| 0FDD1h  | <b>H.KYEA</b> (« KeY EAsy »)                                                                                                                                                                                                |
| Appel : | Au début de la routine interne KYEASY de conversion d'un caractère lu au clavier.                                                                                                                                           |
| Rôle :  | Permet de modifier la manière dont une touche est interprétée sous Basic.                                                                                                                                                   |
| 0FDD6h  | <b>H.NMI</b> (« Non-Masquable Interruption »)                                                                                                                                                                               |
| Appel : | Au début de la routine NMI du Bios (Main-ROM à 0066h) d'interruptions non masquables.                                                                                                                                       |
| Rôle :  | Permet le traitement des interruptions non masquables. (Réservé mais jamais utilisé sauf peut-être en option par le Turbo R en mode R800)                                                                                   |
| 0FDDb   | <b>H.PINL</b> (« Program INput Line »)                                                                                                                                                                                      |
| Appel : | Au début de la routine PINLIN du Bios (Main-ROM à 00AEh) qui gère l'entrée d'une ligne de programme au clavier.                                                                                                             |
| Rôle :  | Permet d'utiliser un autre périphérique d'entrée que le clavier ou gérer le mode 80 colonnes.                                                                                                                               |



|         |                                                                                                                                                                                                    |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0FDE0h  | <b>H.QINL</b> (« Question mark and INput LIne »)                                                                                                                                                   |
| Appel : | Au début de la routine QINLIN du Bios (Main-ROM à 00B4h) de l'entrée au clavier avec affichage d'un point d'interrogation.                                                                         |
| Rôle :  | Permet d'utiliser le mode 80 colonnes, ou un autre périphérique d'entrée que le clavier.                                                                                                           |
| 0FDE5h  | <b>H.INLI</b> (« INput LIne »)                                                                                                                                                                     |
| Appel : | Au début de la routine INLIN du Bios (Main-ROM à 000B1h) d'entrée au clavier.                                                                                                                      |
| Rôle :  | Permet d'utiliser les 80 colonnes, ou un autre périphérique d'entrée que le clavier.                                                                                                               |
| 0FDEAh  | <b>H.ONGO</b> (« ON GOto procedure »)                                                                                                                                                              |
| Appel : | Au début de la routine interne ONGOTP, pour les instructions Basic de type ON GOTO, ON GOSUB.                                                                                                      |
| Rôle :  | Permet de détourner ou modifier ce type d'instructions.                                                                                                                                            |
| 0FDEFh  | <b>H.DSKO</b> (« DiSK sector Output »)                                                                                                                                                             |
| Appel : | Au début de l'instruction DSKO\$ du Basic.                                                                                                                                                         |
| Rôle :  | Ajouter l'instruction sur un système avec disque ou la détourner/modifier lorsqu'elle est présente.                                                                                                |
| Note :  | La Disk-ROM utilise ce Hook pour écrire le secteur d'un disque.                                                                                                                                    |
| 0FDF4h  | <b>H.SETS</b> (« SETs »)                                                                                                                                                                           |
| Appel : | Au début de l'instruction SET du Basic.                                                                                                                                                            |
| Rôle :  | Détourner ou étendre l'instruction SET.                                                                                                                                                            |
| Note :  | Sur MSX1, l'instruction SET n'a pas d'autre effet que d'appeler ce hook et renvoyer une erreur. Sur MSX2 ou plus récent, les instructions SET SCREEN, SET ADJUST, SET TIME, etc appellent ce hook. |
| 0FDF9h  | <b>H.NAME</b> (« NAME »)                                                                                                                                                                           |
| Appel : | Au début de l'instruction NAME du Basic.                                                                                                                                                           |
| Rôle :  | Ajouter l'instruction sur un système avec disque ou la détourner/modifier lorsqu'elle est présente.                                                                                                |
| Note :  | La Disk-ROM utilise ce Hook pour renommer des fichiers.                                                                                                                                            |
| 0FDFEh  | <b>H.KILL</b> (« KILL »)                                                                                                                                                                           |
| Appel : | Au début de l'instruction KILL du Basic.                                                                                                                                                           |
| Rôle :  | Ajouter l'instruction sur un système avec disque ou la détourner/modifier lorsqu'elle est présente.                                                                                                |
| Note :  | La Disk-ROM utilise ce Hook pour effacer des fichiers.                                                                                                                                             |

|         |                                                                                                                                                                                         |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0FE03h  | <b>H.IPL</b>                                                                                                                                                                            |
| Appel : | Au début de l'instruction IPL du Basic.                                                                                                                                                 |
| Rôle :  | L'instruction IPL du Basic ne fait qu'appeler ce Hook car il s'agit d'une instruction réservée pour une utilisation future mais jamais utilisée.                                        |
| Note :  | Un programmeur peut créer sa propre instruction. Voir le chapitre 14.7 « <a href="#">Ajouter une instruction au Basic avec CMD ou IPL</a> » page 548 pour plus de précision.            |
| 0FE08h  | <b>H.COPY</b> (« COPY »)                                                                                                                                                                |
| Appel : | Au début de l'instruction COPY du Basic.                                                                                                                                                |
| Rôle :  | Ajouter l'instruction sur un système avec disque. Permet aussi d'étendre l'instruction. Par exemple, pour d'autres mode d'écran. Peut servir à empêcher la copie de fichier sur disque. |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                                                                |
| 0FE0Dh  | <b>H.CMD</b> (« CoMmand »)                                                                                                                                                              |
| Appel : | Au début de l'instruction CMD du Basic.                                                                                                                                                 |
| Rôle :  | L'instruction CMD du Basic ne fait qu'appeler ce Hook car il s'agit d'une instruction réservée pour une utilisation future mais jamais utilisée.                                        |
| Note :  | Un programmeur peut créer sa propre instruction. Voir le chapitre 14.7 « <a href="#">Ajouter une instruction au Basic avec CMD ou IPL</a> » page 548 pour plus de précision.            |
| 0FE12h  | <b>H.DSKF</b> (« DiSK Free »)                                                                                                                                                           |
| Appel : | Au début de la routine DSKF, utilisée l'instruction DSKF\$ du Basic.                                                                                                                    |
| Rôle :  | Ajouter l'instruction sur un système avec disque. Empêcher de connaître la mémoire disponible sur les disques. Utilisé par défaut par la Disk-ROM pour obtenir un accès aux disques.    |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                                                                |
| 0FE17h  | <b>H.DSKI</b> (« DiSK Input»)                                                                                                                                                           |
| Appel : | Au début de la routine DSKI , utilisée par l'instruction DSKI\$ du Basic.                                                                                                               |
| Rôle :  | Ajouter l'instruction sur un système avec disque ou la détourner/modifier lorsqu'elle est présente.                                                                                     |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                                                                |
| 0FE1Ch  | <b>H.ATTR</b> (« ATTRibute »)                                                                                                                                                           |
| Appel : | Au début de l'instruction ATTR\$ du Basic.                                                                                                                                              |
| Rôle :  | Instruction réservée.                                                                                                                                                                   |
| Note :  | Cette instruction ne fait qu'appeler ce Hook car il s'agit d'une instruction réservée pour une utilisation future mais jamais utilisée.                                                 |

|         |                                                                                                     |
|---------|-----------------------------------------------------------------------------------------------------|
| 0FE21h  | <b>H.LSET</b> (« Left SET »)                                                                        |
| Appel : | Au début de la routine LSET, utilisée par l'instruction LSET du Basic.                              |
| Rôle :  | Ajouter l'instruction sur un système avec disque ou la détourner/modifier lorsqu'elle est présente. |
| Note :  | Utilisé par la Disk-ROM.                                                                            |
| 0FE26h  | <b>H.RSET</b> (« Right SET »)                                                                       |
| Appel : | Au début de la routine RSET, utilisée par l'instruction RSET du Basic.                              |
| Rôle :  | Ajouter l'instruction sur un système avec disque ou la détourner/modifier lorsqu'elle est présente. |
| Note :  | Utilisé par la Disk-ROM.                                                                            |
| 0FE2Bh  | <b>H.FIEL</b> (« FIELd »)                                                                           |
| Appel : | Au début de la routine FIELD, utilisée par l'instruction FIELD du Basic.                            |
| Rôle :  | Ajouter l'instruction sur un système avec disque ou la détourner/modifier lorsqu'elle est présente. |
| Note :  | Utilisé par la Disk-ROM.                                                                            |
| 0FE30h  | <b>H.MKI\$</b> (« MaKe Integer »)                                                                   |
| Appel : | Au début de la routine MKI, utilisée par l'instruction MKI\$ du Basic.                              |
| Rôle :  | Ajouter l'instruction sur un système avec disque ou la détourner/modifier lorsqu'elle est présente. |
| Note :  | Utilisé par la Disk-ROM.                                                                            |
| 0FE35h  | <b>H.MKS\$</b> (« MaKe Simple precision »)                                                          |
| Appel : | Au début de la routine MKS, utilisée par l'instruction MKS\$ du Basic.                              |
| Rôle :  | Ajouter l'instruction sur un système avec disque ou la détourner/modifier lorsqu'elle est présente. |
| Note :  | Utilisé par la Disk-ROM.                                                                            |
| 0FE3Ah  | <b>H.MKD\$</b> (« MaKe Double precision »)                                                          |
| Appel : | Au début de la routine MKD, utilisée par l'instruction MKD\$ du Basic.                              |
| Rôle :  | Ajouter l'instruction sur un système avec disque ou la détourner/modifier lorsqu'elle est présente. |
| Note :  | Utilisé par la Disk-ROM.                                                                            |

|         |                                                                                                     |
|---------|-----------------------------------------------------------------------------------------------------|
| 0FE3Fh  | <b>H.CVI</b> (« ConVert Integer »)                                                                  |
| Appel : | Au début de la routine CVI, utilisée par l'instruction CVI du Basic.                                |
| Rôle :  | Ajouter l'instruction sur un système avec disque ou la détourner/modifier lorsqu'elle est présente. |
| Note :  | Utilisé par la Disk-ROM.                                                                            |
| 0FE44h  | <b>H.CVS</b> (« ConVert Single precision »)                                                         |
| Appel : | Au début de la routine interne CVS, utilisée par l'instruction CVS du Basic.                        |
| Rôle :  | Ajouter l'instruction sur un système avec disque ou la détourner/modifier lorsqu'elle est présente. |
| Note :  | Utilisé par la Disk-ROM.                                                                            |
| 0FE49h  | <b>H.CVD</b> (« ConVert Double precision »)                                                         |
| Appel : | Au début de la routine interne CVD, utilisé par l'instruction CVD du Basic.                         |
| Rôle :  | Ajouter l'instruction sur un système avec disque ou la détourner/modifier lorsqu'elle est présente. |
| Note :  | Utilisé par la Disk-ROM.                                                                            |
| 0FE4Eh  | <b>H.GETP</b> (« GET file Pointer »)                                                                |
| Appel : | Au début de la routine interne GETPTR, pour connaître le pointeur du fichier ouvert.                |
| Rôle :  | Traiter la routine.                                                                                 |
| Note :  | Utilisé par la Disk-ROM.                                                                            |
| 0FE53h  | <b>H.SETF</b> (« SET File »)                                                                        |
| Appel : | Au début de la routine interne SETFIL, positionner le pointeur du fichier ouvert au préalable.      |
| Rôle :  | Traiter la routine.                                                                                 |
| Note :  | Utilisé par la Disk-ROM.                                                                            |
| 0FE58h  | <b>H.NOFO</b> (« NO FOr close »)                                                                    |
| Appel : | Au début de la routine interne NOFOR, utilisée par l'instruction OPEN du Basic.                     |
| Rôle :  | Traiter la routine.                                                                                 |
| Note :  | Utilisé par la Disk-ROM.                                                                            |

|         |                                                                                                                                                    |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| 0FE5Dh  | <b>H.NULL</b> (« NULL file Open »)                                                                                                                 |
| Appel : | Au début de la routine NULOPN de la Disk-ROM, utilisée par les instructions LOAD, KILL, MERGE, etc du Basic lors de l'ouverture d'un fichier vide. |
| Rôle :  | Traiter la routine.                                                                                                                                |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                           |
| 0FE62h  | <b>H.NTFL</b> (« NoT FiLe number 0 »)                                                                                                              |
| Appel : | Au début de la routine NTFL0 de la Disk-ROM, utilisée par l'instruction CLOSE du Basic, appelé lorsque le numéro de fichiers est différent de 0.   |
| Rôle :  | Traiter la routine.                                                                                                                                |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                           |
| 0FE67h  | <b>H.MERG</b> (« MERGE program files »)                                                                                                            |
| Appel : | Au début de l'instruction MERGE du Basic.                                                                                                          |
| Rôle :  | Ajouter l'instruction sur un système avec disque ou la détourner/modifier lorsqu'elle est présente.                                                |
| 0FE6Ch  | <b>H.SAVE</b> (« SAVE program files »)                                                                                                             |
| Appel : | Au début de l'instruction SAVE du Basic.                                                                                                           |
| Rôle :  | Ajouter l'instruction sur un système avec disque ou la détourner/modifier lorsqu'elle est présente.                                                |
| 0FE71h  | <b>H.BINS</b> (« BINary Save »)                                                                                                                    |
| Appel : | Au début de la routine interne BINSAV, sauvegarde d'une zone mémoire en binaire, utilisée par l'instruction SAVE du Basic.                         |
| Rôle :  | Traiter la routine.                                                                                                                                |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                           |
| 0FE76h  | <b>H.BINL</b> (« BINary Load »)                                                                                                                    |
| Appel : | Au début de la routine interne BINLOD, chargement d'une zone mémoire en binaire, utilisée par l'instruction LOAD du Basic.                         |
| Rôle :  | Traiter la routine.                                                                                                                                |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                           |
| 0FE7Bh  | <b>H.FILE</b> (« FILEs »)                                                                                                                          |
| Appel : | Au début de l'instruction FILES du Basic.                                                                                                          |
| Rôle :  | Traiter l'instruction.                                                                                                                             |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                           |

|         |                                                                                                                                          |
|---------|------------------------------------------------------------------------------------------------------------------------------------------|
| 0FE80h  | <b>H.DGET</b> (« Disk GET »)                                                                                                             |
| Appel : | Au début de la routine interne DGET, utilisée par l'instruction GET et PUT du Basic.                                                     |
| Rôle :  | Traiter l'instruction.                                                                                                                   |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                 |
| 0FE85h  | <b>H.FILO</b> (« FILE Output »)                                                                                                          |
| Appel : | Au début de la routine interne FILOU1, sortie dans un fichier.                                                                           |
| Rôle :  | Traiter la routine.                                                                                                                      |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                 |
| 0FE8Ah  | <b>H.INDS</b> (« INput DiSk character »)                                                                                                 |
| Appel : | Au début de la routine interne INDSKC d'entrée d'attribut du disque.                                                                     |
| Rôle :  | Traiter la routine.                                                                                                                      |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                 |
| 0FE8Fh  | <b>H.RSLF</b>                                                                                                                            |
| Appel : | Au début de la routine interne qui sélectionne lecteur précédant comme lecteur actuel.                                                   |
| Rôle :  | Traiter la routine.                                                                                                                      |
| 0FE94h  | <b>H.SAVD</b> (« SAVe Disk »)                                                                                                            |
| Appel : | Au début de la routine interne de sauvegarde du lecteur actuel, utilisée par les instructions LOF, LOC, EOF, FPOS, etc.                  |
| Rôle :  | Ajouter ces instructions sur un système avec disque. Permet aussi de détourner ou modifier ces instructions lorsqu'elles sont présentes. |
| 0FE99h  | <b>H.LOC</b> (« LOCation »)                                                                                                              |
| Appel : | Au début de la routine interne LOC, utilisée par l'instruction LOC du Basic.                                                             |
| Rôle :  | Traiter la routine.                                                                                                                      |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                 |
| 0FE9Eh  | <b>H.LOF</b> (« Length Of File »)                                                                                                        |
| Appel : | Au début de la routine interne LOF, utilisée par l'instruction LOF du Basic.                                                             |
| Rôle :  | Détourner l'instruction.                                                                                                                 |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                 |

|         |                                                                                                                                         |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0FEA3h  | <b>H.EOF</b> (« End Of File »)                                                                                                          |
| Appel : | Au début de la routine interne EOF, utilisée lors de l'instruction EOF du Basic.                                                        |
| Rôle :  | Détourner l'instruction. Utilisé par défaut par la Disk-ROM pour obtenir un accès aux disques.                                          |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                |
| 0FEA8h  | <b>H.FPOS</b> (« File POSition »)                                                                                                       |
| Appel : | Au début de l'instruction FPOS.                                                                                                         |
| Rôle :  | Instruction réservée.                                                                                                                   |
| Note :  | Cette instruction ne fait qu'appeler ce Hook car il s'agit d'une instruction réservée pour une utilisation future mais jamais utilisée. |
| 0FEADh  | <b>H.BAKU</b> (« BAcK Up »)                                                                                                             |
| Appel : | Au début de la routine interne BAKUPT.                                                                                                  |
| Rôle :  | Traiter la routine.                                                                                                                     |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                |
| 0FEB2h  | <b>H.PARD</b> (« PARse Device name »)                                                                                                   |
| Appel : | Au début de la routine interne PARDEV qui analyse le nom du périphérique.                                                               |
| Rôle :  | Permet de changer le nom du disque logique («device name »).                                                                            |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                |
| 0FEB7h  | <b>H.NODE</b> (« NO DEvice name »)                                                                                                      |
| Appel : | Au début de la routine interne NODEVN qui est appelé lorsque aucun nom n'a été trouvé dans la table des noms de périphérique.           |
| Rôle :  | Permet de mettre un nom au disque dont le nom a été omis.                                                                               |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                |
| 0FEBCh  | <b>H.POSD</b> (« POSsible Disk »)                                                                                                       |
| Appel : | Au début de la routine interne POSDSK.                                                                                                  |
| Rôle :  | Permet de raccorder un disque.                                                                                                          |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                |
| 0FEC1h  | <b>H.DEVN</b> (« DEvice NAME »)                                                                                                         |
| Appel : | Au début de la routine interne DEVNAM pour traiter le nom de périphérique.                                                              |
| Rôle :  | Permet d'étendre un nom de disque logique.                                                                                              |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                |

|         |                                                                                                                                                                                                                                                                                                                          |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0FEC6h  | <b>H.GEND</b> (« GENeral device Dispatcher »)                                                                                                                                                                                                                                                                            |
| Appel : | Au début de la routine interne GENDSP pour assign le nom de périphérique.                                                                                                                                                                                                                                                |
| Rôle :  | Permet d'étendre un nom du disque logique.                                                                                                                                                                                                                                                                               |
| Note :  | Utilisé par la Disk-ROM.                                                                                                                                                                                                                                                                                                 |
| 0FECBh  | <b>H.RUNC</b> (« RUN Clear »)                                                                                                                                                                                                                                                                                            |
| Appel : | Au début de la routine interne RUNC, utilisée par les instructions NEW et RUN du Basic.                                                                                                                                                                                                                                  |
| Rôle :  | Permet le détournement des instructions NEW et RUN.                                                                                                                                                                                                                                                                      |
| 0FED0h  | <b>H.CLEA</b> (« CLEAR clear »)                                                                                                                                                                                                                                                                                          |
| Appel : | Au début de la routine interne CLEARC qui initialise les variables, utilisée par l'instruction CLEAR du Basic.                                                                                                                                                                                                           |
| Rôle :  | Permet d'éviter l'effacement des variables Basic, par exemple.                                                                                                                                                                                                                                                           |
| 0FED5h  | <b>H.LOPD</b> (« LOP and set Default »)                                                                                                                                                                                                                                                                                  |
| Appel : | Au début de la routine interne LOPDFT, initialisation de la table des variables.                                                                                                                                                                                                                                         |
| Rôle :  | Traiter la routine.                                                                                                                                                                                                                                                                                                      |
| 0FEDAh  | <b>H.STKE</b> (« STAcK Error »)                                                                                                                                                                                                                                                                                          |
| Appel : | Au début de la routine interne STKERR, erreur de pile.                                                                                                                                                                                                                                                                   |
| Rôle :  | Traiter la routine.                                                                                                                                                                                                                                                                                                      |
| Note :  | Ce Hook est appelé après la recherche des ROM exécutables dans chaque Slot lors de l'initialisation du MSX, juste avant que le système démarre l'environnement Basic. Ce Hook peut donc vous permettre de ré-exécuter automatiquement votre ROM après que les disques soient installés en y plaçant la routine suivante. |
|         | <pre> H.STKE:     rst    030h        ; Inter-slot call     db     1           ; Numéro du Slot de la ROM     dw     ROM_Exe     ; Address to execute the ROM </pre>                                                                                                                                                      |
|         | Remettez ce Hook dans son état initial une fois la ROM exécutée grâce à ce Hook.                                                                                                                                                                                                                                         |
| 0FEDFh  | <b>H.ISFL</b> (« IS FiLe I/O »)                                                                                                                                                                                                                                                                                          |
| Appel : | Au début de la routine interne ISFLIO qui teste si le fichier à écrire ou à lire.                                                                                                                                                                                                                                        |
| Rôle :  | Traiter la routine.                                                                                                                                                                                                                                                                                                      |
| 0FEE4h  | <b>H.OUTD</b> (« OUT Do »)                                                                                                                                                                                                                                                                                               |
| Appel : | Au début de la routine du Bios OUTDO (Main-ROM à 0018h), sortie d'un caractère sur écran ou imprimante.                                                                                                                                                                                                                  |
| Rôle :  | Faire une routine pour un autre périphérique.                                                                                                                                                                                                                                                                            |



|         |                                                                                                                     |
|---------|---------------------------------------------------------------------------------------------------------------------|
| 0FEE9h  | <b>H.CRDO</b> (« CaRriage left DO »)                                                                                |
| Appel : | Au début de la routine interne CRDO qui émet un code CR (« Carriage Return », 0Dh) et d'un LF (« Line Feed », 0Ah). |
| Rôle :  | Permet, par exemple, d'adapter l'impression des lignes sur une imprimante possédant un line-feed automatique.       |
| 0FEEeh  | <b>H.DSKC</b> (« DiSK Character input »)                                                                            |
| Appel : | Au début de la routine interne DSKCHI, pour l'entrée de l'attribut d'un disque.                                     |
| Rôle :  | Traiter la routine.                                                                                                 |
| 0FEF3h  | <b>H.DOGR</b>                                                                                                       |
| Appel : | Au début de la routine interne DOGRPH, utilisée par les instructions du Basic de traçage graphiques .               |
| Rôle :  | Traiter la routine.                                                                                                 |
| 0FEF8h  | <b>H.PRGE</b> (« PRoGram End »)                                                                                     |
| Appel : | Au début de la routine PRGEND, appelé 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  | <b>H.ERRP</b> (« ERRor Print »)                                                                                     |
| Appel : | Au début de la routine ERRPRT pour l'affichage du message d'erreur sous Basic.                                      |
| Rôle :  | Permet de changer ou ajouter un message d'erreur.                                                                   |
| Note :  | Utilisé par la Disk-ROM.                                                                                            |
| 0FF02h  | <b>H.ERRF</b> (« ERRor Fin »)                                                                                       |
| Appel : | À la fin de la routine qui affiche le message d'erreur sous Basic.                                                  |
| Rôle :  | Permet une action supplémentaire après l'affichage de l'erreur.                                                     |
| 0FF07h  | <b>H.READ</b> (« READy »)                                                                                           |
| Appel : | Au début de la routine READY qui affiche le message « Ok » (ou celui défini par l'utilisateur sur MSX2~).           |
| Rôle :  | Permet, par exemple, de provoquer d'un bip sonore à chaque affichage de « OK ».                                     |
| 0FF0Ch  | <b>H.MAIN</b> (« MAIN entry »)                                                                                      |
| Appel : | Au début de la routine MAIN, utilisée à chaque accès à l'interpréteur Basic.                                        |
| Rôle :  | Permet de programmer une interruption par exemple.                                                                  |

|         |                                                                                                                           |
|---------|---------------------------------------------------------------------------------------------------------------------------|
| 0FF11h  | <b>H.DIRD</b> (« DIRection Do »)                                                                                          |
| Appel : | Au début de la routine DIRDO qui est appelée lors de l'exécution d'instructions en mode direct.                           |
| Rôle :  | Permet toutes les manipulations sur le mode direct du Basic.                                                              |
| 0FF16h  | <b>H.FINI</b> (« Function INterpretation Initialize »)                                                                    |
| Appel : | Au début de la routine FININT, utilisée lors de la traduction d'une instruction Basic par l'interpréteur.                 |
| Rôle :  | Permet de détourner le traitement des instructions Basic.                                                                 |
| 0FF1Bh  | <b>H.FINE</b> (« Function INterpretation End »)                                                                           |
| Appel : | Au début de la routine FINEND, utilisée à la fin de l'interprétation d'une instruction Basic.                             |
| Rôle :  | Ajouter une routine.                                                                                                      |
| 0FF20h  | <b>H.CRUN</b>                                                                                                             |
| Appel : | Au début de la routine CRUNCH, transformation d'une ligne Basic en mots-clefs.                                            |
| Rôle :  | Traiter la routine.                                                                                                       |
| 0FF25h  | <b>H.CRUS</b>                                                                                                             |
| Appel : | Au début de la routine CRUSH qui recherche un mot-clef dans la liste alphabétique en Rom.                                 |
| Rôle :  | Traiter la routine.                                                                                                       |
| 0FF2Ah  | <b>H.ISRE</b> (« IS token REserved »)                                                                                     |
| Appel : | Au début de la routine ISRESV lorsqu'un mot-clef est trouvé par la routine CRUSH.                                         |
| Rôle :  | Traiter la routine.                                                                                                       |
| 0FF2Fh  | <b>H.NTFN</b>                                                                                                             |
| Appel : | Au début de la routine NTFN2 lorsqu'un mot-clef est suivi par un numéro de ligne.                                         |
| Rôle :  | Traiter la routine.                                                                                                       |
| 0FF34h  | <b>H.NOTR</b> (« token NOT Reserved »)                                                                                    |
| Appel : | Au début de la routine NOTRSV lorsque la suite de caractères examinée par la routine CRUNCH ne constitue pas un mot-clef. |
| Rôle :  | Traiter la routine.                                                                                                       |

|         |                                                                                                                   |
|---------|-------------------------------------------------------------------------------------------------------------------|
| 0FF39h  | <b>H.SNGF</b>                                                                                                     |
| Appel : | Au début de la routine SNGFOR, accès aux fonctions mathématiques BCD                                              |
| Rôle :  | Permet d'installer de nouvelles routines mathématiques.                                                           |
| 0FF3Eh  | <b>H.NEWS</b> (« NEW Satement »)                                                                                  |
| Appel : | Au début de la routine NEWSTT, passage à une nouvelle instruction Basic.                                          |
| Rôle :  | Traiter la routine.                                                                                               |
| 0FF43h  | <b>H.GONE</b>                                                                                                     |
| Appel : | Au début de la routine GONE2, utilisée par les instructions de saut (GOTO, THEN, ...)                             |
| Rôle :  | Traiter la routine.                                                                                               |
| 0FF48h  | <b>H.CHRG</b> (« CHaracter GeTteR »)                                                                              |
| Appel : | Au début de la routine CHRGTTR (00010h) de récupération d'un caractère.                                           |
| Rôle :  | Traiter la routine.                                                                                               |
| 0FF4Dh  | <b>H.RETU</b> (« RETURn from sub-routine »)                                                                       |
| Appel : | Au début de l'instruction RETURN du Basic.                                                                        |
| Rôle :  | Changer le traitement de l'instruction.                                                                           |
| 0FF52h  | <b>H.PRTF</b>                                                                                                     |
| Appel : | Au début de l'instruction Basic PRINT                                                                             |
| Rôle :  | Changer le traitement de l'instruction.                                                                           |
| 0FF57h  | <b>H.COMP</b>                                                                                                     |
| Appel : | Au début de la routine COMPRT, boucle interne à l'instruction Basic PRINT                                         |
| Rôle :  | Changer le traitement de l'instruction.                                                                           |
| 0FF5Ch  | <b>H.FINP</b> (« FINal Print »)                                                                                   |
| Appel : | A la fin de l'affichage d'un texte sous Basic.                                                                    |
| Rôle :  | Ajouter une routine après l'affichage d'un texte.                                                                 |
| 0FF61h  | <b>H.TRMN</b>                                                                                                     |
| Appel : | Au début de la routine TRMNOK, traitement d'une mauvaise donnée entrée pour les instructions Basic READ et INPUT. |
| Rôle :  | Traiter l'erreur.                                                                                                 |

|         |                                                                                                                                                                                                                                        |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0FF66h  | <b>H.FRME</b> (« FoRMula EvaLuator »)                                                                                                                                                                                                  |
| Appel : | Au début de la routine interne FRMEVL (04C64h) de reconnaissance d'une expression dans le texte d'un programme Basic.                                                                                                                  |
| Rôle :  | Permet d'ajouter de nouvelles fonctions mathématiques au Basic                                                                                                                                                                         |
| Note :  | Voici la description de la routine FRMEVL.<br>Entrée :        HL = Pointeur du texte.<br>Sortie :        HL = Pointeur vers l'expression trouvée.<br>VALTYP (F663h) = Type de valeur de l'expression.<br>DAC (F7F6h) = Valeur trouvée. |
| 0FF6Bh  | <b>H.NTPL</b>                                                                                                                                                                                                                          |
| Appel : | Par la routine utilisée lors de FRMEVL (04C64h).                                                                                                                                                                                       |
| Rôle :  | Permet d'ajouter de nouvelles fonctions mathématiques au Basic.                                                                                                                                                                        |
| 0FF70h  | <b>H.EVAL</b> (« EVAЛуate statement »)                                                                                                                                                                                                 |
| Appel : | Au début de la routine EVAL (04DC7h), évaluation d'une expression.                                                                                                                                                                     |
| Rôle :  | Permet d'ajouter de nouvelles fonctions mathématiques au Basic                                                                                                                                                                         |
| 0FF75h  | <b>H.OKNO</b> (« OK or NO »), ou <b>H.MDIN</b> (« MiDi IN »)                                                                                                                                                                           |
| Appel : | Au début de la routine de calcul de fonction transcendante (MSX1/2/2+), ou de la routine d'interruption de l'entrée du MSX-MIDI interne (MSX turbo R).                                                                                 |
| Rôle :  | Traiter la routine de calcul de fonctions transcendantes ou de gérer les interruptions de l'entrée du MSX-MIDI interne.                                                                                                                |
| 0FF7Ah  | <b>H.FING</b>                                                                                                                                                                                                                          |
| Appel : | A la fin de la routine de calcul de fonctions transcendantes.                                                                                                                                                                          |
| Rôle :  | Traiter la routine.                                                                                                                                                                                                                    |
| 0FF7Fh  | <b>H.ISMI</b> (« IS MId\$ »)                                                                                                                                                                                                           |
| Appel : | Au début de la routine ISMID\$, utilisée lors de l'instruction MID\$ du Basic.                                                                                                                                                         |
| Rôle :  | Traiter l'instruction.                                                                                                                                                                                                                 |
| 0FF84h  | <b>H.WIDT</b> (« WIDTh of screen »)                                                                                                                                                                                                    |
| Appel : | Au début de la routine WIDTHS, utilisée lors de l'instruction Basic WIDTH.                                                                                                                                                             |
| Rôle :  | Traiter l'instruction.                                                                                                                                                                                                                 |

|         |                                                                                                                                        |
|---------|----------------------------------------------------------------------------------------------------------------------------------------|
| 0FF89h  | <b>H.LIST</b>                                                                                                                          |
| Appel : | Au début de l'instructions Basic LIST (et LLIST).                                                                                      |
| Rôle :  | Traiter l'instruction.                                                                                                                 |
| 0FF8Eh  | <b>H.BUFL</b> (« Buffer Line »)                                                                                                        |
| Appel : | Par la routine de gestion du cache d'une ligne en cours de traitement sous Basic.                                                      |
| Rôle :  | Traiter la routine.                                                                                                                    |
| 0FF93h  | <b>H.FRQI</b> (« FReQuent Interrupt »), ou <b>H.MDTM</b> (« MiDi TiMer »)                                                              |
| Appel : | Au début de la routine FRQINT (MSX1/2/2+), ou au début de la routine d'interruption H.MDTM du Timer du MSX-MIDI interne (MSX Turbo R). |
| Rôle :  | Traiter la routine ou, de gérer l'interruption du Timer (8253) du MSX-MIDI qui se produit toutes les 5 ms.                             |
| 0FF98h  | <b>H.SCNE</b>                                                                                                                          |
| Appel : | Au début de la routine SCNEX2 de l'interpréteur Basic, conversion d'un numéro de ligne en adresse mémoire et inversement.              |
| Rôle :  | Traiter la routine.                                                                                                                    |
| 0FF9Dh  | <b>H.FRET</b> (« FREe up Tempories »)                                                                                                  |
| Appel : | Au début de la routine FRETMP de l'interpréteur Basic, recherche d'un emplacement libre pour stocker une variable alphanumérique.      |
| Rôle :  | Traiter la routine.                                                                                                                    |
| 0FFA2h  | <b>H.PTRG</b> (« PoinTeR Get »)                                                                                                        |
| Appel : | Au début de la routine PTRGET de l'interpréteur Basic, recherche d'une variable.                                                       |
| Rôle :  | Permet d'utiliser d'autres type de variables que celle par défaut.                                                                     |
| 0FFA7h  | <b>H.PHYD</b> (« PHYsical Disk i/o »)                                                                                                  |
| Appel : | Au début de la routine PHYDIO du Bios (Main-ROM à 0144h).                                                                              |
| Rôle :  | Traiter la routine. Normalement, si 0FFA7h contient 0C9h, c'est qu'il n'y a pas de disque installé.                                    |
| Note :  | Utilisée par la Disk-ROM.                                                                                                              |
| 0FFACh  | <b>H.FORM</b> (« disk FORMatter »)                                                                                                     |
| Appel : | Au début de la routine FORMAT du Bios (Main-ROM à 0147h).                                                                              |
| Rôle :  | Traiter la routine.                                                                                                                    |
| Note :  | Utilisée par la Disk-ROM.                                                                                                              |

|         |                                                                                                                                                                                        |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0FFB1h  | <b>H.ERRO</b> (« ERROr »)                                                                                                                                                              |
| Appel : | Au début de la routine ERROR de traitement d'une erreur sous Basic. Le registre A contient le numéro d'erreur (voir le <a href="#">tableau des codes d'erreurs du Basic</a> page 600). |
| Rôle :  | Permet de traiter des erreurs avec sa propre routine.                                                                                                                                  |
| 0FFB6h  | <b>H.LPTO</b> (« Line PrinTer Output »)                                                                                                                                                |
| Appel : | Au début de la routine LPTOUT, sortie d'un caractère sur imprimante.                                                                                                                   |
| Rôle :  | Permet d'utiliser une imprimante non-MSX.                                                                                                                                              |
| 0FFBBh  | <b>H.LPTS</b> (« Line PrinTer Status »)                                                                                                                                                |
| Appel : | Au début de la routine LPTSTT, test de l'état de l'imprimante.                                                                                                                         |
| Rôle :  | Permet d'utiliser une imprimante non-MSX.                                                                                                                                              |
| 0FFC0h  | <b>H.SCRE</b> (« SCREen »)                                                                                                                                                             |
| Appel : | Au début de l'instruction SCREEN du Basic.                                                                                                                                             |
| Rôle :  | Traiter l'instruction.                                                                                                                                                                 |
| 0FFC5h  | <b>H.PLAY</b> (« PLAY »)                                                                                                                                                               |
| Appel : | Au début de l'instruction PLAY du Basic.                                                                                                                                               |
| Rôle :  | Traiter l'instruction.                                                                                                                                                                 |
| 0FFCFh  | <b>H.BGFD</b>                                                                                                                                                                          |
| Appel : | Juste avant chaque accès à un disque. (MSX-DOS1 seulement)                                                                                                                             |
| Rôle :  | Détourner les instructions.                                                                                                                                                            |
| Note :  | Obsolète depuis l'introduction du Bios étendu. Remplacé par la routine DISINT du Bios étendu ( <a href="#">voir Bios étendu</a> page 524).                                             |
| 0FFD4h  | <b>H.ENFD</b> (« ENd Floppy Disk »)                                                                                                                                                    |
| Appel : | Juste après chaque accès à un disque. (MSX-DOS1 seulement)                                                                                                                             |
| Rôle :  | Ajouter une routine après chaque accès à un disque.                                                                                                                                    |
| Note :  | Obsolète depuis l'introduction du Bios étendu. Remplacé par la routine ENAINT du Bios étendu ( <a href="#">voir Bios étendu</a> page 524).                                             |

## 7 Le processeur vidéo (VDP)

### 7.1 *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 au MSX turbo R. Quant aux registres, ils seront entièrement détaillés afin de pouvoir y accéder directement sans avoir de passer par le Bios au cas où vous avez besoin d'un maximum de vitesse.

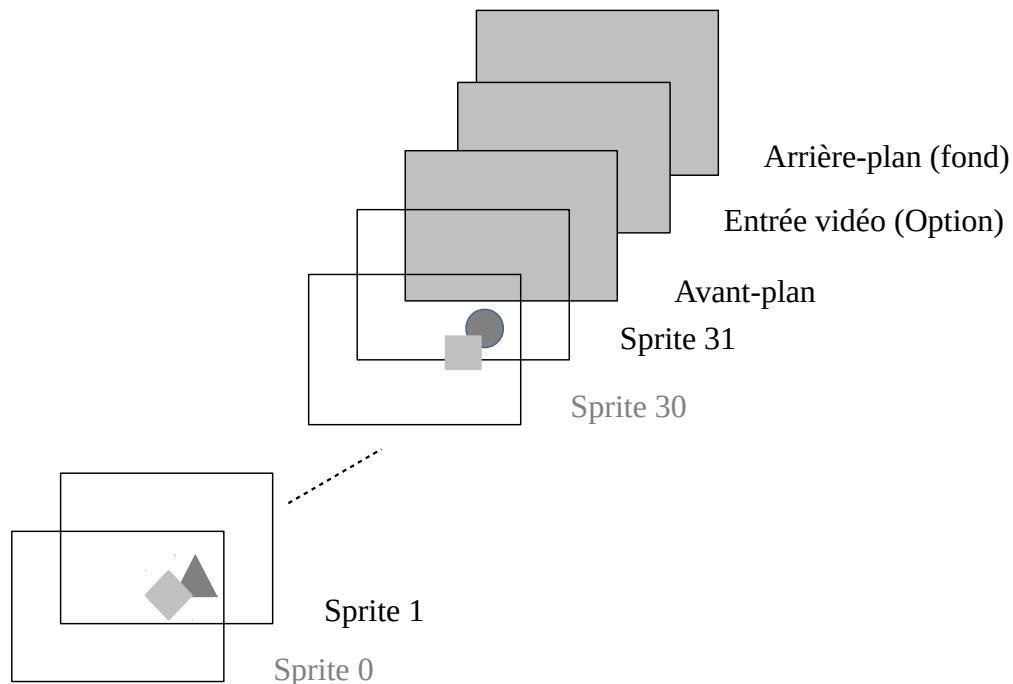
### 7.2 *Introduction au processeur vidéo*

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 dédiée (128 Ko en général) qui offre une haute résolution graphique pour du matériel grand public de l'époque et autorise l'affichage de deux cent cinquante six couleurs simultanément à l'écran et même plusieurs milliers sur MSX2+. Il 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.

### 7.3 *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 processeur dédié à l'affichage. En contre partie, il est nécessaire de transférer les données vers la VRAM via des ports E/S. 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 données, Sprite, 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 généralement 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 comme les zones en VRAM où se trouvent les données qui servent, par exemple, à afficher l'avant-plan ou les Sprite, la couleur du texte ou de l'arrière-plan alors que quelques registres ont une fonction spéciale, qui déclenche automatiquement une commande graphique, le tracé d'une ligne par exemple, lors d'une écriture dans le registre 46. Vous trouverez tout au long de ce chapitre l'explication sur le contenu de chaque registre ainsi qu'un exposé des différents modes graphiques et des Sprite.

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.



L'arrière-plan n'est visible que sur les bordures ou à travers la couleur 0 de l'avant-plan. L'entrée vidéo remplace tout l'arrière-plan lorsqu'il est activé.

32 Sprite peuvent être affichés par dessus l'avant-plan mais si il y a plus de 4 ou 8 Sprite (selon le mode d'écran) sur une même ligne seules les 4 ou 8 Sprite prioritaires pourront être affichés sur cette ligne.

## 7.4 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 accédez directement au VDP via les ports d'entrée/sortie pour une application qui nécessite une vitesse la plus élevée possible car l'utilisation du Bios peut ralentir significativement les accès.

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, ASCII et Microsoft 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 d'entrée/sortie (ports E/S) :



Dans le Bios en Main-ROM, il y a 2 octets à l'adresse 0006h (VDP.DR) et 0007h (VDP.DW) qui servent à indiquer les ports E/S qui permettent d'accéder au VDP. L'adresse 0006h contient le numéro du premier port d'entrée (lecture) et l'adresse 0007h le numéro du premier port d'écriture. Les autres ports se trouvent à la suite comme indiqué dans le tableau suivant. (Si vous ne comprenez pas la signification de Main-ROM, allez voir le chapitre concernant [la ROM principale](#).)

| Port de lecture du VDP | Numéro du port d'entrée du CPU | Fonction                                  |
|------------------------|--------------------------------|-------------------------------------------|
| 0                      | Contenu de 0006h (VDP.DR)      | Lecture d'une donnée en VRAM              |
| 1                      | Contenu de 0006h (VDP.DR) +1   | Lecture du registre de statut sélectionné |

| Port d'écriture au VDP | Numéro du port de sortie du CPU | Fonction                                                                  |
|------------------------|---------------------------------|---------------------------------------------------------------------------|
| 0                      | Contenu de 0007h (VDP.DW)       | Écrire une donnée en VRAM                                                 |
| 1                      | Contenu de 0007h (VDP.DW) +1    | Écrire dans un registre ou définir l'adresse d'accès en VRAM              |
| 2                      | Contenu de 0007h (VDP.DW) +2    | Écrire dans la palette des couleurs                                       |
| 3                      | Contenu de 0007h (VDP.DW) +3    | Envoi de données successives dans un ou une suite de registre de contrôle |

Les MSX1 sont équipés en général d'un VDP de première génération (par exemple un TMS9918 ou TMS9929 selon le modèle). Ces VDP ont deux ports d'écriture et deux de lecture (ports 0 et 1).

Les MSX de générations suivantes sont équipés d'un VDP V9938 (MSX2) ou d'un V9958 (MSX2+ et MSX turbo R). Ces VDP ont deux ports d'écriture (ports 2 et 3) supplémentaires.

#### Notes :

- Certains MSX1 sont équipés d'un V9938 (le Yamaha CX5MII et le Spectravideo SVI-738). Il y a même quelques MSX2 équipés d'un V9958. Leur Bios ne contient cependant que les routines correspondantes à leur génération.
- Dans les faits, tous les MSX utilisent les ports 098h et 099h pour accéder au VDP interne. Les ports d'écriture 098h à 09Bh ont été ajoutés à partir du MSX2. Ça a d'ailleurs été officialisé à la sortie du MSX2+. Seuls les VDP externes utilisent des ports différents. Et le seul VDP externe dont les ports E/S sont standardisés est le V9938 utilisé pour l'extension MSX1 vers MSX2 ou pour le mode 80 colonnes. Ce V9938 utilise les ports de 088h à 08Bh.
- Si vous décidez de faire des accès directs au VDP, vous devez respecter les timings indiqués dans la documentation technique des TMS9918 à TMS9929, du V9938 et V9958. En général, pour être sûr d'éviter les problèmes dans les modes SCREEN1 à 3, n'utilisez pas IN, INI, OUT et OUTI directement les uns à la suite des autres pour écrire ou lire dans la VRAM hors de la [Vblank](#). Mettez un temps d'attente de 27 cycles (T-state) minimum entre deux accès à la VRAM. N'utilisez pas non plus INIR ou OTIR hors de la [Vblank](#).

## Ecrire dans un registre de contrôle (0 à 23, 25 et 32 à 46)

1. Envoyer la donnée à écrire sur le port 1.
2. Envoyer, toujours sur le port 1, le numéro du registre dans lequel vous voulez écrire avec le bit 7 à 1, ce qui revient à ajouter 80h au numéro du registre. Si les bit 6 et 7 sont à 0 la valeur ne sera pas prise en compte et vous devriez recommencer l'opération en ré-envoyant la donnée.

Si le registre de statut 0 est lu avant l'écriture du numéro du registre (à cause d'une interruption par exemple), la donnée sera perdue. De plus, comme la palette utilise le même cache de deux octets, veillez à qu'aucune donnée ne soit envoyée à la palette entre l'envoi la donnée et le numéro du registre.

Exemple en assembleur pour MSX2 ou supérieur :

```
; Activer le mode d'affichage à 60 Hz. (Mettre le bit 2 du
; registre de contrôle 9 du VDP à 0)

; Sur un téléviseur ou un moniteur en 50 Hz, l'image "sautera".
; Remettre le bit 2 à 0 dans le registre neuf afin de
; rétablir une image normale.

DEBUT:  ld      a,(0007h)      ; La Main-ROM doit être présente !
        ld      c,a           ; C = Numéro du port d'écriture 0 au VDP.
        inc     c             ; C = Port d'écriture 1 au VDP.
;
        ld      a,(0ffe8h)    ; Met le contenu de RG9SAV 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 la variable RG9SAV
        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 RG9SAV afin de préserver
; l'état des autres bits du registre
```

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 des données dans une suite de registres ou d'écrire plusieurs fois dans le même registre. Pour cela, à partir du V9938, un 3<sup>e</sup> port a été ajouté au processeur vidéo. Voici comment procéder :

1. Écrire le numéro du registre dans lequel vous désirez écrire dans le registre 17 de contrôle avec la méthode précédente.
2. 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. >

```

DEBUT:
    ld    a,(0007h)      ; La Main-ROM doit être présente !
    ld    c,a            ; C = Numéro du port d'écriture 0 au VDP.
    inc   c              ; C = Port d'écriture 1 au VDP.

    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 RG9SAV 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 RG9SAV

    call  ATTEND         ; Appel le sous-programme d'attente

    or    2              ; Met le bit 2 à 1 (mode 50 Hz)
    ld    (0ffe8h),a     ; Actualise RG9SAV
    out   (c),a          ; Envoi sur le port 3
    ret

ATTEND:
    ld    b,a            ; Préserve A
    push  bc             ; et C
    ld    bc,0           ; Boucle d'attente

BCL:
    dec   bc             ; Décremente BC
    nop                      ; Petit temps mort
    ld    a,b
    or    c
    cp    0
    jr    nz,BCL         ; Saute si BC > 0
    pop   bc             ; Restitue C
    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.
2. Envoyez sur le port 2 du VDP, deux octets qui codent la nouvelle palette de cette couleur au format suivant.

|                 | bit 7 | bit 6       | bit 5 | bit 4 | bit 3 | bit 2      | bit 1 | bit 0 |
|-----------------|-------|-------------|-------|-------|-------|------------|-------|-------|
| Premier octet : | 0     | rouge (0~7) |       |       | 0     | bleu (0~7) |       |       |

|                  | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2      | bit 1 | bit 0 |
|------------------|-------|-------|-------|-------|-------|------------|-------|-------|
| Deuxième octet : | 0     | 0     | 0     | 0     | 0     | vert (0~7) |       |       |

Veillez à qu'il n'y ait pas d'accès aux registres de contrôle entre l'envoi de deux données de la palette car le même cache de deux octets est utilisé pour faire ces opérations. De plus, comme une lecture du registre de statut initialise ce cache, veuillez aussi à qu'aucune interruption ne se produise.

Exemple :

```
; Définir la couleur 1 (noir par défaut) en vert (MSX2~)
    org 0c000h
DEBUT:
    ld  a,(0007h) ; La Main-ROM doit être présente !
    ld  c,a       ; C = Numéro du port d'écriture 0 au VDP.
    inc c        ; C = Port d'écriture 1 au VDP.
;
    ld  a,1       ; A=1
    di                    ; Interruptions interdites.
    out (c),a     ; Envoie de la donnée 1.
    ld  a,80h+16  ; A=16 (avec le bit 7 à 1)
    out (c),a     ; Ecriture de la donnée 1 dans le registre 16.
;
    inc c        ; Port E/S du port 2 d'écriture
    ld  a,42h     ; rouge=4 bleu =2
    out (c),a     ; Envoie de la donnée 42h.
    ld  a,5       ; vert=5
    out (c),a     ; Envoie de la donnée 5.
    ei                    ; Interruptions autorisées
    ret
```

## Lire un registre de statut du VDP

Sur un MSX1, il n'y a qu'un seul registre de statut donc il suffit de lire le port correspondant. Sur MSX2 ou plus récent, par défaut vous pouvez procéder comme sur MSX1 seulement pour lire le registre de statut 0. Autrement, il faut procéder de la façon suivante.

1. Écrire le numéro de registre de statut que vous désirez lire (0 ~ 9) dans le registre de contrôle 15.
2. Lire le port 1 du VDP pour obtenir le contenu du registre. (Sur MSX1, seule cette étape est nécessaire.)
3. Remettre 0 dans le registre de contrôle 15. Cette étape est nécessaire car à chaque interruption la routine du 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,(0007h)    ; La Main-ROM doit être présente !
    ld  c,a          ; C = Numéro du port d'écriture 0 au VDP.
    inc c           ; C = Port d'écriture 1 au VDP.
    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,(0006h)     ; La Main-ROM doit être présente !
    ld  c,a           ; C = Numéro du port de lecture 0 du VDP
    inc c            ; C = Numéro du port de lecture 1 du VDP
    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
```

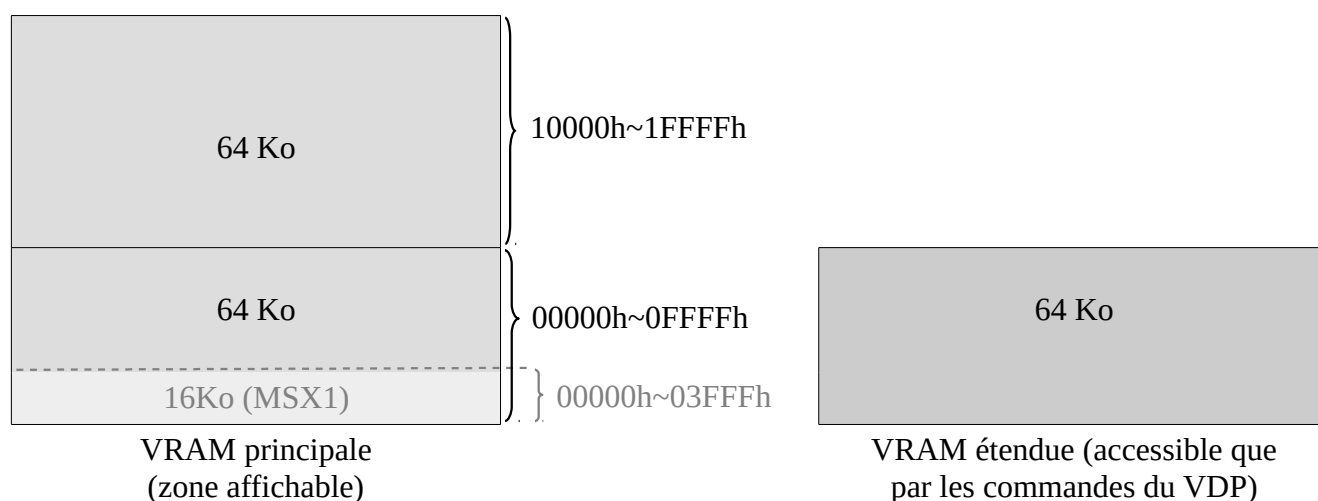
Note : À chaque interruption la routine d'interruption lit le [registre de statut 0](#) avec l'instruction IN A, (099h) seule donc sur MSX2 ou plus récent, il faut remettre le [registre 15](#) à 0 avant qu'une interruption se produise après la lecture d'un autre registre de statut pour assurer le bon fonctionnement du système.

## Lecture et écriture dans la mémoire vidéo

Un MSX est équipé de 16 Ko , 64 Ko ou 128 Ko de mémoire vive dédiée à la vidéo (VRAM). Tous les MSX1 ont 16Ko de VRAM. Les MSX2 commercialisés en France comportent tous 128 Ko mais il en existe qui n'ont que 64Ko. Les générations suivantes ont tous 128Ko de VRAM. Cette mémoire ne se trouve pas dans un Slot, ce faisant, le Z80 ne peut pas l'adresser. Seul le processeur vidéo peut le faire.

Les VDP v9938 et v9958 peuvent accéder à 64 Ko de VRAM supplémentaire, ce qui peut faire un total de 192Ko mais cette possibilité n'est pas incluse aux standard MSX. Le schéma ci-dessous montre comment la mémoire vidéo est disposée.

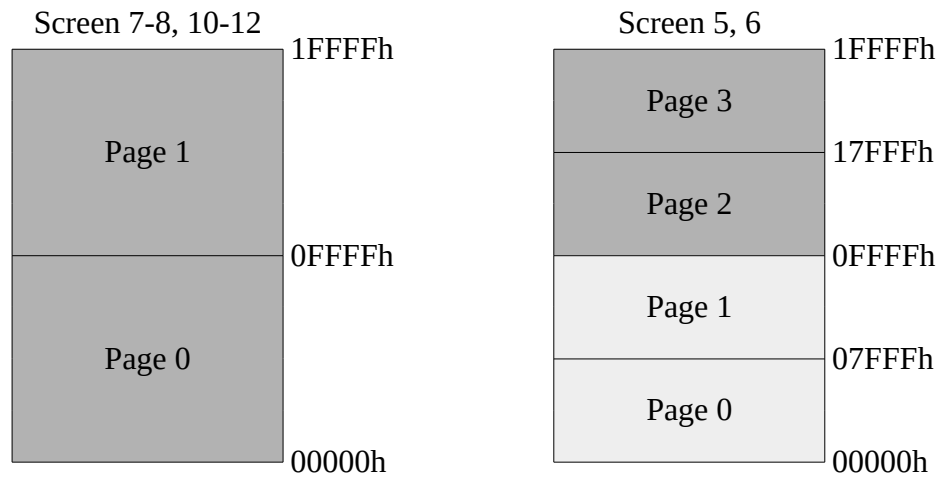
Carte de la VRAM



Il existe trois méthodes permettant de manipuler la mémoire vidéo. La première consiste à utiliser le Bios, puisqu'on y trouve les routines FILVRM, LDIRMV, RDVRM, SETRD, SETWRT, WRTVRM qui permettent écrire ou lire dans 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 disponibles à partir du MSX2. Pour ce qui est de cette dernière solution, elle est détaillée dans la partie concernant les commandes propres au VDP.

Notez que, dans les modes graphiques 6 et 7 (SCREEN 7 à 8 et 10 à 12), les octets des VRAM sont entrelacés pour que le VDP puisse y accéder plus rapidement. C'est la raison pour laquelle ces modes d'écran ne sont pas disponibles pour les MSX2 ayant que 64Ko de VRAM.

Dans les modes d'écran 5 à 8 et 10 à 12, le système divise la VRAM par « page » de la façon suivante. Veuillez à ne pas confondre ces pages avec les tables Bitmap / YAE / YJK.



Les pages en gris foncées ne sont utilisables que sur les MSX ayant 128 Ko de VRAM à cause de l'entrelacement des octets. Le système ne gère pas la VRAM étendue.

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.

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.)

1. Envoyer sur le port 1 du VDP les 8 bits de poids faible de l'adresse (bits A7 à A0).
2. 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    | A8    |

Si les bit 6 et 7 sont à 0 la valeur ne sera pas prise en compte et vous devriez recommencer l'opération en ré-envoyant les 8 bits de poids faible.

3. 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 assembleur :

< 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,(0007h)    ; La Main-ROM doit être présente !
ld       c,a          ; C = Numéro du port d'écriture 0 au VDP.
inc      c            ; C = Port d'écriture 1 au VDP.
;
ld       a,0          ; 3 bits de poids fort      } Supprimez
di                          ; Interruptions interites

```

```

        out    (c),a      ;                               } ces quatre
        ld     a,80h+14   ; 3 bits de poids fort         } lignes sur
        out    (c),a      ; dans le registre 14          } MSX1
;
        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
;
        push   hl         ; Temps d'attente
        pop    hl        ; pour VDP MSX1
;
        dec    c          ; Numéro du port E/S relié au port 0 dans C
        ld     hl,DATA    ; Position des données
        ld     b,9        ; Neuf lettres
ENVOIS:
        outi                               ; Transfert des données sur le port 0
        djnz   ENVOIS     ; (Peut-être remplacé par OTIR sur MSX2~)
        ret
DATA:    db      'CA MARCHE'

```



## 7.5 Les registres de statut du VDP

Les registres de statut donnent des informations sur l'état actuel du VDP. Ils sont donc accessibles en lecture seulement. Voici la liste complète de tous ces registres avec la signification des informations qu'ils contiennent.

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 0 :</b> | F     | 5S    | C     | 5/9SN |       |       |       |       | (Tous les VDP) |

5/9SN = Ces bits donnent le numéro du 5<sup>e</sup> Sprite (ou du 9<sup>ème</sup> dans les SCREEN 4 à 12) lorsque le bit 5S se trouve à 1.

C = Ce bit passe à 1 lorsque deux Sprite entrent en collision.

5S = Indicateur qui passe à 1 lorsque 5 Sprite (9 dans les SCREEN 4 à 12) se trouvent sur une même ligne.

F = Indicateur de demande d'interruption qui se produit à chaque fin d'affichage de l'avant-plan (de l'avant-plan). Ce bit repasse à 0 (la demande d'interruption est annulé) chaque fois que le registre est lu ou que le VDP est initialisé (de façon externe).

Note : Si malencontreusement, le VDP provoque une interruption pendant la lecture de ce registre, le bit F n'aura par le temps de passer à 1 et la valeur lue sera 0. Donc, n'essayer pas de cadencer votre programme à 50/60 Hz par une boucle qui lit ce registre sous peine de "perdre" quelques frames. Il y a la variable système JIFFY (0FC9Eh) pour ça.

|                     | bit 7     | bit 6      | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                                  |
|---------------------|-----------|------------|-------|-------|-------|-------|-------|-------|----------------------------------|
| <b>Registre 1 :</b> | <u>FL</u> | <u>LPS</u> | ID    |       |       |       |       | FH    | ( <u>V9938</u> et <u>V9958</u> ) |

FH = Indicateur de demande d'interruption qui se produit juste avant le balayage de la ligne horizontale indiquée par le registre 19. Ce bit repasse à 0 (la demande d'interruption est annulé) chaque fois que le registre est lu ou que le VDP est initialisé (de façon externe).

ID = Ces 5 bits 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. (Utilisé uniquement sur les MSX2 coréens Daewoo CPC-300 et CPC-400/400S .) (Ignoré 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. (Utilisé uniquement sur les MSX2 coréens Daewoo CPC-400 et CPC400S.) (Ignoré sur le V9958.)

Note : Pour plus de renseignements sur le fonctionnement du crayon optique et de la souris, voir le paragraphe 7.10 « [A propos de la souris et du crayon optique](#) », page 370.

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| <b>Registre 2 :</b> | TR    | VR    | HR    | BD    | 1     | 1     | EO    | CE    | (V9938 et V9958) |

CE = Ce bit (Command Execution) est à 1, lorsque le VDP est en train d'exécuter une commande interne.

EO = Indicateur de période. 0 pour la première période; 1 pour la seconde période.

BD = Ce bit (Bordure Detect) passe à 1 lorsque la commande SRCH a trouvée la couleur de bordure.

HR = Ce bit (Horizontal Retrace) est à 1 pendant le balayage de la zone non visible de gauche et de droite de l'écran (HBLANK).

VR = Ce bit (Vertical Retrace) est à 1 pendant le balayage de la zone non visible supérieure et inférieure de l'écran ([Vblank](#)).

TR = Ce bit (Transfer ready) est à 1 lorsque le transfert d'un octet entre le CPU et la VRAM est prêt (voir les commandes [HMMC](#), [LMMC](#) ou [LMCM](#) page 294).

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| <b>Registre 3 :</b> | X7    | X6    | X5    | X4    | X3    | X2    | X1    | X0    | (V9938 et V9958) |
| <b>Registre 4 :</b> | 1     | 1     | 1     | 1     | 1     | 1     | 1     | X8    | (V9938 et V9958) |

X0 à X8 = Indiquent l'abscisse du point de collision des Sprite.

(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 |                  |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| <b>Registre 5 :</b> | Y7    | Y6    | Y5    | Y4    | Y3    | Y2    | Y1    | Y0    | (V9938 et V9958) |
| <b>Registre 6 :</b> | 1     | 1     | 1     | 1     | 1     | 1     | Y9    | Y8    | (V9938 et V9958) |

Y0 à Y9 = Indiquent l'ordonnée du point de collision des Sprite.

(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 |                  |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| <b>Registre 7 :</b> | C7    | C6    | C5    | C4    | C3    | C2    | C1    | C0    | (V9938 et V9958) |

C0 à C7 = Utilisés par les commandes POINT et LMCM. Contient l'octet lu en VRAM.

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| <b>Registre 8 :</b> | BX7   | BX6   | BX5   | BX4   | BX3   | BX2   | BX1   | BX0   | (V9938 et V9958) |
| <b>Registre 9 :</b> | 1     | 1     | 1     | 1     | 1     | 1     | 1     | BX8   | (V9938 et V9958) |

BX0 à BX8 = Indiquent l'abscisse de la couleur trouvée lors d'une recherche avec la commande SRCH du VDP.

## 7.6 Les registres de contrôle du processeur video

Ces registres permettent de contrôler le comportement du VDP. Ils sont accessibles seulement en écriture. Il est donc conseillé, voir nécessaire, d'actualiser les variables système correspondantes juste après une écriture dans ces registres (si vous n'utilisez pas le BIOS pour cela).

Voici la liste complète de tous les registres de contrôle et la signification des informations qu'ils contiennent classés par genre :

### Registres de mode de fonctionnement (0, 1, 8, 9 et 25)

|                     | bit 7 | bit 6 | bit 5      | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|------------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 0 :</b> | 0     | DG    | <u>IE2</u> | IE1   | M5    | M4    | M3    | EV    | (Tous les VDP) |

EV = Entrée vidéo externe. 1 pour activer ; 0 pour désactiver (valeur par défaut).

M3 à M5 = Bits de mode graphique. (Seul le bit M3 est utilisé sur MSX1 mais mettez M4 et M5 à 0 dans ce cas pour garder la compatibilité avec les MSX de génération supérieure.)

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 sauf les MSX2 coréens Daewoo CPC-300 et CPC-400/400S, mettre à 0 sinon.) (Inutilisé 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.)

Notes :

- Pour plus de détails sur les bits de mode graphique, voir la page suivante.
- Le système sauvegarde la valeur écrite dans ce registre dans la variable RG0SAV (0F3DFh).

|                     |              |       |       |       |       |       |       |       |                |
|---------------------|--------------|-------|-------|-------|-------|-------|-------|-------|----------------|
|                     | bit 7        | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
| <b>Registre 1 :</b> | <u>4/16k</u> | BL    | IE0   | M1    | M2    | 0     | SI    | MAG   | (Tous les VDP) |

MAG = Doublement de la taille des pixels des Sprite. 1 pour double ; 0 pour taille ordinaire.

SI = Taille des Sprite. 1 pour 16x16 ; 0 pour 8x8.

M1 et M2 = Bits de mode graphique. (Ces bits se combinent avec M3~5 du registre 0)

IE0 = 1 pour autoriser l'interruption qui se produit à chaque fin d'affichage de l'écran (avant-plan).

BL = Affichage de l'écran. 1 pour activer (valeur par défaut). Les commandes du VDP travaillent un peu plus vite lorsque l'écran est désactivé.

4/16k = Taille de la VRAM. 1 pour 16Ko ; 0 pour 4Ko. Etant donné qu'il n'y a pas de MSX1 avec seulement 4Ko de VRAM et que ce bit est ignoré par le V9938 et le V9958, alors mieux faut mettre le toujours à 1. (VDP MSX1 uniquement)

Note : Le système sauvegarde la valeur écrite dans ce registre dans la variable RG1SAV (0F3E0h).

Modes d'écran possibles :

| Mode        | M5 | M4 | M3 | M2 | M1 | VDP ayant ce mode            |
|-------------|----|----|----|----|----|------------------------------|
| Texte 1     | 0  | 0  | 0  | 0  | 1  | Tous les VDP                 |
| Texte 2     | 0  | 1  | 0  | 0  | 1  | <u>V9938</u> et <u>V9958</u> |
| Multi-color | 0  | 0  | 0  | 1  | 0  | Tous les VDP                 |
| Graphique 1 | 0  | 0  | 0  | 0  | 0  | Tous les VDP                 |
| Graphique 2 | 0  | 0  | 1  | 0  | 0  | Tous les VDP                 |
| Graphique 3 | 0  | 1  | 0  | 0  | 0  | <u>V9938</u> et <u>V9958</u> |
| Graphique 4 | 0  | 1  | 1  | 0  | 0  | <u>V9938</u> et <u>V9958</u> |
| Graphique 5 | 1  | 0  | 0  | 0  | 0  | <u>V9938</u> et <u>V9958</u> |
| Graphique 6 | 1  | 0  | 1  | 0  | 0  | <u>V9938</u> et <u>V9958</u> |
| Graphique 7 | 1  | 1  | 1  | 0  | 0  | <u>V9938</u> et <u>V9958</u> |

Le système quant à lui peut avoir jusqu'à 13 modes d'écran (SCREEN 0 à 12). Voici les modes du VDP utilisés par le système :

| Modes du système             | Modes du VDP     | Résolution et couleurs affichable                               |
|------------------------------|------------------|-----------------------------------------------------------------|
| SCREEN 0                     | Texte 1          | 40 colonnes, 2 couleurs (4 couleurs possible sur MSX2~)         |
| SCREEN 0                     | Texte 2          | 80 colonnes, 2 couleurs (4 couleurs possible)                   |
| SCREEN 1                     | Multi-color      | 32 colonnes, 16 couleurs (2 couleurs par caractères)            |
| SCREEN 2                     | Graphique 1      | 256x192, 16 couleurs (2 couleurs par ligne de motif)            |
| SCREEN 3                     | Graphique 2      | 64x48, 16 couleurs (sans contrainte)                            |
| SCREEN 4                     | Graphique 3      | Equivalent au SCREEN 2 mais avec des Sprite MSX2                |
| SCREEN 5                     | Graphique 4      | 256x212, 16 couleurs (sans contrainte)                          |
| SCREEN 6                     | Graphique 5      | 512x212, 4 couleurs (sans contrainte)                           |
| SCREEN 7                     | Graphique 6      | 512x212, 16 couleurs (sans contrainte)                          |
| SCREEN 8                     | Graphique 7      | 256x212, 256 couleurs (sans contrainte)                         |
| SCREEN 9                     | Graphique 4 ou 5 | 40 / 80 colonnes en Hangeul, 4 couleurs (MSX2 coréen seulement) |
| <a href="#">SCREEN 10/11</a> | Graphique 7 YAE  | 256x212, 12599 couleurs (avec contrainte)                       |
| <a href="#">SCREEN 12</a>    | Graphique 7 YJK  | 256x212, 19268 couleurs (avec contrainte)                       |

- Le mode SCREEN 9 du système est basé sur le mode Graphique 4 (ou 5 selon le nombre de colonnes utilisées) mais géré à la façon d'un mode texte par le Basic. Ce mode n'existe que sur certains MSX2 coréens.
- Le mode SCREEN 10 et 11 est obtenu en mettant le mode Graphique 7 plus les bits YJK et YAE du [registre 25](#) à 1. La différence entre le mode SCREEN 10 et 11 se situe dans la gestion des couleurs du tracé par les instructions du Basic (RGB ou YJK).
- Le mode SCREEN 12 est obtenu en mettant le mode Graphique 7 plus le bit YJK à 1 et le bit YAE à 0 ([registre 25](#)).

Pour la description détaillée des modes graphiques du système, voir le [chapitre des modes d'affichage](#) (page 321).

|                     |           |           |           |       |       |       |       |       |                                  |
|---------------------|-----------|-----------|-----------|-------|-------|-------|-------|-------|----------------------------------|
|                     | bit 7     | bit 6     | bit 5     | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                                  |
| <b>Registre 8 :</b> | <u>MS</u> | <u>LP</u> | <u>TP</u> | CB    | VR    | 0     | SPD   | BW    | ( <u>V9938</u> et <u>V9958</u> ) |

BW = Affichage en noir et blanc. Inutilisé sur MSX.

SPD = Désactivation des Sprite. Mettre à 1 pour désactiver. Les commandes du VDP travaillent un peu plus vite ainsi !

VR = Défini le type de mémoire vidéo (« Video Ram »). 1 pour 64Ko × 1 bit ou bien 64Ko × 4 bits, ou 0 pour 16 Ko × 1 bit ou bien 16 Ko × 4 bits. Cette valeur est définie à l'initialisation du système et ne doit pas être modifiée.

CB = Direction du bus de couleur (« Color Bus ») du VDP. 1 pour entrée ; 0 pour sortie.

TP = Couleur 0 redéfinissable. Sur MSX1, la couleur 0 est toujours « transparente » pour permettre de voir l'image de l'entrée vidéo, entière ou en partie. A partir du MSX2, il est possible d'utiliser la couleur 0 comme les autres en mettant ce bit à 1. Ceci permet d'avoir une couleur supplémentaire. TP affecte également les Sprite de la même manière.

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 = Mode crayon optique. 1 pour activer. (Mode utilisé uniquement par les MSX2 coréens Daewoo CPC-300 et CPC-400/400S, mettre à 0 autrement.) (Sans effet sur le V9958.)

MS = Mode souris. 1 pour activer. (Inutilisé sur MSX, mettre à 0.) (Sans effet sur le V9958.)

Note : Le système sauvegarde la valeur qu'il écrit dans ce registre dans la variable REG8SAV (0FFE7h).

|                     |       |       |       |       |       |       |                 |       |                                  |
|---------------------|-------|-------|-------|-------|-------|-------|-----------------|-------|----------------------------------|
|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1           | bit 0 |                                  |
| <b>Registre 9 :</b> | LN    | 0     | S1    | S0    | IL    | E0    | $\overline{NT}$ | DC    | ( <u>V9938</u> et <u>V9958</u> ) |

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 pour rafraîchir l'affichage à 50 Hertz (pour les TV PAL et SECAM). Autrement, le rafraîchissement se fait à 60 Hertz (pour les TV NTSC et PAL-M).

E0 = Ce bit, lorsqu'il est à 1, sert à activer l'affichage alterné de 2 tables Bitmap ou de couleurs (la table actuelle et celle supérieure). La table supérieure est affichée pendant la deuxième période. (Voir le [registre 13](#) pour les précisions.)

IL = Permet d'afficher en mode entrelacé. Le bit 2 (E0) doit être aussi mis à 1 pour être pris en compte et les bits N15 et N16 du [registre 2](#) doivent indiquer la table Bitmap paire à entrelacer avec la table suivante. (0 à l'initialisation.)

S0 et S1 = Affichage. 0 pour Normal ; 1 pour Numérisation, incrustation, etc ; 2 pour Vidéo externe.

LN = 1 pour régler la hauteur de l'écran (l'avant-plan) à 212 pixels, sinon 192 pixels.

Note : Le système sauvegarde la valeur qu'il écrit dans ce registre dans la variable REG9SAV (0FFE8h).

|                      |       |       |                         |       |       |       |       |       |         |
|----------------------|-------|-------|-------------------------|-------|-------|-------|-------|-------|---------|
|                      | bit 7 | bit 6 | bit 5                   | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |         |
| <b>Registre 25 :</b> | 0     | CMD   | $\overline{\text{VDS}}$ | YAE   | YJK   | WTE   | MSK   | SP2   | (V9958) |

SP2 = 0 pour scrolling horizontal de l'écran sur une table Bitmap. (Valeur par défaut)

1 pour scrolling horizontal de l'écran sur deux tables. (La valeur du [registre 2](#) doit être impaire)

MSK = Active un masque d'une largeur de 8 ou 16 pixels sur la partie gauche de l'écran afin de masquer des copies lorsqu'on fait un défilement utilisant une seule table Bitmap.

WTE = 0 désactive la fonction « Wait », le VDP fonctionne comme un V9938. (Valeur par défaut)

1 active la fonction « Wait ». Met en attente le CPU pendant qu'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 des Sprite devient comme en SCREEN 4, 5 et 7.

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 :   | Y1    |       |       |       |       | bits de poids faible de K |       |       |
| Second octet :    | Y2    |       |       |       |       | bits de poids fort de K   |       |       |
| Troisième octet : | Y3    |       |       |       |       | bits de poids faible de J |       |       |
| Quatrième octet : | Y4    |       |       |       |       | 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 :

$$R=Y+J$$

$$G=Y+K$$

$$B=1,25Y-0,5J-0,25K$$

$$Y=(2R+G+4B)/8$$

$$J=(6R-G+4B)/8$$

$$K=(-2R+7G-4B)/8$$

YAE = 1 pour configurer la table Bitmap du SCREEN 8 en mode YAE. (Ignoré si YJK est à 0.) 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 :   | Y1    |       |       |       | A1    | bits de poids faible de K |       |       |
| Second octet :    | Y2    |       |       |       | A2    | bits de poids fort de K   |       |       |
| Troisième octet : | Y3    |       |       |       | A3    | bits de poids faible de J |       |       |
| Quatrième octet : | Y4    |       |       |       | A4    | bits de poids fort de J   |       |       |

La couleur du premier pixel 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.

$\overline{\text{VDS}}$  = Ce bit détermine le signal émis par la broche 8 du VDP. Mis à 1, un signal  $\overline{\text{VDS}}$  remplacera le signal d'horloge à 3,579545 MHz qui cadence le Z80A. (Laisser toujours ce bit à 0 !)

CMD = Ce bit permet d'utiliser les commandes du VDP dans tous les modes d'écran. Pour les modes SCREEN 0 à 4, les coordonnées X et Y fonctionnent comme en SCREEN 8. (0 par défaut.)

Note : Le système sauvegarde la valeur écrite dans ce registre dans la variable REG25SAV (0FFFAh).

## Registres des tables en VRAM (2, 3, 10, 4, 5, 11 et 6)

Ces registres définissent les tables en VRAM qui constituent ce qu'il y a à afficher à l'écran (avant-plan et Sprite).

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 2 :</b> | 0     | N16   | N15   | N14   | N13   | N12   | N11   | N10   | (Tous les VDP) |

Ce registre détermine l'emplacement de la table des positions de caractère, des motifs, Bitmap, YAE ou YJK.

La fonction de ces bits varie plus ou moins en fonction du mode d'écran utilisé. Voir la [description des modes d'écran](#) pour les détails.

Note : Le système sauvegarde la valeur écrite dans ce registre dans la variable REG2SAV (0F3E1h).

|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------------------------|
| <b>Registre 3 :</b>  | C13   | C12   | C11   | C10   | C9    | C8    | C7    | C6    | (Tous les VDP)                   |
| <b>Registre 10 :</b> | 0     | 0     | 0     | 0     | 0     | C16   | C15   | C14   | ( <u>V9938</u> et <u>V9958</u> ) |

Les registres 3 et 10 spécifient l'adresse en VRAM de la table des couleurs des caractères ou des motifs.

La fonction de ces bits varie plus ou moins en fonction du mode d'écran utilisé. Voir la [description des modes d'écran](#) pour les détails.

Notes :

- Le système sauvegarde la valeur écrite dans le registre 3 dans la variable REG3SAV (0F3E2h) et celle du registre 10 dans la variable RG10SAV (0FFE9h).
- Le registre 10 est à prendre en compte à partir du MSX2 même dans les modes SCREEN 1 à 2.



|                     |       |       |       |       |       |       |       |       |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
| <b>Registre 4 :</b> | 0     | 0     | F16   | F15   | F14   | F13   | F12   | F11   | (Tous les VDP) |

Ce registre détermine l'emplacement de la table des formes de caractère ou des motifs.

La fonction de ces bits varie plus ou moins en fonction du mode d'écran utilisé. Voir la [description des modes d'écran](#) pour les détails.

Note : Le système sauvegarde la valeur écrite dans ce registre dans la variable REG4SAV (0F3E3h).

|                      |       |       |       |       |       |       |       |       |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
| <b>Registre 5 :</b>  | S14   | S13   | S12   | S11   | S10   | S9    | S8    | S7    | (Tous les VDP)   |
| <b>Registre 11 :</b> | 0     | 0     | 0     | 0     | 0     | 0     | S16   | S15   | (V9938 et V9958) |

Modes d'écran SCREEN 0 à 2 :

S7 à S13 = Ces bits codent les 7 bits de poids fort de l'adresse du début de la table des attributs de Sprite. 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 Sprite est 037h\*80h=1B80h.

S14 = Ignoré. L'adresse peut varier entre 0 et 3F80h.

Dans les autres modes d'écran, les Sprite fonctionnent en mode 2. C'est à dire qu'une table de couleur de Sprite de 512 octets est ajoutée juste devant celle de la table des attributs de Sprite du mode 1. Chacun des octets de cette seconde table définissent la couleur d'une ligne horizontale des Sprite et quelques autres réglages. Les octets de la table des attributs qui définissait la couleur d'un Sprite en mode 1 est utilisé par le VDP pour autre chose.

SCREEN 4 à 6 :

S7 à S16 = Les bits S7-S8 sont toujours considérés à 0 et S9 doit toujours être à 1. L'adresse est obtenue par la formule (valeur de S7-S16) x 200h.

Autres modes d'écran :

S7 à S16 = Ces 10 bits codent l'adresse du début des données des attributs de Sprite comme sur MSX1 (voir ci-dessus). L'adresse peut varier entre 0 et 1FF80h.

Note : Le système sauvegarde la valeur écrite dans ce registre 5 dans la variable RG5SAV (0F3E4h) et celle du registre 11 dans la variable RG11SAV (0FFEAh).

|                     |       |       |       |       |       |       |       |       |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
| <b>Registre 6 :</b> | 0     | 0     | P16   | P15   | P14   | P13   | P12   | P11   | (Tous les VDP) |

Modes d'écran SCREEN 0 à 4 :

P11 à P13 = Codent les 3 bits de poids fort de l'adresse en VRAM du début des données de la forme des Sprite. L'adresse véritable s'obtient en multipliant la valeur sur 3 bits par 800h. Par conséquent, pour mettre la table des formes de Sprite à l'adresse 2000h, il faut diviser 2000h par 800h et écrire le résultat dans ce registre.

P14 à P16 = Ignorés. L'adresse peut varier dans ces conditions qu'entre 0 et 3800h.

Autres modes d'écran :

P11 à P16 = Codent les 6 bits de poids fort de l'adresse du début des données pour générer les Sprite. L'adresse véritable s'obtient en multipliant la valeur sur 6 bits par 800h. (Varie entre 0 et 1F800h)

Note : Le système sauvegarde la valeur écrite dans ce registre dans la variable REG6SAV (0F3E5h).

## Registres de réglage des couleurs de texte et de fond, et du clignotement (7, 12 et 13)

|                     |       |       |       |       |       |       |       |       |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
| <b>Registre 7 :</b> | TC3   | TC2   | TC1   | TC0   | BD3   | BD2   | BD1   | BD0   | (Tous les VDP) |

BD0 à BD3 = Couleur de l'arrière plan. En SCREEN 6, les bits BD0 et BD1 définissent la couleur de bordure et lorsque le bit BD3 est à 0, les lignes impaires de l'arrière plan prennent la couleur 0 et les lignes paires celle de l'arrière plan.

TC0 à TC3 = Couleur du texte dans le mode SCREEN 0. Ces bits sont ignorés dans les autres modes d'écran sauf en SCREEN 8 et 10 à 12. Dans ces modes, ces bits définissent aussi la couleur de l'arrière-plan (les 4 bits de point fort) comme en SCREEN 8.

Note : Le système sauvegarde la valeur écrite dans ce registre dans la variable REG7SAV (0F3E6h).

|                      |       |       |       |       |       |       |       |       |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
| <b>Registre 12 :</b> | T23   | T22   | T21   | T20   | BC3   | BC2   | BC1   | BC0   | (V9938 et 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.

Note : Le système sauvegarde la valeur écrite dans ce registre dans la variable RG12SAV (0FFEBh).

|                      |       |       |       |       |       |       |       |       |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
| <b>Registre 13 :</b> | ON3   | ON2   | ON1   | ON0   | OF3   | OF2   | OF1   | OF0   | (V9938 et V9958) |

Ce registre sert à régler le temps des deux périodes d'un affichage alterné

OF0 à OF3 = Règle le temps de la seconde période.

ON0 à ON3 = Règle le temps de la première période. (Si ces bits sont tous à 0, l'affichage périodique sera comme désactivé.)

Note : Le système sauvegarde la valeur qu'il écrit dans ce registre dans la variable RG13SAV (0FFEBh).

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 textes. La couleur de texte est définie par le registre 12. Celle-ci 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 première période.
2. Charger le registre 12 avec les couleurs du texte pour la seconde période.
3. Modifier la table des couleurs en VRAM. Cette table occupe 240 octets (0800h à 08EFh par défaut), chaque bit correspond à un 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 TO I+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 !":? 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 clignent.

Dans les modes SCREEN 5 à 12, ce registre sert à paramétrer les deux périodes d'affichage de deux tables Bitmap paire et impaire. La table paire inférieure s'affiche pendant première période. La table impaire pendant la seconde période. Il y a alternance des deux tables lorsque le bit 2 (E0) du registre 9 est à 1 si la valeur de 4 bits de poids fort du registre 13 (ON0~ON3) est supérieur à 0.

Marche à suivre :

Les tables Bitmap vont par deux. En SCREEN 5 et 6, on peut choisir de les faire afficher en alternance, les tables 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 table 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 tables sur la seconde période.

Exemple d'utilisation en Basic :

```
10 COLOR 10,0,0:SCREEN 5
20 ON STOP GOSUB 180: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 A$=INKEY$
```

```

90 IF A$="1" THEN VDP(14)=&H33:VDP(10)=VDP(10) OR 4
100 IF A$="2" THEN VDP(14)=&H3:VDP(10)=VDP(10) OR 4
110 IF A$="3" THEN VDP(14)=&H30:VDP(10)=VDP(10) OR 4
120 IF A$="4" THEN VDP(14)=&H0:VDP(10)=VDP(10) OR 4
130 IF A$="5" THEN VDP(14)=&H33:VDP(10)=VDP(10) AND 251
140 IF A$="6" THEN VDP(14)=&H3:VDP(10)=VDP(10) AND 251
150 IF A$="7" THEN VDP(14)=&H30:VDP(10)=VDP(10) AND 251
160 IF A$="8" THEN VDP(14)=&H0:VDP(10)=VDP(10) AND 251
170 GOTO 80
180 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 table 0 s'affiche pendant la première période puis les tables 1 et 0 s'affichent une frame sur deux pendant la deuxième période.
- 2 = Les tables 1 et 0 s'affichent une frame sur deux continuellement. L'affichage périodique est désactivé.
- 3 = La table 0 s'affiche pendant la première période. Il n'y a pas de deuxième période donc la table 0 reste toujours affichée.
- 4 = Les tables 1 et 0 s'affichent une frame sur deux continuellement. L'affichage périodiques est désactivé.
- 5 = Affiche la table 0 pendant la première période puis affiche la table 1 pendant la deuxième période.
- 6 = Affiche la table 1. L'affichage périodique est désactivé.
- 7 = Affiche la table 0 pendant la première période. Il n'y a pas de deuxième période donc la table 0 reste toujours affichée.
- 8 = Affiche la table 1. L'affichage périodique est désactivé. (paramètres appliqué à initialisation)

## Registres d'accès (14, 15, 16 et 17)

|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------------------------|
| <b>Registre 14 :</b> | 0     | 0     | 0     | 0     | 0     | A16   | A15   | A14   | ( <u>V9938</u> et <u>V9958</u> ) |

A14 à A16 = 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 « [Lecture et écriture dans la mémoire vidéo](#) », page 270.

Note : Le système sauvegarde la valeur écrite dans ce registre dans la variable RG14SAV (0FFEDh).

|                      |       |       |       |       |       |       |       |       |                                                    |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------------------------------------------|
|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                                                    |
| <b>Registre 15 :</b> | 0     | 0     | 0     | 0     | ST3   | ST2   | ST1   | ST0   | ( <a href="#">V9938</a> et <a href="#">V9958</a> ) |

ST0 à ST3 = Indiquent le numéro de registre de statut que l'on désire lire (0 ~ 9). Voir le paragraphe 7.4 « Comment accéder aux registres du VDP » rubrique « [Lire un registre de statut](#) » page 269 pour plus de précisions.

Note : Le système sauvegarde la valeur écrite dans ce registre dans la variable RG15SAV (0FFEEh).

|                      |       |       |       |       |       |       |       |       |                                                    |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------------------------------------------|
|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                                                    |
| <b>Registre 16 :</b> | 0     | 0     | 0     | 0     | CC3   | CC2   | CC1   | CC0   | ( <a href="#">V9938</a> et <a href="#">V9958</a> ) |

CC0 à CC3 = Spécifie le numéro de la couleur (0 ~ 15) à laquelle la palette sera modifié en envoyant des données au port d'entrée/sortie 09Ah. Les données sont envoyées par pair au format ci-dessous.

|             |       |       |       |       |       |       |       |       |                   |
|-------------|-------|-------|-------|-------|-------|-------|-------|-------|-------------------|
|             | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                   |
| Port 09Ah : | 0     | R2    | R1    | R0    | 0     | B2    | B1    | B0    | Première écriture |
| Port 09Ah : | 0     | 0     | 0     | 0     | 0     | V2    | V1    | V0    | Seconde écriture  |

Une fois les deux données du RVB envoyées, le numéro de la couleur est incrémenté.

Voir le paragraphe « Comment accéder au VDP » rubrique « [Ecrire dans un registre de la palette des couleurs](#) », page 268, 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 |

Note : Le système sauvegarde la valeur écrite dans ce registre dans la variable RG16SAV (0FFEFh).

|                      |       |       |       |       |       |       |       |       |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
| <b>Registre 17 :</b> | AII   | 0     | RS5   | RS4   | RS3   | RS2   | RS1   | RS0   | (V9938 et 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 spécifié s'écrit sur le port d'entrée/sortie 9Bh. En mode auto-incrémentation (AII à 0), chaque donnée écrite sur le port d'entrée/sortie 9Bh provoque une incrémentation du numéro de registre contenu dans le registre 17. Cela permet d'envoyer des données dans une série de registre.

Voir le paragraphe 7.4 « Comment accéder au VDP » rubrique « [Ecrire dans un registre de contrôle](#)», page 266, pour plus de précisions.

Note : Le système sauvegarde la valeur écrite dans ce registre dans la variable RG17SAV (0FFF0h).

## Registres d'affichage et interruption ligne (18 à 23)

|                      |       |       |       |       |       |       |       |       |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
| <b>Registre 18 :</b> | V3    | V2    | V1    | V0    | H3    | H2    | H1    | H0    | (V9938 et V9958) |

Ce registre permet de décaler l'emplacement de l'avant-plan.

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.

Notes :

- Veillez à que le bit CE du registre 2 de lecture soit à 0 avant d'accéder au registre 18 lorsque l'affichage de l'écran ou des Sprite est désactivé.
- Le système sauvegarde la valeur écrite dans ce registre dans la variable RG18SAV (0FFF1h) .

|                      |       |       |       |       |       |       |       |       |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
| <b>Registre 19 :</b> | IL7   | IL6   | IL5   | IL4   | IL3   | IL2   | IL1   | IL0   | (V9938 et 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.

Note : Le système sauvegarde la valeur qu'il écrit dans ce registre dans la variable RG19SAV (0FFF2h) .

|                      |       |       |       |       |       |       |       |       |         |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|---------|
|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |         |
| <b>Registre 20 :</b> | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | (V9938) |
| <b>Registre 21 :</b> | 0     | 0     | 1     | 1     | 1     | 0     | 1     | 1     | (V9938) |
| <b>Registre 22 :</b> | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 1     | (V9938) |

La valeur de ces 3 registres ne doivent pas être modifié. Elles indiquent la fréquence du signal « Colorburst » prévu pour une sortie vidéo composite. Exception à la règle, on peut les mettre à 0 pour couper la sortie composite. Il suffit de remettre les valeurs initiales pour rétablir la sortie composite.

Notes :

- Beaucoup de MSX2 n'utilisent pas ce signal car ils utilisent un dispositif externe pour faire la sortie composite à partir du signal RGB. Le V9958 n'a pas de signal « Colorburst ».
- Le système sauvegarde les valeurs qu'il écrit dans ces registres dans la variable REG20SAV (0FFF3h), REG21SAV (0FFF4h) et REG22SAV (0FFF5h).



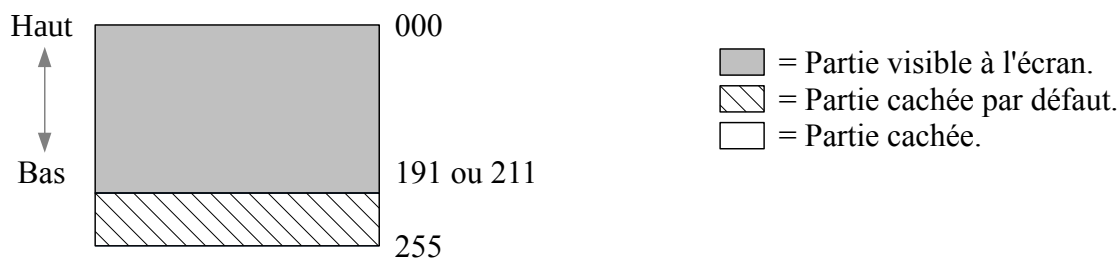
|                      |       |       |       |       |       |       |       |       |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
| <b>Registre 23 :</b> | DO7   | DO6   | DO5   | DO4   | DO3   | DO2   | DO1   | DO0   | (V9938 et V9958) |

Ce registre permet de décaler verticalement l’affichage de l’avant-plan. La hauteur de l’avant-plan est de 256 lignes (0 à 255). La partie visible peut être de 192 ou 212 lignes selon le bit LN (bit 7) du registre 9 de contrôle.

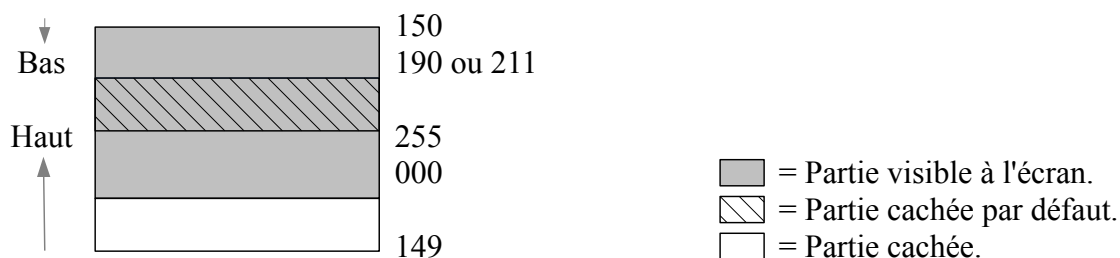
D0 à D7 = Défini la ligne à laquelle commence l’affichage de l’avant-plan. (0 à l’initialisation)

Note : Le système sauvegarde la valeur qu’il écrit dans ce registre dans la variable REG23SAV (0FFF6h).

#### Affichage d'une page sans décalage :



#### Exemple avec 150 dans le registre 23 :



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 :

```

EXTROM equ 0015Fh
CHGMODS 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
org 0c000h
DEBUT:
    ld a,8
    ld ix,CHGMOD
    call EXTROM ; SCREEN 8
;
    ld a,0ah
    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,(0007h) ; C = Numéro du port E/S correspondant
    ld c,a ; au port 1 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,EXIT
      call DOWN
      call BREAKX ; Test du CTRL+STOP
      jr c,EXIT ; Sortie du programme si CTRL+STOP
      jp LOOP
;
EXIT: 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 |                           |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|---------------------------|
| <b>Registre 26 :</b> | 0     | 0     | HO8   | HO7   | HO6   | HO5   | HO4   | HO3   | ( <a href="#">V9958</a> ) |
| <b>Registre 27 :</b> | 0     | 0     | 0     | 0     | 0     | HO2   | HO1   | HO0   | ( <a href="#">V9958</a> ) |

Ces registres permettent de décaler horizontalement l’affichage de l’avant-plan.

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)

Note : Le système sauvegarde les valeurs écrites dans ces registres dans les variables REG26SAV (0FFFBh) et REG27SAV (0FFFCh).

## Registres des commandes graphiques internes du VDP (32 à 46)

Les quinze derniers registres du VDP sont utilisés pour l'exécution des commandes graphiques. Voir le paragraphe 7.7 « [Les commandes internes du VDP](#) », page 294, pour plus de précisions.

|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| <b>Registre 32 :</b> | SX7   | SX6   | SX5   | SX4   | SX3   | SX2   | SX1   | SX0   | (V9938 et V9958) |
| <b>Registre 33 :</b> | 0     | 0     | 0     | 0     | 0     | 0     | 0     | SX8   | (V9938 et V9958) |

SX0 à SX8 = Codent l'abscisse du pixel d'origine ( $0 \leq X \leq 511$ )

|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| <b>Registre 34 :</b> | SY7   | SY6   | SY5   | SY4   | SY3   | SY2   | SY1   | SY0   | (V9938 et V9958) |
| <b>Registre 35 :</b> | 0     | 0     | 0     | 0     | 0     | 0     | SY9   | SY8   | (V9938 et V9958) |

SY0 à SY9 = Codent l'ordonnée du pixel d'origine ( $0 \leq Y \leq 1023$ )

|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| <b>Registre 36 :</b> | DX7   | DX6   | DX5   | DX4   | DX3   | DX2   | DX1   | DX0   | (V9938 et V9958) |
| <b>Registre 37 :</b> | 0     | 0     | 0     | 0     | 0     | 0     | 0     | DX8   | (V9938 et V9958) |

DX0 à DX8 = Abscisse du premier pixel de destination ( $0 \leq X \leq 511$ )

|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| <b>Registre 38 :</b> | DY7   | DY6   | DY5   | DY4   | DY3   | DY2   | DY1   | DY0   | (V9938 et V9958) |
| <b>Registre 39 :</b> | 0     | 0     | 0     | 0     | 0     | 0     | DY9   | DY8   | (V9938 et V9958) |

DY0 à DY9 = Codent l'ordonnée du pixel de destination ( $0 \leq Y \leq 1023$ )

|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| <b>Registre 40 :</b> | NX7   | NX6   | NX5   | NX4   | NX3   | NX2   | NX1   | NX0   | (V9938 et V9958) |
| <b>Registre 41 :</b> | 0     | 0     | 0     | 0     | 0     | 0     | 0     | NX8   | (V9938 et V9958) |

NX0 à NX8 = Déplacement horizontal ( $0 \leq NX \leq 511$ ).

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 |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| <b>Registre 42 :</b> | NY7   | NY6   | NY5   | NY4   | NY3   | NY2   | NY1   | NY0   | (V9938 et V9958) |
| <b>Registre 43 :</b> | 0     | 0     | 0     | 0     | 0     | 0     | NY9   | NY8   | (V9938 et V9958) |

NY0 à NY9 = Déplacement vertical ( $0 \leq NY \leq 1023$ ).

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 |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| <b>Registre 44 :</b> | CL7   | CL6   | CL5   | CL4   | CL3   | CL2   | CL1   | CL0   | (V9938 et V9958) |

CL0 à CL7 = Codent la couleur.

|                      | bit 7 | bit 6          | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
|----------------------|-------|----------------|-------|-------|-------|-------|-------|-------|------------------|
| <b>Registre 45 :</b> | 0     | <del>MXC</del> | MXD   | MXS   | DIY   | DIX   | EQ    | MAJ   | (V9938 et V9958) |

Registre de données pour les commandes du VDP. Nous verrons ces bits en détail dans le chapitre 7 « [Les commandes internes du V9938 et V9958](#) ».

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 |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| <b>Registre 46 :</b> | CM3   | CM2   | CM1   | CM0   | LO3   | LO2   | LO1   | LO0   | (V9938 et V9958) |

LO0 à LO3 = Définissent le code de l'opérateur logique à utiliser.

CM0 à CM3 = Définissent la commande du VDP à exécuter. ([Voir le chapitre qui suit](#))

## 7.7 Les commandes internes du VDP

Les VDP V9938 et V9958 disposent d'un jeu de commandes qui permettent de copier une zone de mémoire vers une autre, de tracer quelques formes simples, etc dans les graphiques SCREEN 5 à 12. 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 graphiques distinctes plus une commande d'arrêt (« STOP »).

Voici un tableau récapitulatif :

| Mnémonique | Code | Source | Destination | Signification                         |
|------------|------|--------|-------------|---------------------------------------|
| HMMC       | 0F0h | Z80    | VRAM        | High speed Move to Memory from CPU    |
| YMMM       | 0E0h | VRAM   | VRAM        | Y Move to Memory from Memory          |
| HMMM       | 0D0h | VRAM   | VRAM        | High speed Move to memory from memory |
| HMMV       | 0C0h | VDP    | VRAM        | High speed Move to Memory from VDP    |
| LMMC       | 0Bxh | Z80    | VRAM        | Logical Move to Memory from CPU       |
| LMCM       | 0Axh | VRAM   | Z80         | Logical Move to CPU from Memory       |
| LMMM       | 09xh | VRAM   | VRAM        | Logical Move to Memory from memory    |
| LMMV       | 08xh | VDP    | VRAM        | Logical Move to Memory from VDP       |
| LINE       | 07xh | VDP    | VRAM        | LINE                                  |
| SRCH       | 060h | VDP    | VRAM        | SeaRCH a border color                 |
| PSET       | 05xh | VDP    | VRAM        | Point SET                             |
| POINT      | 040h | VRAM   | VDP         | Is POINT set ?                        |
| STOP       | 00h  | -      | -           | STOP                                  |

Notes :

- Il est nécessaire de vérifier que le bit CE du [registre 2 de statut](#) soit à 0 avant d'exécuter une de ces commandes pour s'assurer qu'une commande ne soit pas en cours d'exécution.
- Les commandes sont aussi utilisables dans les modes d'écran 0 à 4 avec le V9958. (Voir le bit CMD du [registre 25](#))

A partir du SCREEN 5, le VDP peut travailler aussi en coordonnées X et Y. Dans ce cas, il n'y a pas de notions d'adresse mémoire. Tout se passe comme si le VDP opérait sur un écran géant de 256x1024 (ou 512x1024) dont le système gère ça sous forme de page. La partie visible ne serait alors qu'une des pages indiquées dans le schéma suivant :

| SCREEN 5                  | VRAM   | SCREEN 6                  |
|---------------------------|--------|---------------------------|
| 0, 0 255, 0<br>page 0     | 00000H | 0, 0 511, 0<br>page 0     |
| 0, 255 255, 255           | 05FFFH | 0, 255 511, 255           |
| 0, 256 255, 256<br>page 1 | 08000H | 0, 256 511, 256<br>page 1 |
| 0, 511 255, 511           | 0FFFFH | 0, 511 511, 511           |
| 0, 512 255, 512<br>page 2 | 10000H | 0, 512 511, 512<br>page 2 |
| 0, 767 255, 767           | 17FFFH | 0, 767 511, 767           |
| 0, 768 255, 768<br>page 3 | 18000H | 0, 768 511, 768<br>page 3 |
| 0, 1023 255, 1023         | 1FFFFH | 0, 1023 511, 1023         |

| SCREEN 7                  | VRAM   | SCREEN 8 à 12             |
|---------------------------|--------|---------------------------|
| 0, 0 511, 0<br>page 0     | 00000H | 0, 0 255, 0<br>page 0     |
| 0, 255 511, 255           | 0FFFFH | 0, 255 255, 255           |
| 0, 256 511, 256<br>page 1 | 10000H | 0, 256 255, 256<br>page 1 |
| 0, 511 511, 511           | 1FFFFH | 0, 511 255, 511           |

Note : La VRAM est configurée physiquement de façon à ce que les octets paires contiennent les pages paires et les octets impaires les pages impaires. Il faut en tenir compte si vous devez changer de mode sans initialiser la VRAM ou lors de l'utilisation de la VRAM étendue. C'est la raison qui fait qu'un MSX2 n'ayant que 64Ko de VRAM ne peut pas afficher d'écran en mode SCREEN 7 ni 8.

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.

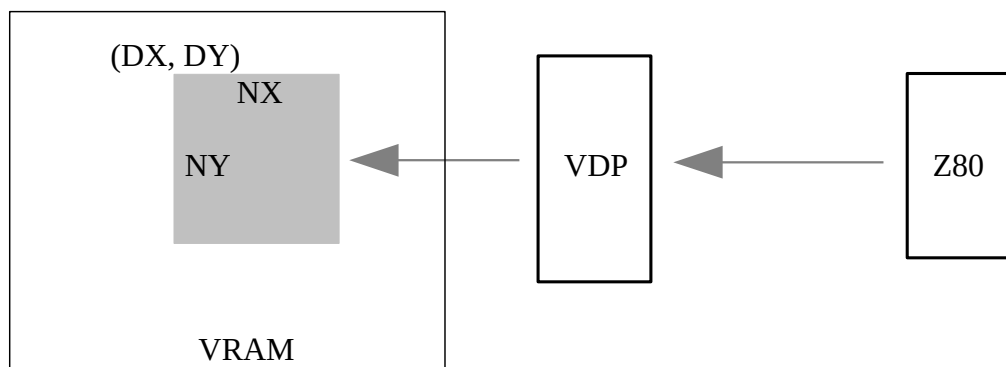
| Opérateur | Code | Fonction                                                        |
|-----------|------|-----------------------------------------------------------------|
| IMP       | 0000 | La couleur affichée sera remplacée par la nouvelle.             |
| AND       | 0001 | Couleur résultante = (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 | Les bits de la couleur affichée seront inversés.                |

| 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 de 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 de 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 DX0 est ignoré.

En SCREEN 6, les bits DX0 et DX1 sont ignorés.

DY0 à DY9 = Ordonnée du premier pixel de destination.

|               |     |     |     |     |     |     |     |     |                        |
|---------------|-----|-----|-----|-----|-----|-----|-----|-----|------------------------|
| 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 (largeur).

NY0 à NY9 = Nombre de pixels à placer dans la direction verticale. (hauteur)

Registre 44 :

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| CR7 | CR6 | CR5 | CR4 | CR3 | CR2 | CR1 | CR0 |
|-----|-----|-----|-----|-----|-----|-----|-----|

donnée

En SCREEN 5 et 7 :

CR0 à CR3 = Couleur du pixel d'abscisse impair.

CR4 à CR7 = Couleur du pixel d'abscisse pair.

En 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.

En SCREEN 8 à 12 :

CR0 à CR7 = Couleur du pixel.

Registre 45 :

|   |   |     |   |     |     |   |   |
|---|---|-----|---|-----|-----|---|---|
| 0 | - | MXD | - | DIY | DIX | - | - |
|---|---|-----|---|-----|-----|---|---|

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 la VRAM principale ; 1 pour la VRAM étendue des MSX ayant 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 l'étape 3. Si, par contre, ce bit est à 1, continuer en 4.
4. Envoyer l'octet suivant à mettre en VRAM dans le registre 44 et reprendre toute l'opération à l'étape 3.

Une fois la commande terminée, les registres de commande se trouveront dans l'état suivant :

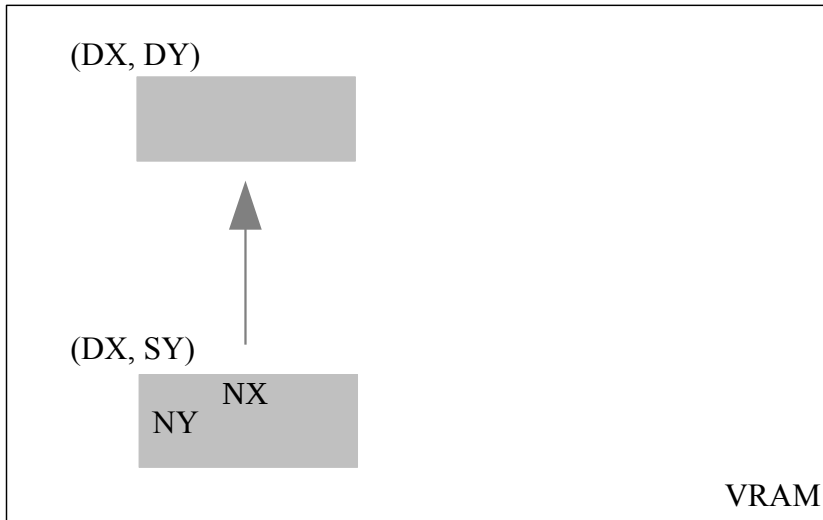
|           |           |           |           |           |           |          |          |
|-----------|-----------|-----------|-----------|-----------|-----------|----------|----------|
| 32~33(SX) | 34~35(SY) | 36~37(DX) | 38~39(DY) | 40~41(NX) | 42~43(NY) | 44       | 45       |
| inchangé  | inchangé  | inchangé  | *         | inchangé  | **        | inchangé | inchangé |

(\*) Prend la dernière valeur en fin de commande.

(\*\*) Prend la valeur de la longueur entre point départ et la fin de l'écran lorsque celui-ci est atteint.

## YMMM (Y Move to Memory from Memory)

Schéma :



Fonction : YMMM effectue 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<br>(0 à 1023) |
| Registre 35 : | 0     | 0     | 0     | 0     | 0     | 0     | SY9   | SY8   |                               |

SY0 à SY9 = Ordonnée du pixel d'origine.

En SCREEN 5 et 7, SX0 est ignoré.

En SCREEN 6, SX0 et SX1 sont ignorés.

|               |     |     |     |     |     |     |     |     |                                          |
|---------------|-----|-----|-----|-----|-----|-----|-----|-----|------------------------------------------|
| Registre 36 : | DX7 | DX6 | DX5 | DX4 | DX3 | DX2 | DX1 | DX0 | abscisse source/destination<br>(0 à 511) |
| Registre 37 : | 0   | 0   | 0   | 0   | 0   | 0   | 0   | DX8 |                                          |
| Registre 38 : | DY7 | DY6 | DY5 | DY4 | DY3 | DY2 | DY1 | DY0 | ordonnée de destination<br>(0 à 1023)    |
| Registre 39 : | 0   | 0   | 0   | 0   | 0   | 0   | DY9 | DY8 |                                          |

DX0 à DX8 = Abscisse du premier pixel source et de destination.

En SCREEN 5 et 7, DX0 est ignoré.

En SCREEN 6, DX0 et DX1 sont ignorés.

DY0 à DY9 = Ordonnée du pixel de destination.

|               | 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   | 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 copier dans la direction horizontale.

NY0 à NY9 = Nombre de pixels à copier dans la direction verticale.

|               |   |   |     |   |     |     |   |   |            |
|---------------|---|---|-----|---|-----|-----|---|---|------------|
| Registre 45 : | 0 | - | MXD | - | DIY | DIX | - | - | 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 la VRAM principale ; 1 pour 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. Vérifiez que ce bit soit à 0 avant d'exécuter la commande suivante ou de changer de paramètre.

Une fois la commande terminée, les registres de commande se trouveront dans l'état suivant :

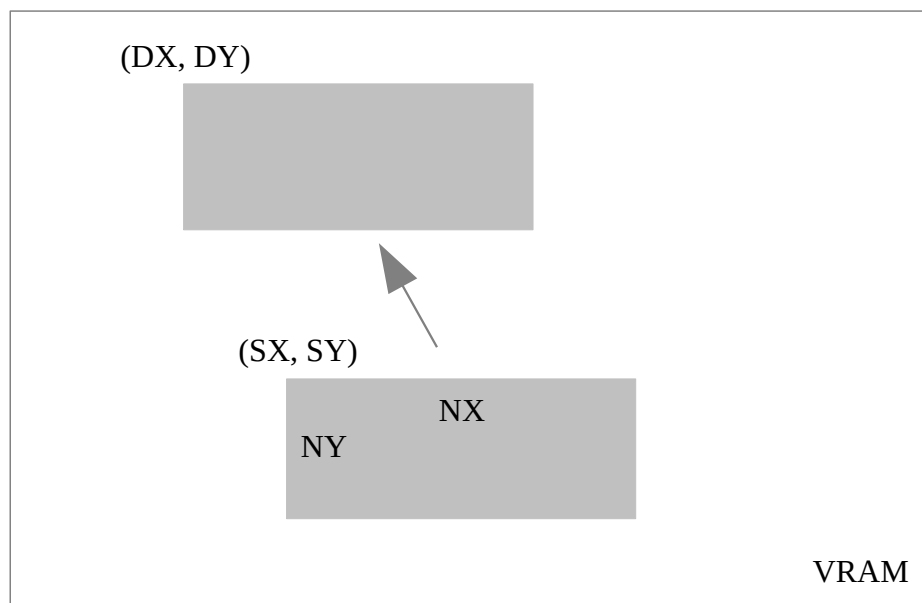
|           |            |            |            |            |            |          |          |
|-----------|------------|------------|------------|------------|------------|----------|----------|
| 32~33(SX) | 34~35 (SY) | 36~37 (DX) | 38~39 (DY) | 40~41 (NX) | 42~43 (NY) | 44       | 45       |
| inchangé  | *          | inchangé   | *          | inchangé   | **         | inchangé | inchangé |

(\*) Prend la dernière valeur en fin de commande.

(\*\*) Prend la valeur de la longueur entre point départ et la fin de l'écran lorsque celui-ci est atteint.

## HMMM (High Speed Move to Memory from Memory)

Schéma :



Fonction : HMMM effectue 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<br>(0 à 256/511) |
| Registre 33 : | 0     | 0     | 0     | 0     | 0     | 0     | 0     | SX8   |                                  |
| Registre 34 : | SY7   | SY6   | SY5   | SY4   | SY3   | SY2   | SY1   | SY0   | ordonnée source<br>(0 à 1023)    |
| Registre 35 : | 0     | 0     | 0     | 0     | 0     | 0     | SY9   | SY8   |                                  |

SX0 à SX8 = Abscisse du pixel d'origine.

En SCREEN 5 et 7, SX0 est ignoré.

En SCREEN 6, SX0 et SX1 sont ignorés.

SY0 à SY9 = Ordonnée du pixel d'origine.

|               | 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 de destination<br>(0 à 511)  |
| Registre 37 : | 0     | 0     | 0     | 0     | 0     | 0     | 0     | DX8   |                                       |
| Registre 38 : | DY7   | DY6   | DY5   | DY4   | DY3   | DY2   | DY1   | DY0   | ordonnée de destination<br>(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, DX0 est ignoré.

En SCREEN 6, NX0 et DX1 sont ignorés.

DY0 à DY9 = Ordonnée du pixel de destination.

|               |     |     |     |     |     |     |     |     |                                     |
|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-------------------------------------|
| Registre 40 : | NX7 | NX6 | NX5 | NX4 | NX3 | NX2 | NX1 | NX0 | déplacement horizontal<br>(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<br>(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 | - | MXD | MXS | DIY | DIX | - | - | 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 la VRAM principale ; 1 pour la VRAM étendue des MSX ayant 192 Ko de VRAM.

MXD = Mémoire vidéo de destination. 0 pour la VRAM principale ; 1 pour la VRAM étendue des MSX ayant 192 Ko de VRAM.

- Écrire 11010000B (0D0h) dans le registre 46 pour exécuter la commande.
- 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 la commande terminée, les registres de commande du VDP se trouveront dans l'état suivant :

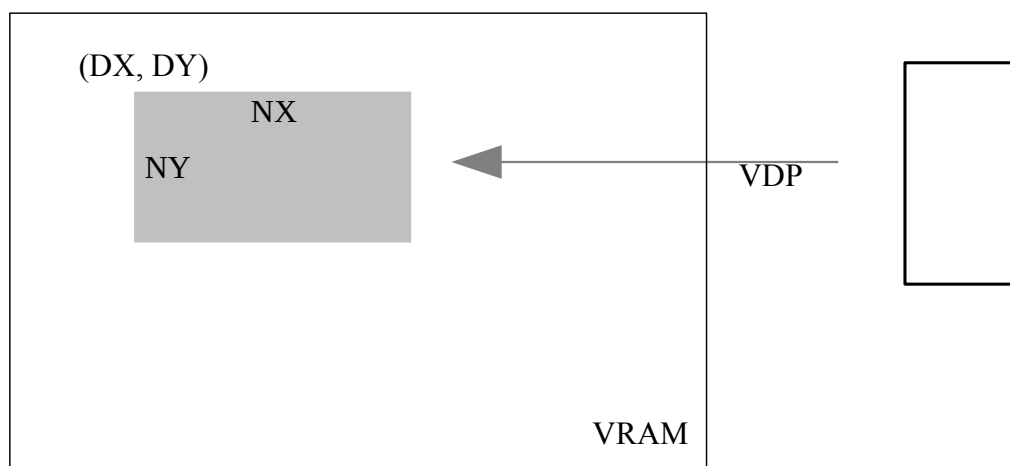
|           |            |            |            |            |            |          |          |
|-----------|------------|------------|------------|------------|------------|----------|----------|
| 32~33(SX) | 34~35 (SY) | 36~37 (DX) | 38~39 (DY) | 40~41 (NX) | 42~43 (NY) | 44       | 45       |
| inchangé  | *          | inchangé   | *          | inchangé   | **         | inchangé | inchangé |

(\*) Prend la dernière valeur en fin de commande.

(\*\*) Prend la valeur de la longueur entre point départ et la fin de l'écran lorsque celui-ci est atteint.

## 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, DX0 est ignoré.

En SCREEN 6, DX0 et DX1 sont ignorés.

DY0 à DY9 = Ordonnée du pixel origine.

|               | 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   | 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.

NY0 à NY9 = Nombre de pixels à placer dans la direction verticale.

|               |     |     |     |     |     |     |     |     |        |
|---------------|-----|-----|-----|-----|-----|-----|-----|-----|--------|
| Registre 44 : | CR7 | CR6 | CR5 | CR4 | CR3 | CR2 | CR1 | CR0 | donnée |
|---------------|-----|-----|-----|-----|-----|-----|-----|-----|--------|

En SCREEN 5 et 7 :

CR0 à CR3 = Couleur du pixel d'abscisse impair.

CR4 à CR7 = Couleur du pixel d'abscisse pair.

En 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.

En SCREEN 8 à 12 :

CR0 à CR7 = Couleur du pixel.

|               |   |   |     |   |     |     |   |   |            |
|---------------|---|---|-----|---|-----|-----|---|---|------------|
| Registre 45 : | 0 | - | MXD | - | DIY | DIX | - | - | 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 la VRAM principale ; 1 pour la VRAM étendue des MSX ayant 192 Ko de VRAM.

- Écrire 11000000B (0C0h) dans le registre 46 pour exécuter la commande.
- 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 la commande terminée, les registres de commande se trouveront dans l'état suivant :

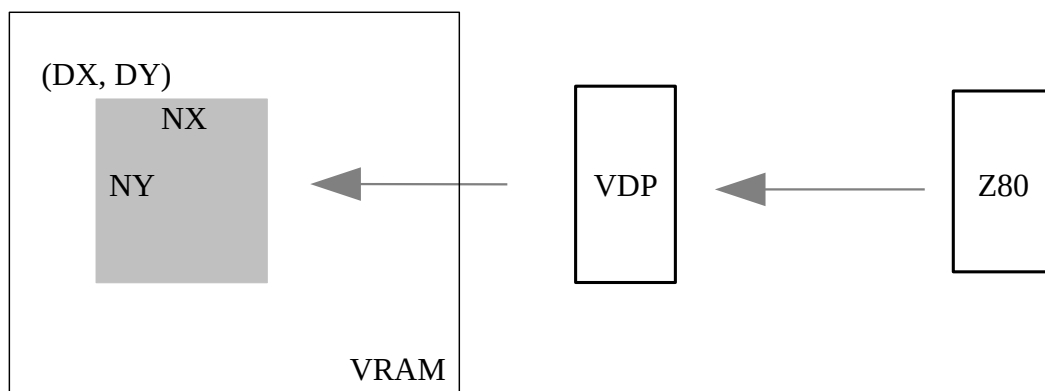
|           |            |            |            |            |            |          |          |
|-----------|------------|------------|------------|------------|------------|----------|----------|
| 32~33(SX) | 34~35 (SY) | 36~37 (DX) | 38~39 (DY) | 40~41 (NX) | 42~43 (NY) | 44       | 45       |
| inchangé  | inchangé   | inchangé   | *          | inchangé   | **         | inchangé | inchangé |

(\*) Prend la dernière valeur en fin de commande.

(\*\*) Prend la valeur de la longueur entre point départ et la fin de l'écran lorsque celui-ci est atteint.



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 VRAM peuvent être toutes différentes. De plus, la mémoire vidéo est définie par des coordonnées X et 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<br>(0 à 256/511) |
| Registre 33 : | 0     | 0     | 0     | 0     | 0     | 0     | 0     | SX8   |                                  |
| Registre 34 : | SY7   | SY6   | SY5   | SY4   | SY3   | SY2   | SY1   | SY0   | ordonnée source<br>(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<br>(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<br>(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.

|               |       |       |       |       |       |       |       |       |        |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|--------|
|               | bit 7 | bit 6 | bit 5 | bit 4 | bit 4 | bit 2 | bit 1 | bit 0 |        |
| Registre 44 : | CR7   | CR6   | CR5   | CR4   | CR3   | CR2   | CR1   | CR0   | donnée |

En SCREEN 5 et 7 : CR0 à CR3 = Couleur. CR4 à CR7 bits sont ignorés.

En SCREEN 6 : CR0 et CR1 = Couleur. CR2 à CR7 bits sont ignorés.

En SCREEN 8 : CR0 à CR7 = Couleur.

|               |   |   |     |   |     |     |   |   |            |
|---------------|---|---|-----|---|-----|-----|---|---|------------|
| 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 la VRAM principale ; 1 pour VRAM étendue des MSX ayant 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 : | 1 | 0 | 1 | 1 | LO3 | LO2 | LO1 | LO0 | commande+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.
4. Tester l'état du bit CE. Si celui-ci est à 1, alors le processus est terminé. Sinon, on passe à l'étape suivante.
5. 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.
6. 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 la commande terminée, les registres de commande du VDP se trouveront dans l'état suivant :

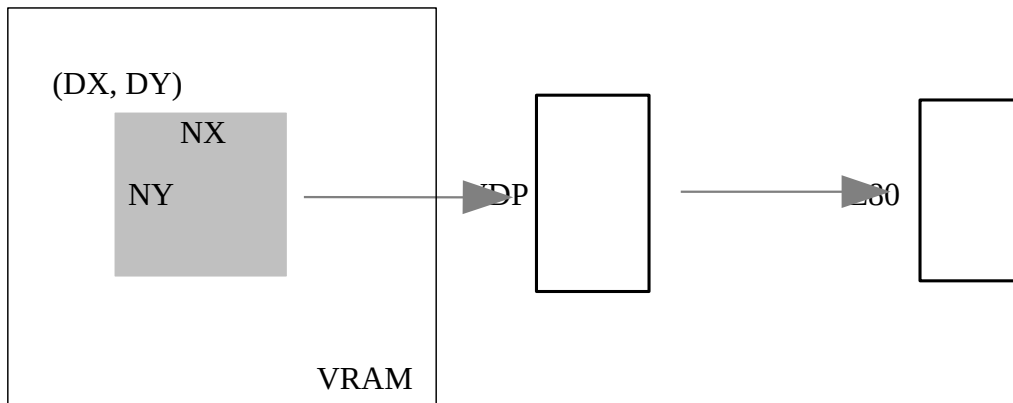
|           |            |            |            |            |            |          |          |
|-----------|------------|------------|------------|------------|------------|----------|----------|
| 32~33(SX) | 34~35 (SY) | 36~37 (DX) | 38~39 (DY) | 40~41 (NX) | 42~43 (NY) | 44       | 45       |
| inchangé  | inchangé   | inchangé   | *          | inchangé   | **         | inchangé | inchangé |

(\*) Prend la dernière valeur en fin de commande.

(\*\*) Prend la valeur de la longueur entre point départ et la fin de l'écran lorsque celui-ci est atteint.

## 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<br>(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<br>(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<br>(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<br>(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 | - | - | MXS | DIY | DIX | - | - |
|---|---|---|-----|-----|-----|---|---|

 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 la VRAM principale ; 1 pour la VRAM étendue des MSX ayant 192 Ko de VRAM.

2. Écrivez la valeur comme suit dans le registre 46 pour exécuter la commande.

Registre 46 : 

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | - | - | - | - |
|---|---|---|---|---|---|---|---|

 commande

3. Lire le bit TR du registre 2 de statut, si celui-ci se trouve à 1, c'est que le VDP n'a pas encore transféré toutes les données, vous pouvez donc lire le registre de statut 7 pour obtenir la donnée suivante issue de la mémoire vidéo. Si par contre, bit TR passera à 0 une fois que toutes les données auront été lues.

La couleur dans le registre de statut 7 prend la forme suivante.

4. Tester l'état du bit CE, si celui-ci est à 0, alors le processus est terminé. Dans le cas contraire (CE à 1), on recommence à l'étape 2.

Notes :

- TR doit être à 0 pour exécuter cette commande. (Lire le registre 7 de statut si ce n'est pas le cas)
- 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 la commande terminée, les registres de commande du VDP se trouveront dans l'état suivant :

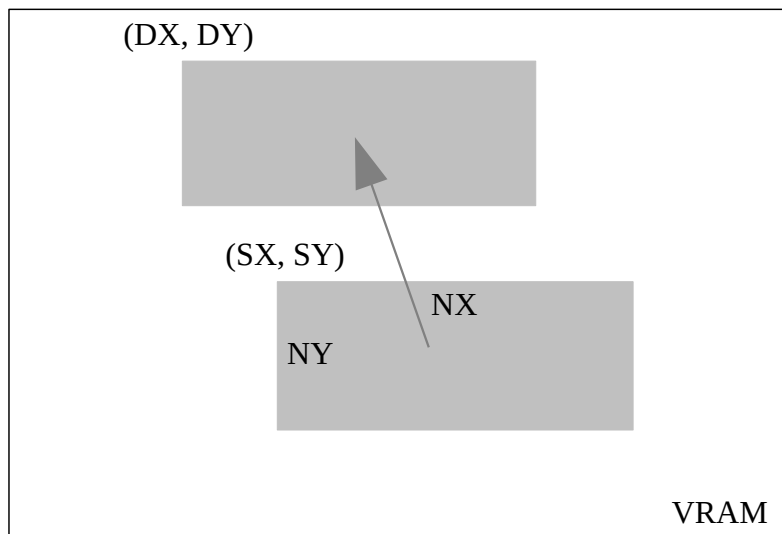
|           |            |            |            |            |            |          |          |
|-----------|------------|------------|------------|------------|------------|----------|----------|
| 32~33(SX) | 34~35 (SY) | 36~37 (DX) | 38~39 (DY) | 40~41 (NX) | 42~43 (NY) | 44       | 45       |
| inchangé  | *          | inchangé   | inchangé   | inchangé   | **         | inchangé | inchangé |

(\*) Prend la dernière valeur en fin de commande.

(\*\*) Prend la valeur de la longueur entre point départ et la fin de l'écran lorsque celui-ci est atteint.

## LMMM (Logical Move to Memory from Memory)

Schéma :



Fonction : LMMM fait la copie d'une zone de mémoire vidéo vers 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 |
| Registre 33 : | 0     | 0     | 0     | 0     | 0     | 0     | 0     | SX8   | (0 à 511)       |
| Registre 34 : | SY7   | SY6   | SY5   | SY4   | SY3   | SY2   | SY1   | SY0   | ordonnée source |
| Registre 35 : | 0     | 0     | 0     | 0     | 0     | 0     | SY9   | SY8   | (0 à 1023)      |

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 de 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 de destination |
| Registre 39 : | 0   | 0   | 0   | 0   | 0   | 0   | DY9 | DY8 | (0 à 1023)              |

DX0 à DX8 = Abscisse du premier pixel de destination.

DY0 à DY9 = Ordonnée du pixel de destination.

|               | bit 7 | bit 6 | bit 5 | bit 4 | bit 4 | bit 2 | bit 1 | bit 0 |                                     |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|-------------------------------------|
| Registre 40 : | NX7   | NX6   | NX5   | NX4   | NX3   | NX2   | NX1   | NX0   | déplacement horizontal<br>(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<br>(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 | - | MXD | MXS | DIY | DIX | - | - | 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 la VRAM principale ; 1 pour la VRAM étendue des MSX ayant 192 Ko de VRAM.

MXD = Mémoire vidéo de destination. 0 pour la VRAM principale ; 1 pour la VRAM étendue des MSX ayant 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 | commande+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 la commande terminée, les registres de commande se trouveront dans l'état suivant :

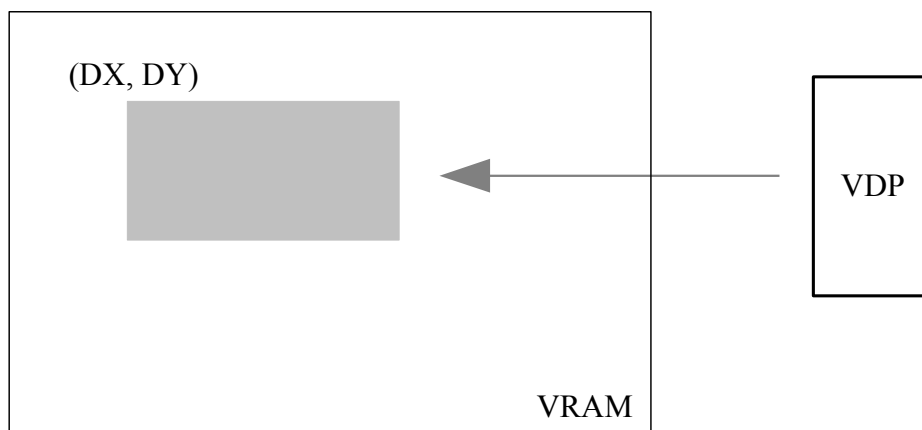
|           |            |            |            |            |            |          |          |
|-----------|------------|------------|------------|------------|------------|----------|----------|
| 32~33(SX) | 34~35 (SY) | 36~37 (DX) | 38~39 (DY) | 40~41 (NX) | 42~43 (NY) | 44       | 45       |
| inchangé  | *          | inchangé   | *          | inchangé   | **         | inchangé | inchangé |

(\*) Prend la dernière valeur en fin de commande.

(\*\*) Prend la valeur de la longueur entre point départ et la fin de l'écran lorsque celui-ci est atteint.

## 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 36 : | DX7   | DX6   | DX5   | DX4   | DX3   | DX2   | DX1   | DX0   | abscisse de destination<br>(0 à 511)  |
| Registre 37 : | 0     | 0     | 0     | 0     | 0     | 0     | 0     | DX8   |                                       |
| Registre 38 : | DY7   | DY6   | DY5   | DY4   | DY3   | DY2   | DY1   | DY0   | ordonnée de destination<br>(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<br>(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<br>(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.

|               |       |       |       |       |       |       |       |       |        |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|--------|
|               | bit 7 | bit 6 | bit 5 | bit 4 | bit 4 | bit 2 | bit 1 | bit 0 |        |
| Registre 44 : | CR7   | CR6   | CR5   | CR4   | CR3   | CR2   | CR1   | CR0   | donnée |

En SCREEN 5 et 7 : CR0 à CR3 = Couleur. CR4 à CR7 bits sont ignorés.

En SCREEN 6 : CR0 et CR1 = Couleur. CR2 à CR7 bits sont ignorés.

En SCREEN 8 : CR0 à CR7 = Couleur.

|               |   |   |     |   |     |     |   |   |            |
|---------------|---|---|-----|---|-----|-----|---|---|------------|
| Registre 45 : | 0 | - | MXD | - | DIY | DIX | - | - | 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 la VRAM principale ; 1 pour la VRAM étendue des MSX ayant 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 | commande+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 la commande terminée, les registres de commande se trouveront dans l'état suivant :

|           |            |            |            |            |            |          |          |
|-----------|------------|------------|------------|------------|------------|----------|----------|
| 32~33(SX) | 34~35 (SY) | 36~37 (DX) | 38~39 (DY) | 40~41 (NX) | 42~43 (NY) | 44       | 45       |
| inchangé  | inchangé   | inchangé   | *          | inchangé   | *          | inchangé | inchangé |

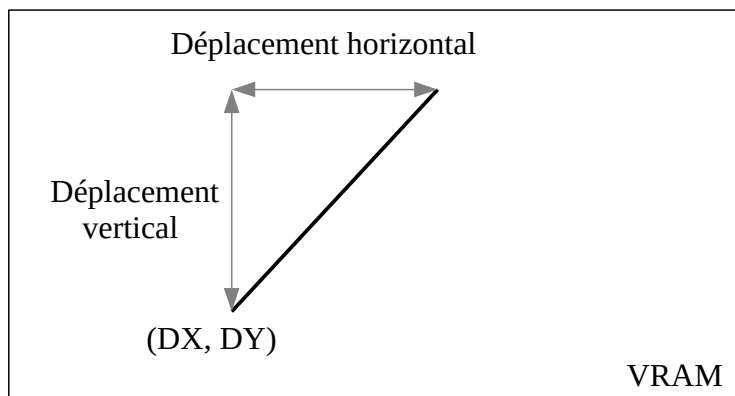
(\*) Prend la dernière valeur en fin de commande.

(\*\*) Prend la valeur de la longueur entre point départ et la fin de l'écran lorsque celui-ci est atteint.



## 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 de destination<br>(0 à 511)  |
| Registre 37 : | 0     | 0     | 0     | 0     | 0     | 0     | 0     | DX8   |                                       |
| Registre 38 : | DY7   | DY6   | DY5   | DY4   | DY3   | DY2   | DY1   | DY0   | ordonnée de destination<br>(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<br>(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<br>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.

|               |       |       |       |       |       |       |       |       |        |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|--------|
|               | bit 7 | bit 6 | bit 5 | bit 4 | bit 4 | bit 2 | bit 1 | bit 0 |        |
| Registre 44 : | CR7   | CR6   | CR5   | CR4   | CR3   | CR2   | CR1   | CR0   | donnée |

En SCREEN 5 et 7 : CR0 à CR3 = Couleur. CR4 à CR7 bits sont ignorés.

En SCREEN 6 : CR0 et CR1 = Couleur. CR2 à CR7 bits sont ignorés.

En SCREEN 8 : CR0 à CR7 = Couleur.

|               |   |   |     |   |     |     |   |     |            |
|---------------|---|---|-----|---|-----|-----|---|-----|------------|
| Registre 45 : | 0 | - | MXD | - | DIY | DIX | - | 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 la VRAM étendue sur les MSX ayant 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 : | 0 | 1 | 1 | 1 | LO3 | LO2 | LO1 | LO0 | commande+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 |

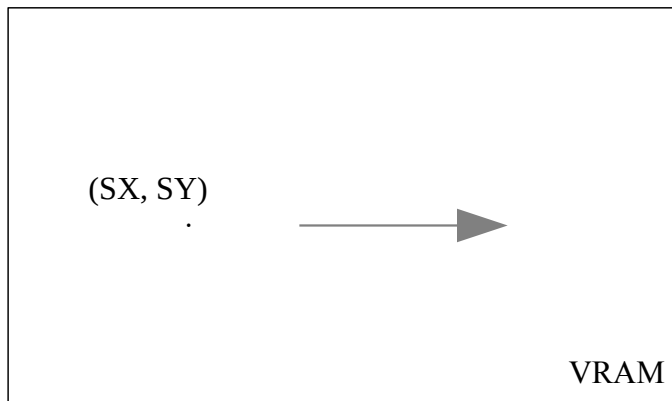
- Le bit CE du registre de statut 2 restera à 1 tant que le processus n'est pas terminé. Vérifier que ce bit soit à 0 avant d'exécuter la commande suivante ou de changer de paramètre.

Une fois la commande terminée, les registres de commande du VDP se trouveront dans l'état suivant :

|           |            |            |            |            |            |          |          |
|-----------|------------|------------|------------|------------|------------|----------|----------|
| 32~33(SX) | 34~35 (SY) | 36~37 (DX) | 38~39 (DY) | 40~41 (NX) | 42~43 (NY) | 44       | 45       |
| inchangé  | inchangé   | inchangé   | *          | inchangé   | inchangé   | inchangé | inchangé |

(\*) Prend la dernière valeur en fin de commande.

Schéma :



Fonction : SRCH permet de rechercher une couleur de contour donnée sur une ligne horizontale. Cette instruction est particulièrement utile pour le 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<br>(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<br>(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 |
|---------------|-----|-----|-----|-----|-----|-----|-----|-----|--------|

En SCREEN 5 et 7 : CR0 à CR3 = Couleur. Les bits CR4 à CR7 doivent être mis à 0.

En SCREEN 6 : CR0 et CR1 = Couleur. Les bits CR2 à CR7 doivent être mis à 0.

En SCREEN 8 : CR0 à CR7 = Couleur.

Registre 45 : 

|   |   |     |   |   |     |    |   |
|---|---|-----|---|---|-----|----|---|
| 0 | - | MXD | - | - | DIX | EQ | - |
|---|---|-----|---|---|-----|----|---|

 paramètres

EQ = 0 pour que la recherche s'arrête lorsqu'une couleur est différente de celle de contour,

1 pour que la recherche s'arrête lorsqu'une couleur est la même que celle de contour.

DIX = Direction horizontale de la recherche à partir du pixel origine.

0 pour chercher vers la droite ; 1 pour chercher vers la gauche.

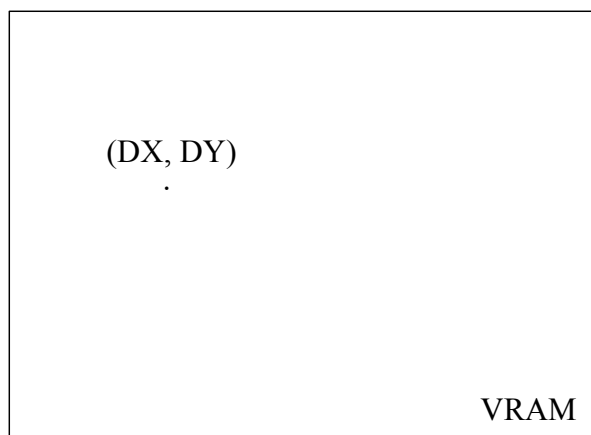
MXD = Mémoire vidéo de destination. 0 pour la VRAM principale ; 1 pour la VRAM étendue sur les MSX ayant 192 Ko de VRAM.

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 1, c'est que la commande s'est terminée sur la même couleur que celle de contour (bit EQ à 1) ou une couleur différente (bit EQ à 0). L'abscisse de ce pixel s'obtient alors en lisant le contenu des registres de statut 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. Si BD est à 0, c'est que la commande s'est terminée en fin de ligne.

Une fois la commande terminée, les registres de commande se trouveront dans l'état suivant :

|           |            |            |            |            |            |          |          |
|-----------|------------|------------|------------|------------|------------|----------|----------|
| 32~33(SX) | 34~35 (SY) | 36~37 (DX) | 38~39 (DY) | 40~41 (NX) | 42~43 (NY) | 44       | 45       |
| inchangé  | inchangé   | inchangé   | inchangé   | inchangé   | inchangé   | inchangé | inchangé |

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 |
| Registre 33 : | 0     | 0     | 0     | 0     | 0     | 0     | 0     | SX8   | (0 à 511)       |
| Registre 34 : | SY7   | SY6   | SY5   | SY4   | SY3   | SY2   | SY1   | SY0   | ordonnée source |
| Registre 35 : | 0     | 0     | 0     | 0     | 0     | 0     | SY9   | SY8   | (0 à 1023)      |

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 | couleur |
|---------------|-----|-----|-----|-----|-----|-----|-----|-----|---------|

En SCREEN 5 et 7 : CR0 à CR3 = Couleur. CR4 à CR7 bits sont ignorés.

En SCREEN 6 : CR0 et CR1 = Couleur. CR2 à CR7 bits sont ignorés.

En SCREEN 8 : CR0 à CR7 = Couleur.

|               |   |   |     |   |   |   |   |   |            |
|---------------|---|---|-----|---|---|---|---|---|------------|
| Registre 45 : | 0 | 0 | MXD | 0 | 0 | 0 | 0 | 0 | paramètres |
|---------------|---|---|-----|---|---|---|---|---|------------|

MXD = Mémoire de destination. 0 pour la VRAM principale ; 1 pour la VRAM étendue des MSX ayant 192 Ko de VRAM.

2. Écrire le code de la commande (5h) 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.

|               |       |       |       |       |       |       |       |       |                    |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|--------------------|
|               | bit 7 | bit 6 | bit 5 | bit 4 | bit 4 | bit 2 | bit 1 | bit 0 |                    |
| Registre 46 : | 0     | 1     | 0     | 1     | LO3   | LO2   | LO1   | LO0   | commande+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 |

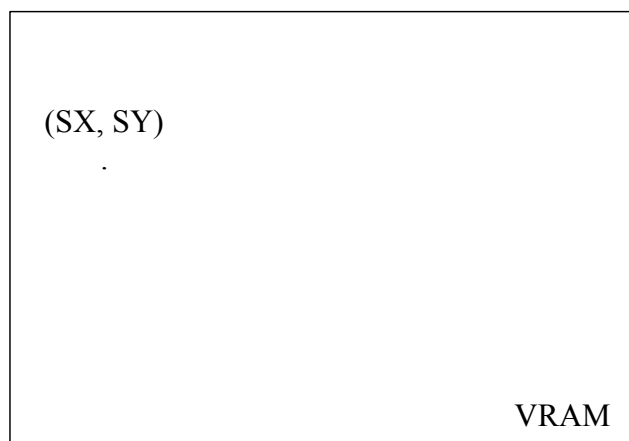
3. Le bit CE du registre 2 de statut restera à 1 tant que le processus n'est pas terminé. Vérifier que ce bit soit à 0 avant d'exécuter la commande suivante ou de changer de paramètre.

Une fois la commande terminée, les registres de commande du VDP se trouveront dans l'état suivant :

|           |            |            |            |            |            |          |          |
|-----------|------------|------------|------------|------------|------------|----------|----------|
| 32~33(SX) | 34~35 (SY) | 36~37 (DX) | 38~39 (DY) | 40~41 (NX) | 42~43 (NY) | 44       | 45       |
| inchangé  | inchangé   | inchangé   | inchangé   | inchangé   | inchangé   | inchangé | inchangé |

## 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 |
| Registre 33 : | 0     | 0     | 0     | 0     | 0     | 0     | 0     | SX8   | (0 à 511)       |
| Registre 34 : | SY7   | SY6   | SY5   | SY4   | SY3   | SY2   | SY1   | SY0   | ordonnée source |
| Registre 35 : | 0     | 0     | 0     | 0     | 0     | 0     | SY9   | SY8   | (0 à 1023)      |

SX0 à SX8 = Abscisse du pixel d'origine.

SY0 à SY9 = Ordonnée du pixel d'origine.

|               |   |   |   |     |   |   |   |   |            |
|---------------|---|---|---|-----|---|---|---|---|------------|
| Registre 45 : | - | - | - | MXS | - | - | - | - | paramètres |
|---------------|---|---|---|-----|---|---|---|---|------------|

MXS = Mémoire vidéo source. 0 pour la VRAM principale ; 1 pour la VRAM étendue des MSX ayant 192 Ko de VRAM.

2. Écrire le code de la commande 1000000B (040h) dans le registre 46 pour exécuter la commande.

|               | bit 7 | bit 6 | bit 5 | bit 4 | bit 4 | bit 2 | bit 1 | bit 0 |                    |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|--------------------|
| Registre 46 : | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0     | commande+opérateur |

3. Lire le bit CE du registre 2 de statut. Si il est à 1, alors l'instruction n'est pas terminée, recommencer l'étape 3. Lorsque le bit CE passe à 0, on peut poursuivre à l'étape 4.
4. Lire le registre de statut 7, il contient le code de la couleur du pixel sous la forme suivante.

|                        |       |       |       |       |       |       |       |       |        |
|------------------------|-------|-------|-------|-------|-------|-------|-------|-------|--------|
|                        | bit 7 | bit 6 | bit 5 | bit 4 | bit 4 | bit 2 | bit 1 | bit 0 |        |
| Registre de statut 7 : | CR7   | CR6   | CR5   | CR4   | CR3   | CR2   | CR1   | CR0   | donnée |

En SCREEN 5 et 7 : CR0 à CR3 = Couleur. CR4 à CR7 bits sont ignorés.

En SCREEN 6 : CR0 et CR1 = Couleur. CR2 à CR7 bits sont ignorés.

En SCREEN 8 : CR0 à CR7 = Couleur.

Une fois la commande terminée, les registres de commande du VDP se trouveront dans l'état suivant :

|           |            |            |            |            |            |             |          |
|-----------|------------|------------|------------|------------|------------|-------------|----------|
| 32~33(SX) | 34~35 (SY) | 36~37 (DX) | 38~39 (DY) | 40~41 (NX) | 42~43 (NY) | 44          | 45       |
| inchangé  | inchangé   | inchangé   | inchangé   | inchangé   | inchangé   | Couleur lue | inchangé |



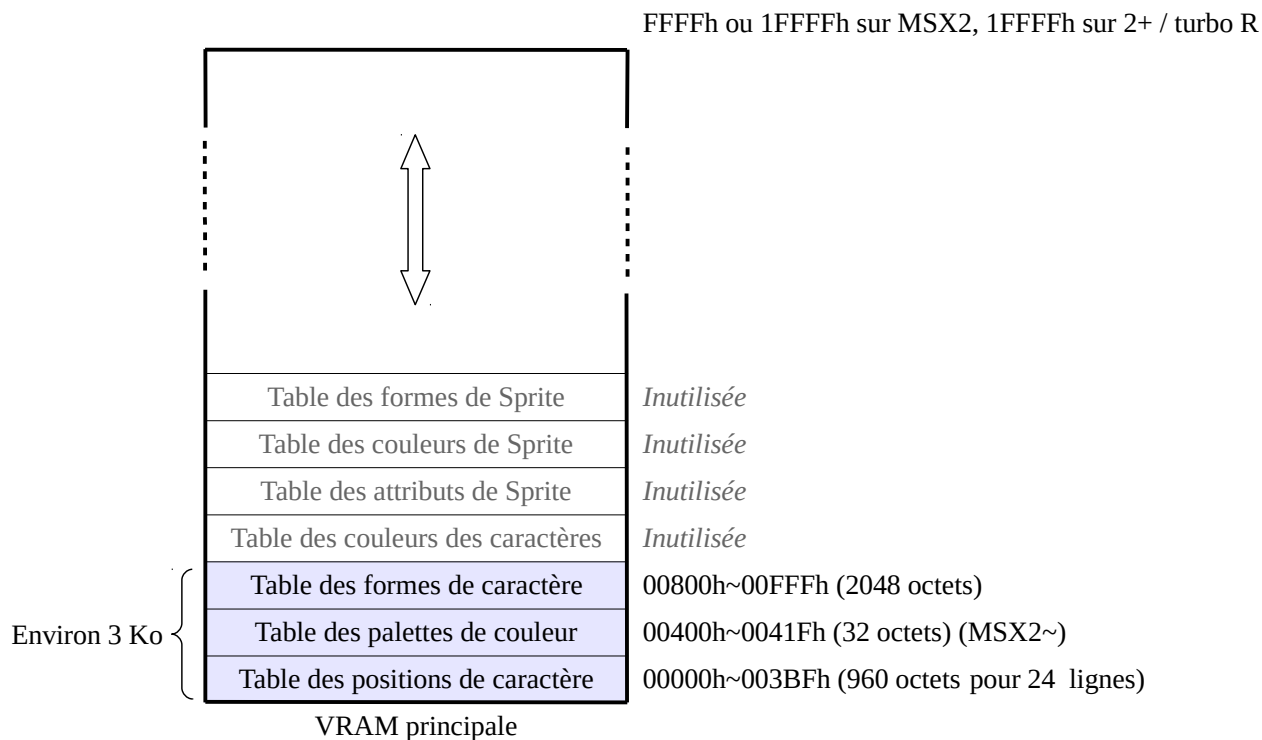
## 7.8 Les différents modes d'affichage

Nous allons voir en détail comment fonctionne les modes d'affichage de chaque VDP.

### SCREEN 0 à 40 colonnes (mode texte 1)

Résolution : 256 × 192 pixels.  
Type : Mode texte, 40 colonnes × 24 lignes, caractères 6 × 8.  
Couleurs : 2 sur 15 couleurs,  
2 sur 16 couleurs redéfinissables parmi 512 à partir du MSX2.  
Sprite : Inutilisés.  
VDP : Tous.

La carte de la VRAM par défaut se présente ainsi



Exécutons le programme Basic suivant pour voir comment fonctionne ce mode d'écran.

```
10 COLOR15,4: SCREEN 0: WIDTH 40
20 PRINT"HELLO!"
```

Le texte « HELLO! » s'affiche sur le coin tout en haut à gauche de l'écran ainsi :

|         |   |   |   |   |   |   |  |     |
|---------|---|---|---|---|---|---|--|-----|
| Ligne 0 | H | E | L | L | O | ! |  | ... |
| Ligne 1 |   |   |   |   |   |   |  | ... |
| Ligne 2 |   |   |   |   |   |   |  | ... |
| ⋮       |   |   |   |   |   |   |  |     |

La position de chaque caractère à l'écran est codée sur un octet dans la table des positions de caractère. Il y a 40 colonnes sur 24 lignes possibles, soit 960 caractères donc 960 octets. Chaque octet renferme le code du caractère à afficher à l'écran. (Voir [les codes des caractères](#) page 592.) Ainsi, si à l'écran se trouvent uniquement les caractères « HELLO! » dans le coin supérieur gauche, la table des formes 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 « ! » |

En dehors du OK et du curseur qui s'affiche à la suite, le reste de la table est rempli de 32 (020h), le code du caractère espace, puisqu'elle a été effacée à l'initialisation du mode d'écran.

Ces caractères sont disposés en VRAM de la façon suivante dans la table des formes de caractère :

| Adresse | +0         | +1         | +2         | +3         | +4         | +5         | +6         | ... |
|---------|------------|------------|------------|------------|------------|------------|------------|-----|
| 00000h  | 48h<br>"H" | 45h<br>"E" | 4Ch<br>"L" | 4Ch<br>"L" | 4Fh<br>"O" | 21h<br>"!" | 20h<br>" " | ... |
| 00028h  | 20h<br>" " | 20h<br>" " | 20h<br>" " | 20h<br>" " | 20h<br>" " | ...        |            |     |
| 00050h  | 20h<br>" " | 20h<br>" " | 20h<br>" " | ...        |            |            |            |     |
| 00078h  | 20h<br>" " | ...        | ...        |            |            |            |            |     |
| ⋮       |            |            |            |            |            |            |            |     |

À l'initialisation du SCREEN 0 par le système, la table des formes de caractère de la Main-ROM est copiée tel quelle vers la table des formes de caractère en VRAM.

Format de la table des formes de caractère :

| Caractère 00h | Caractère 01h | Caractère 02h | Caractère 03h | ... jusqu'à 0FFh |
|---------------|---------------|---------------|---------------|------------------|
| 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        |               |                  |

Celle-ci contient 2048 octets regroupés par paquets de 8 octets. Ce qui donne 2048 divisé par 8, soit 256 paquets qui définissent chacun la forme d'un caractère.

Chacun des caractères ont une taille de  $6 \times 8$  à l'écran bien que leur taille réelle est de  $8 \times 8$  dans la table des formes de caractère. Les bits à 1 correspondent à la couleur du texte et les bits à 0, à la couleur 0. Les colonnes des bits 0 et 1 sont ignorées à l'affichage.

Par exemple, le 65<sup>e</sup> paquet code le caractère « A » et ressemble à ceci :

|        |          |     |   |      |   |           | 7  | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----------|-----|---|------|---|-----------|----|---|---|---|---|---|---|---|
| 00A08h | contient | 32  | = | 020h | = | 00100000b | => |   |   |   |   |   |   |   |
| 00A09h | contient | 80  | = | 050h | = | 01010000b | => |   |   |   |   |   |   |   |
| 00A0Ah | contient | 136 | = | 088h | = | 10001000b | => |   |   |   |   |   |   |   |
| 00A0Bh | contient | 136 | = | 088h | = | 10001000b | => |   |   |   |   |   |   |   |
| 00A0Ch | contient | 248 | = | 0F8h | = | 11111000b | => |   |   |   |   |   |   |   |
| 00A0Dh | contient | 136 | = | 088h | = | 10001000b | => |   |   |   |   |   |   |   |
| 00A0Eh | contient | 136 | = | 088h | = | 10001000b | => |   |   |   |   |   |   |   |
| 00A0Fh | contient | 0   | = | 000h | = | 00000000b | => |   |   |   |   |   |   |   |

Le caractère n°65, donc l'octet  $65 \times 8 = 520$  (208h). On trouvera donc l'information qui nous intéresse à l'adresse de début de la table des formes (00800h) plus 208h, soit 00A08h.

Les deux bits de poids faible ne sont pas pris en compte (car tout caractère se trouve défini dans une matrice  $6 \times 8$ ).

Autre exemple, faisons la démarche en sens inverse, redéfinissons à présent le caractère « A ».

[illegible]

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. L'exécution de SCREEN 0 fera revenir les choses à la normale.

**Registres directement impliqués pour définir le mode SCREEN 0 à 40 colonnes :**

|                     | bit 7 | bit 6 | bit 5          | bit 4     | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|----------------|-----------|-------|-------|-------|-------|----------------|
| <b>Registre 0 :</b> | 0     | DG    | <del>IE2</del> | IE1       | M5    | M4    | M3    | EV    | (Tous les VDP) |
| <b>Registre 1 :</b> | 4/16k | BL    | IE0            | <b>M1</b> | M2    | 0     | SI    | MAG   | (Tous les VDP) |

Les bits M2 à M5 doivent être mis à 0 et M1 à 1 pour enclencher le mode SCREEN0 à 40 colonnes.

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 2 :</b> | 0     | N16   | N15   | N14   | N13   | N12   | N11   | N10   | (Tous les VDP) |

N10 à N16 codent les 7 bits de poids fort de l'adresse de la table des positions de caractère. L'adresse véritable s'obtient donc en multipliant la valeur de ces 7 bits par 400h. Sur MSX1, seuls les bits N10 à N13 sont pris en compte.

A l'initialisation du mode SCREEN 0 à 40 colonnes, ce registre contient 00h.

|                     |       |       |       |       |       |       |       |       |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
| <b>Registre 4 :</b> | 0     | 0     | F16   | F15   | F14   | F13   | F12   | F11   | (Tous les VDP) |

F11 à F16 codent les 6 bits de poids fort de l'adresse de la table des formes de caractère. L'adresse véritable s'obtient donc en multipliant la valeur de ces 6 bits par 800h. Sur MSX1, seuls les bits F11 à F13 sont pris en compte.

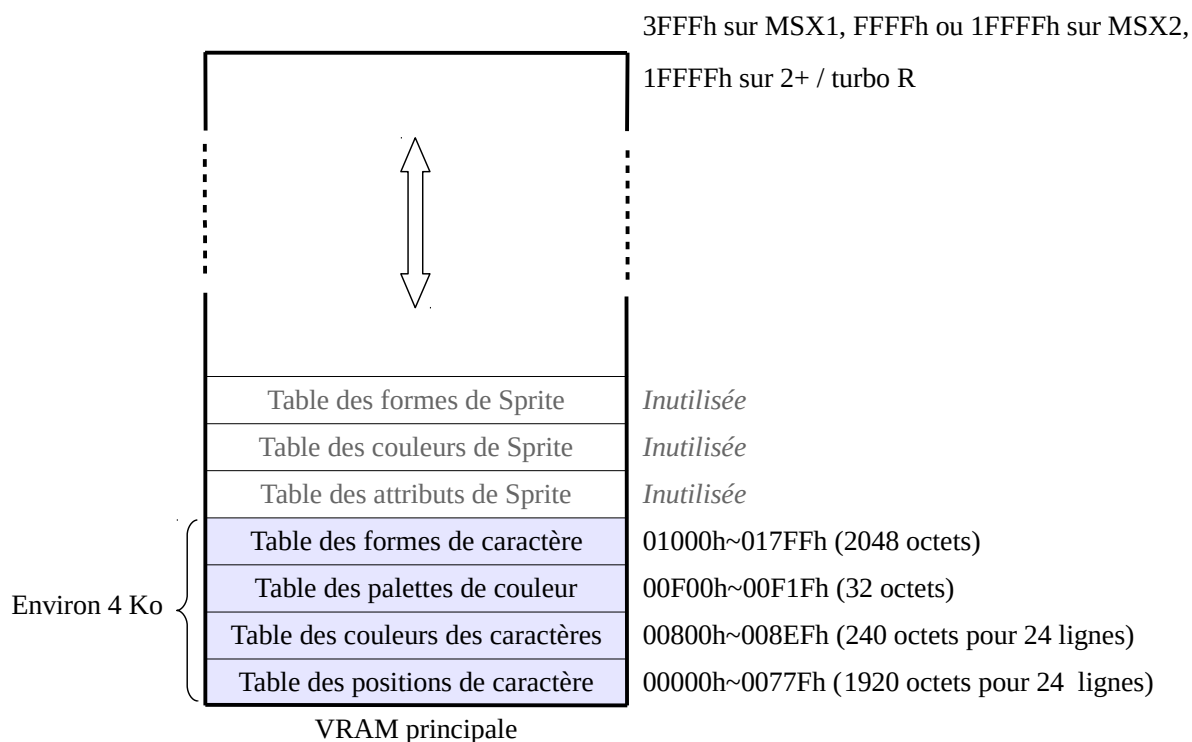
A l'initialisation du mode SCREEN 0 à 40 colonnes, ce registre contient 01h.

La couleur des caractères et de la bordure (arrière-plan) sont déterminés par le [registre 7](#).

## SCREEN 0 à 80 colonnes (mode texte 2)

|              |                                                         |
|--------------|---------------------------------------------------------|
| Résolution : | 512 × 192 / 212 pixels.                                 |
| Type :       | Mode texte, 80 colonnes × 24 lignes de caractère 6 × 8. |
| Couleurs :   | 4 couleurs sur 16 redéfinissables parmi 512.            |
| Sprite :     | Inutilisés.                                             |
| VDP :        | V9938 et V9958                                          |

La carte de la VRAM se présente ainsi à l'initialisation :



Le mode 80 colonnes fonctionne de manière très proche au mode texte 40 colonnes (voir ci-dessus pour plus de précisions). La table des positions de caractère contient toujours un octet par caractère à l'écran, mais cette fois 80 colonnes sur 24 lignes donc 1920 octets. Chaque octet renferme le code du caractère à afficher à l'écran. (Voir [les codes des caractères](#) page 592.)

```
10 SCREEN 0: WIDTH 80: 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
```

Celui-ci transforment tous les « B » de l'écran en un caractère « E » en bits inversés.

Ainsi, si à l'écran se trouvent uniquement les caractères « B B B B » dans le coin supérieur gauche, la table des positions commencera par :

|        |                              |
|--------|------------------------------|
| 00000h | 65 (41h), code du « B »      |
| 00001h | 32 (020h), code d'espacement |
| 00002h | 65 (41h), code du « B »      |
| 00003h | 32 (020h), code d'espacement |
| 00004h | 65 (41h), code du « B »      |
| 00005h | 32 (020h), code d'espacement |
| 00006h | 65 (41h), code du « B »      |
| 00007h | 32 (020h), code d'espacement |
| ⋮      | ⋮                            |

Le programme redéfinit le caractère « B » ainsi :

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |    | en binaire | en hexadécimal |
|---|---|---|---|---|---|---|---|----|------------|----------------|
| ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | => | 11111000b  | = 0F8h         |
| ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | => | 10001000b  | = 088h         |
| ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | => | 10111000b  | = 0B8h         |
| ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | => | 10011000b  | = 098h         |
| ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | => | 10111000b  | = 0B8h         |
| ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | => | 10001000b  | = 088h         |
| ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | => | 10001000b  | = 088h         |
| ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | => | 11111000b  | = 0F8h         |

L'instruction SCREEN 0 fera revenir les choses à la normale.

La table des couleurs de caractère permet de définir la couleur lors d'un clignotement le texte. Chaque bit correspond à un caractère à l'écran, voir le paragraphe 7.6 sur [les registres de contrôle du processeur vidéo](#) (Registres 12 et 13).

### **Registres directement impliqués pour définir le mode SCREEN 0 à 80 colonnes :**

|                     | bit 7 | bit 6 | bit 5 | bit 4     | bit 3 | bit 2     | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-----------|-------|-----------|-------|-------|----------------|
| <b>Registre 0 :</b> | 0     | DG    | IE2   | IE1       | M5    | <b>M4</b> | M3    | EV    | (Tous les VDP) |
| <b>Registre 1 :</b> | 4/16k | BL    | IE0   | <b>M1</b> | M2    | 0         | SI    | MAG   | (Tous les VDP) |

Les bits M2, M3 et M5 doivent être mis à 0 et M1 et M4 à 1 pour enclencher le mode SCREEN 0 à 80 colonnes.

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1      | bit 0      |                |
|---------------------|-------|-------|-------|-------|-------|-------|------------|------------|----------------|
| <b>Registre 2 :</b> | 0     | N16   | N15   | N14   | N13   | N12   | <b>N11</b> | <b>N10</b> | (Tous les VDP) |

N10 et N11 doivent être normalement mis à 1. Si N10 est mis à 0, les 64 premières lignes sont reproduites à l'identique sur les 64 suivantes mais avec un décalage de 12 caractères vers la droite sur l'axe horizontal. N11 est ignoré.

N12 à N16 codent les 5 bits de poids fort de l'adresse de la table des positions de caractère. L'adresse véritable s'obtient donc en multipliant la valeur de ces 5 bits par 1000h.

A l'initialisation du mode SCREEN 0 à 80 colonnes, ce registre contient 03h.

|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| <b>Registre 3 :</b>  | C13   | C12   | C11   | C10   | C9    | C8    | C7    | C6    | (Tous les VDP)   |
| <b>Registre 10 :</b> | 0     | 0     | 0     | 0     | 0     | C16   | C15   | C14   | (V9938 et V9958) |

Les registres 3 et 10 spécifient l'adresse de la table des couleurs des caractères. Cette table permet d'afficher des caractères avec deux autres couleurs pendant un certains laps de temps. (Voir la description du registre 7 et celle du bit 7 (LN) du [registre 9](#) pour plus de précision).

A l'initialisation du mode SCREEN 0 à 80 colonnes, ce registre contient 02h.

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 4 :</b> | 0     | 0     | F16   | F15   | F14   | F13   | F12   | F11   | (Tous les VDP) |

F11 à F16 codent les 7 bits de poids fort de l'adresse de la table des formes de caractère. L'adresse véritable s'obtient donc en multipliant la valeur de ce registre par 400h. Si seul le bit F11 est à 1, le registre se comportera comme si il contient la valeur 02h.

A l'initialisation du mode SCREEN 0 à 80 colonnes, ce registre contient 02h.

La couleur des caractères et de la bordure (arrière-plan) sont déterminés par le [registre 7](#).

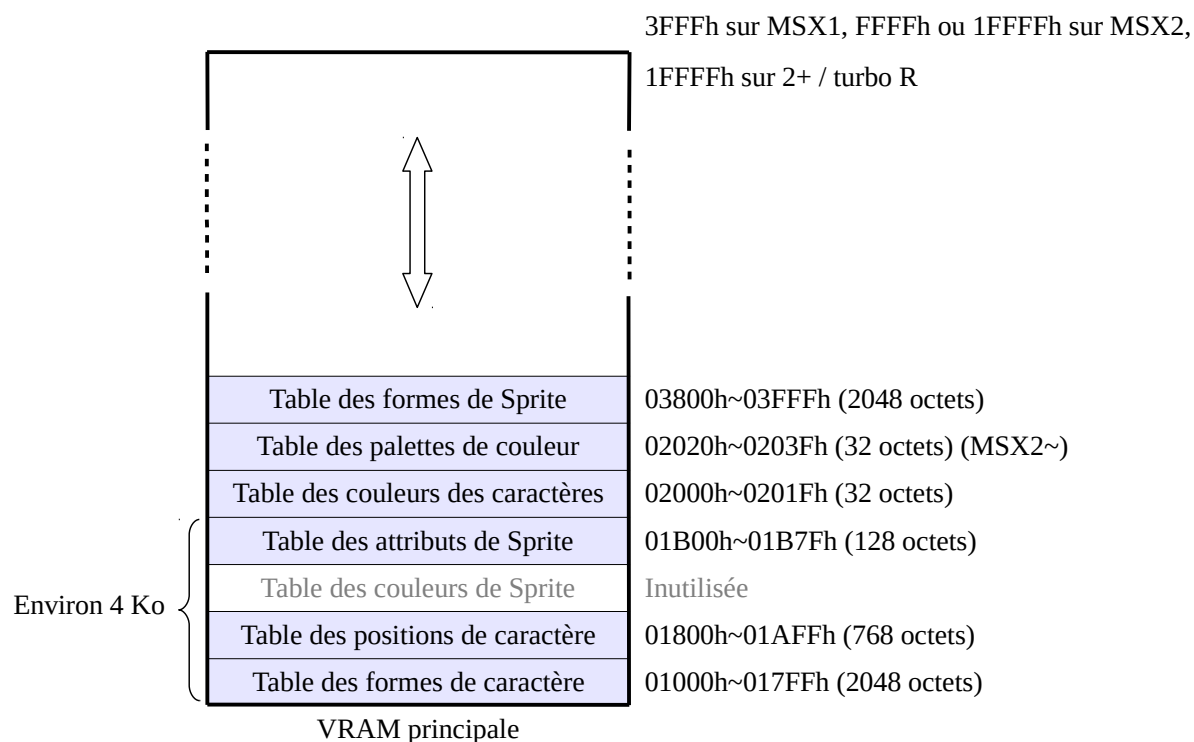
Il est possible de définir l'écran sur 26,5 lignes en mettant le bit 7 (LN) du [registre 9](#) à 1. Dans ce cas, la table des positions de caractère prendra 2160 octets (00000h~0086Fh) et la table des couleurs de caractère 270 octets (00800h~0090Dh).



## SCREEN 1 (mode graphique 1)

Résolution : 256 × 192 pixels.  
Type : Mode texte, 32 colonnes × 24 lignes, caractères 8 × 8,  
Couleurs : 15 couleurs sur MSX1,  
16 couleurs redéfinissables parmi 512 à partir du MSX2.  
Sprite : Type 1.  
VDP : Tous

La carte de la VRAM se présente ainsi à l'initialisation :



Le SCREEN 1 est appelé « Mode graphique 1 » mais fonctionne à peu près à la manière du mode texte 40 colonnes avec des couleurs et des Sprite de type 1 en plus. La table des positions de caractère est toujours constituée d'un octet par caractère à l'écran. Comme il y a 32 colonnes sur 24 lignes, la table a une taille de 768 octets. (Voir [les codes des caractères](#) page 592.)

Exécutez le petit programme suivant pour voir le fonctionnement :

```
10 SCREEN 1: WIDTH 32: PRINT"A BEN"
```

Maintenant, les caractères « A BEN » se trouvent dans le coin supérieur gauche l'écran, la table des positions de caractère 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 |
| ⋮      | ⋮                           |

Hormi le « OK » et le curseur qui s'affichent après l'exécution, le reste de la table ne comprend des 32 (020h), code d'espacement, puisque la table des positions de caractère est rempli de ce code à l'initialisation de l'écran.

La table des formes de caractère est composée de 2048 octets regroupés par paquets de 8 octets. Ce qui donne 256 paquets. Chacun d'eux correspond à un caractère et en définit la forme. Par exemple, le 32<sup>e</sup> paquet code le caractère d'espacement. La position dans la table de ce paquet se calcul ainsi :  $32 \times 8 = 256$  (100h). On trouvera donc l'information qui nous intéresse à l'adresse 0000h (table des formes de caractère) + 100h, soit 0100h. Les octets de 0100h à 0107h en VRAM sont toutes remplies de 00h (de vide). 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 car ils se trouvent cette fois définis dans une matrice  $8 \times 8$ .

Entrez 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
```

C'est fou ce que l'on peut faire avec 7 petites lignes Basic ! Après un CTRL+STOP, l'instruction SCREEN 1 fera revenir les choses comme auparavant.

Quant à la table des couleurs des caractères, 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 80
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
```

### Registres directement impliqués pour définir le mode SCREEN 1 :

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 0 :</b> | 0     | DG    | IE2   | IE1   | M5    | M4    | M3    | EV    | (Tous les VDP) |
| <b>Registre 1 :</b> | 4/16k | BL    | IE0   | M1    | M2    | 0     | SI    | MAG   | (Tous les VDP) |

M1, M3 à M5 doivent être mis à 0. Le bit M2 à 1 pour enclencher le mode SCREEN 1.

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 2 :</b> | 0     | N16   | N15   | N14   | N13   | N12   | N11   | N10   | (Tous les VDP) |

N10 à N13 codent les 4 bits de poids fort de l'adresse de la table des positions de caractère. L'adresse véritable s'obtient donc en multipliant la valeur de ces 4 bits par 400h. Sur MSX1, seuls les bits N10 à N13 sont pris en compte.

A l'initialisation du mode SCREEN 1, ce registre contient 06h.

|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| <b>Registre 3 :</b>  | C13   | C12   | C11   | C10   | C9    | C8    | C7    | C6    | (Tous les VDP)   |
| <b>Registre 10 :</b> | 0     | 0     | 0     | 0     | 0     | C16   | C15   | C14   | (V9938 et V9958) |

Les bits C6 à C16 codent les 8 bits de poids fort de l'adresse de la table des couleurs des caractères. L'adresse réelle en VRAM s'obtient donc en multipliant le contenu du registre 3 par 40h. Sur MSX1, il n'y a pas de registre 10.

A l'initialisation du mode SCREEN 1, ce registre contient 80h. Le registre 10 est ignoré.

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 4 :</b> | 0     | 0     | F16   | F15   | F14   | F13   | F12   | F11   | (Tous les VDP) |

F13 à F16 codent les 6 bits de poids fort de l'adresse de la table des formes de motif. L'adresse véritable s'obtient en multipliant la valeur de ces 6 bits par 800h. Sur MSX1, seuls les bits F11 à F13 sont pris en compte.

A l'initialisation du mode SCREEN 1, ce registre contient 00h.

La couleur des caractères et de la bordure (arrière-plan) sont déterminés par le [registre 7](#).

## SCREEN 2 (mode graphique 2)

Résolution : 256 × 192 pixels.

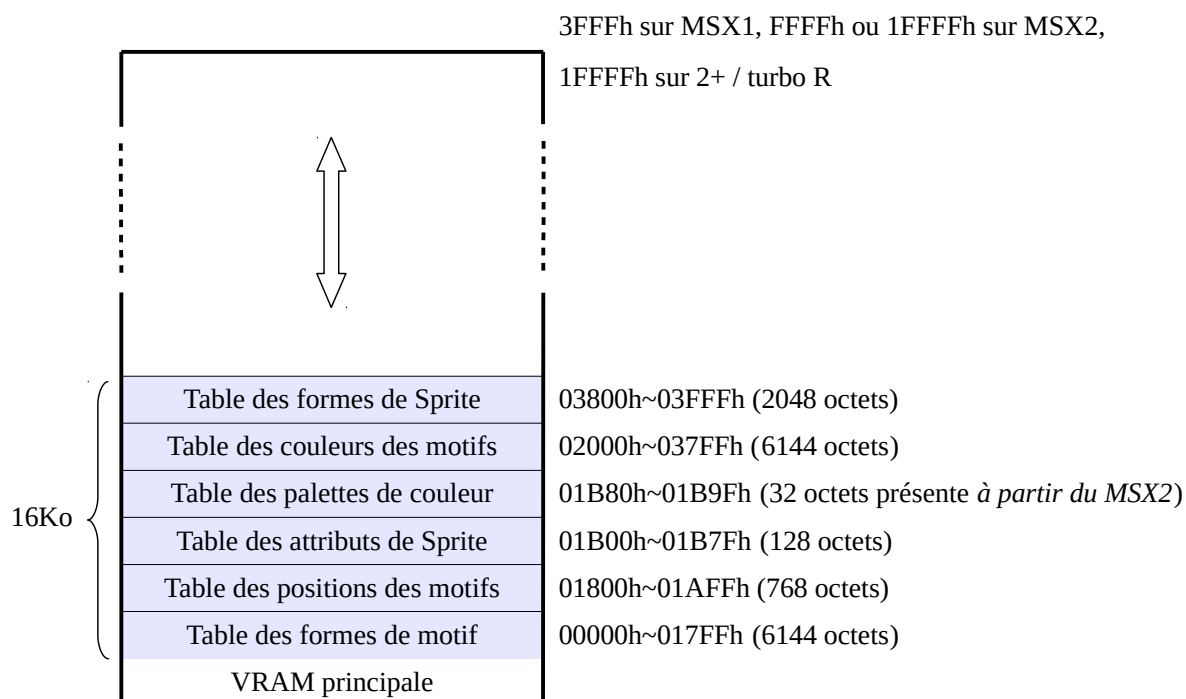
Type : Motifs 8 × 8 pixels sur 32 colonnes et 8 x 3 lignes.

Couleurs : 15 couleurs dont 2 par ligne horizontale de motif sur MSX1,  
16 couleurs parmi 512 dont 2 par ligne horizontale de motif à partir du MSX2.

Sprite : Type 1.

VDP : Tous

La carte de la VRAM se présente ainsi à l'initialisation :



Le SCREEN 2 est un mode d'écran graphique dont la partie visible est divisée en trois bandes horizontales. Chaque bande est composée d'un jeu de 256 motifs (32 x 8 motifs).

Le [registre 7](#) défini la couleur de bordure (arrière-plan).

Le système initialise la table des positions des motifs de la façon suivante :

|                              |     |     |     |     |     |     |     |     |     |     |     |     |
|------------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1 <sup>er</sup> jeu de motif | 0   | 1   | 2   | 3   | 4   | ... | ... | 27  | 28  | 29  | 30  | 31  |
|                              | 32  | 33  | 34  | 35  | ... | ... | ... | 60  | 61  | 62  | 63  | ... |
|                              | ⋮   |     |     |     |     |     |     |     |     |     |     | ⋮   |
|                              | ⋮   |     |     |     |     |     |     |     |     |     |     | ⋮   |
|                              | 192 | 193 | ... |     |     |     |     | ... |     | 122 | 123 |     |
|                              | 224 | 225 | 226 | ... |     |     |     | ... | 253 | 254 | 255 |     |
| 2 <sup>e</sup> jeu de motif  | 0   | 1   | 2   | 3   | 4   | ... | ... | 27  | 28  | 29  | 30  | 31  |
|                              | 32  | 33  | 34  | 35  | ... | ... | ... | 60  | 61  | 62  | 63  | ... |
|                              | ⋮   |     |     |     |     |     |     |     |     |     |     | ⋮   |
|                              | ⋮   |     |     |     |     |     |     |     |     |     |     | ⋮   |
|                              | 192 | 193 | ... |     |     |     |     | ... |     | 122 | 123 |     |
|                              | 224 | 225 | 226 | ... |     |     |     | ... | 253 | 254 | 255 |     |
| 3 <sup>e</sup> jeu de motif  | 0   | 1   | 2   | 3   | 4   | ... | ... | 27  | 28  | 29  | 30  | 31  |
|                              | 32  | 33  | 34  | 35  | ... | ... | ... | 60  | 61  | 62  | 63  | ... |
|                              | ⋮   |     |     |     |     |     |     |     |     |     |     | ⋮   |
|                              | ⋮   |     |     |     |     |     |     |     |     |     |     | ⋮   |
|                              | 192 | 193 | ... |     |     |     |     | ... |     | 122 | 123 |     |
|                              | 224 | 225 | 226 | ... |     |     |     | ... | 253 | 254 | 255 |     |

La table des formes a le même format qu'en SCREEN 1. 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 motif. Sachant que dans ce mode d'écran, il y a 3 jeux de motifs, la table des formes fait donc  $3 \times 2048$  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 de motifs. 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 font (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 formes de caractère. 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, Motif N°10
40 FOR I=0 TO 7: READ A$: VPOKE I,VAL("&H"+A$): NEXT
50 '
60 ' Table des couleurs, Motif N°0
70 FOR I=&H2000 TO &H2007: READ A$: VPOKE I,VAL("&H"+A$): NEXT
80 '
90 ' Table des positions, 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 GOTO 160
170 '
180 ' Données pour la forme
190 DATA 00,BB,00,DD,00,BB,00,DD
200 '
210 ' Données pour les couleurs
220 DATA 00,60,00,50,00,A0,00,20

```

Nous avons redéfini la forme du motif 0, puis nous l'avons affichée sur toutes la première et la septième ligne, ainsi que sur les côtés.

Vous remarquerez, à la ligne 150, que tous les dessins s'effectuent derrière notre cadre (ce qui est logique). Attention cependant à ne pas utiliser les pixels dans le cadre entre (0, 0) et (7, 7).

### **Registres directement impliqués pour définir le mode SCREEN 2 :**

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 0 :</b> | 0     | DG    | IE2   | IE1   | M5    | M4    | M3    | EV    | (Tous les VDP) |
| <b>Registre 1 :</b> | 4/16k | BL    | IE0   | M1    | M2    | 0     | SI    | MAG   | (Tous les VDP) |

M1 à M5 doivent être mis à 0 pour enclencher le mode SCREEN 2.

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 2 :</b> | 0     | N16   | N15   | N14   | N13   | N12   | N11   | N10   | (Tous les VDP) |

N10 à N16 codent les 7 bits de poids fort de l'adresse de la table des positions des motifs. L'adresse véritable s'obtient donc en multipliant la valeur de ces 7 bits par 400h. Sur MSX1 seuls les bits N10 à N13 sont pris en compte.

A l'initialisation du mode SCREEN 2, ce registre contient 06h.

|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| <b>Registre 3 :</b>  | C13   | C12   | C11   | C10   | C9    | C8    | C7    | C6    | (Tous les VDP)   |
| <b>Registre 10 :</b> | 0     | 0     | 0     | 0     | 0     | C16   | C15   | C14   | (V9938 et V9958) |

Les bits C6 à C12 doivent normalement être mis à 1. Voici la description de chacun d'eux lorsque mis à 0 :

- C6 - La partie de la table des couleurs des motifs qui correspond aux lignes verticales 0 à 63 est réutilisée pour les lignes 64 à 127, et celle pour les lignes verticales 128 à 191 est réutilisée pour les lignes 128 à 255. Attention car sur les VDP de Texas Instrument, ce bit agit de la même façon sur la table des formes de motif.
- C7 - La partie de la table des couleurs des motifs qui correspond aux lignes verticales 0 à 127 est réutilisée pour les lignes 128 à 255. Attention car sur les VDP de Texas Instrument, ce bit agit de la même façon sur la table des formes de motif.
- C8 - La partie de la table des couleurs des motifs qui correspond aux lignes horizontales 0 à 7 est réutilisée pour les lignes 8 à 15, celle pour les lignes 15 à 23 est réutilisée pour les lignes 24 à 31, celle pour les lignes 32 à 39 est réutilisée pour les lignes 40 à 47, etc.
- C9 - La partie de la table des couleurs des motifs qui correspond aux lignes horizontales 0 à 15 est réutilisée pour les lignes 16 à 31, celle pour les lignes 32 à 47 est réutilisée pour les lignes 48 à 63, celle pour les lignes 64 à 79 est réutilisée pour les lignes 80 à 96, etc.
- C10 - La partie de la table des couleurs des motifs qui correspond aux lignes horizontales 0 à 31 est réutilisée pour les lignes 32 à 63, celle pour les lignes 32 à 47 est réutilisée pour les lignes 48 à 63, celle pour les lignes 64 à 95 est réutilisée pour les lignes 96 à 128, etc.
- C11 - La partie de la table des couleurs des motifs qui correspond aux lignes horizontales 0 à 63 est réutilisée pour les lignes 64 à 127.
- C12 - La partie de la table des couleurs des motifs qui correspond aux lignes horizontales 0 à 63 est réutilisée pour les lignes 128 à 255.

Les bits C13 à C16 codent les 4 bits de poids fort de l'adresse de la table des couleurs des motifs. L'adresse réelle en VRAM s'obtient donc en multipliant cette valeur sur 4 bits par 2000h. Sur MSX1, il n'y a pas de registre 10.

A l'initialisation du mode SCREEN 2, ces registres contiennent FFh et 00h.



|                     |       |       |       |       |       |       |            |            |                |
|---------------------|-------|-------|-------|-------|-------|-------|------------|------------|----------------|
|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1      | bit 0      |                |
| <b>Registre 4 :</b> | 0     | 0     | F16   | F15   | F14   | F13   | <i>F12</i> | <i>F11</i> | (Tous les VDP) |

Les bits F11 et F12 doivent normalement être mis à 1. Si F12 est à 0, le premier tiers de la table des formes de motif sera réutilisé pour afficher le second tiers de l'écran. Si le bit F11 est à 0, le premier tiers de la table des formes de motif sera réutilisé pour afficher le troisième tiers de l'écran.

F13 à F16 codent les 4 bits de poids fort de l'adresse de la table des formes de motif. L'adresse véritable s'obtient en multipliant la valeur de ces 4 bits par 2000h. Sur MSX1, seuls le bit F13 est pris en compte. L'adresse ne peut donc être que 0000h ou 4000h.

A l'initialisation du mode SCREEN 2, ce registre contient 03h.

## SCREEN 3 (mode multicolore)

Résolution :  $64 \times 48$ .

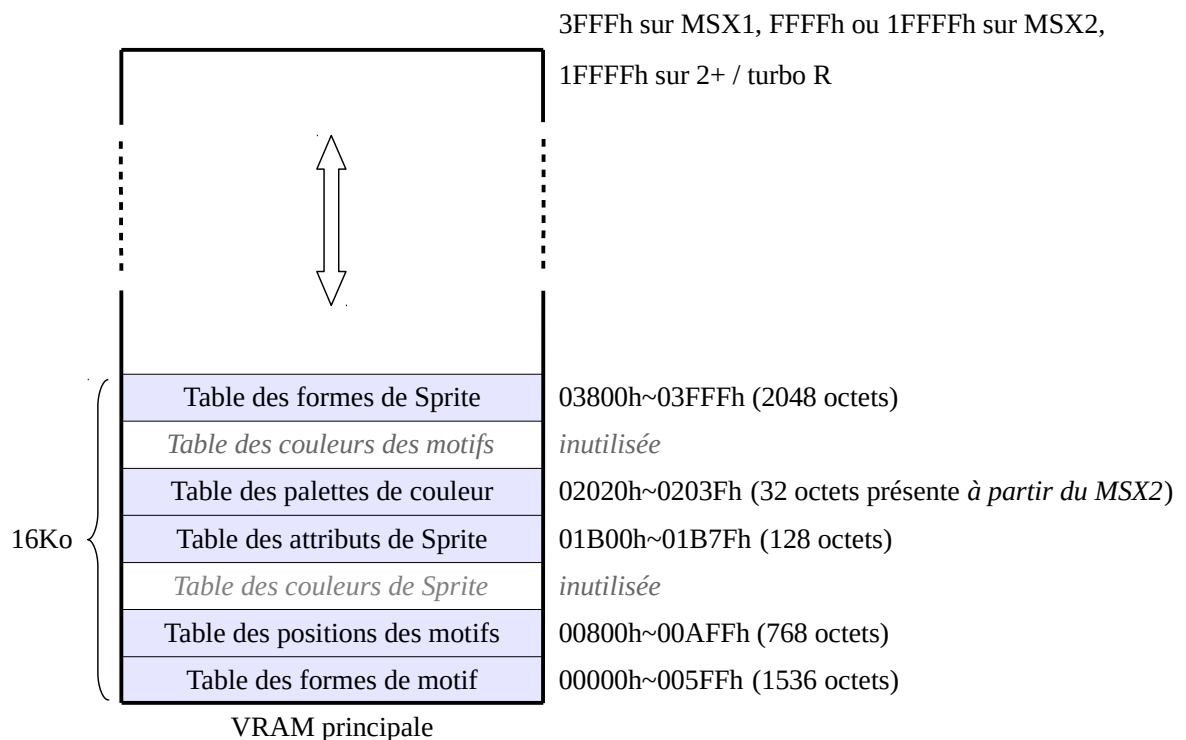
Type : Motif de  $2 \times 2$  pixels.

Couleurs : 15 couleurs sur MSX1,  
16 couleurs redéfinissables parmi 512 à partir du MSX2.

Sprite : Type 1.

VDP : Tous.

La carte de la VRAM se présente ainsi à l'initialisation :























Le mode d'écran SCREEN 3 ressemble à un mode d'écran Bitmap mais fonctionne un peu comme le SCREEN 2, mais avec une résolution de 64 sur 48. L'écran est donc composé de motifs.

Il y a 4 jeux de 192 motifs. Leur taille est de  $2 \times 2$  pixels. Chaque octet de la table des formes de motif définissent la couleur des 2 pixels horizontaux d'un motif.

A l'initialisation, les 8 premiers octets de la table formes de motif correspondent aux 4 premiers motifs de chaque jeu. Les 8 octets suivant correspondent aux 4 motifs en haut à gauche de l'écran.

Voici un petit programme pour visualiser l'agencement de la table des formes de motif :

| Y\X | 0                                                                                 | 1                                                                                 | 2                                                                                 | 3                                                                                 | ... |                                              |
|-----|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-----|----------------------------------------------|
| 0   |  |  |  |  |     | 10 COLOR 15,4,4: SCREEN3                     |
| 1   |  |  |  |  |     | 20 VPOKE 0,&H61 ' Forme du motif 0 du jeu 0  |
| 2   |  |  |                                                                                   |                                                                                   |     | 30 VPOKE 1,&H15                              |
| 3   |  |  |                                                                                   |                                                                                   |     | 40 VPOKE 2,&HBD ' Forme du motif 0 du jeu 1  |
| 4   |  |  |                                                                                   |                                                                                   |     | 50 VPOKE 3,&HDB                              |
| 5   |  |  |                                                                                   |                                                                                   |     | 60 VPOKE 4,&H2A ' Forme du motif 0 du jeu 2  |
| 6   |  |  |                                                                                   |                                                                                   |     | 70 VPOKE 5,&HA2                              |
| 7   |  |  |                                                                                   |                                                                                   |     | 80 VPOKE 6,&H63 ' Forme du motif 0 du jeu 3  |
| 8   |                                                                                   |                                                                                   |                                                                                   |                                                                                   |     | 90 VPOKE 7,&H36                              |
| :   |                                                                                   |                                                                                   |                                                                                   |                                                                                   |     | 100 VPOKE 8,&HD7 ' Forme du motif 1 du jeu 0 |
|     |                                                                                   |                                                                                   |                                                                                   |                                                                                   |     | 110 VPOKE 9,&HA2                             |
|     |                                                                                   |                                                                                   |                                                                                   |                                                                                   |     | 120 GOTO 120                                 |

Ainsi chaque colonnes de 2 x 8 pixels sont composées de 4 motifs qui sont disposés comme indiqué dans le tableau de la page suivante.

Avant de passer à la suite, voici un programme d'exemple pour montrer comment fonctionne le SCREEN 3 en BASIC.

```

10 COLOR15,4,1: SCREEN 3
15 ' Tracer un point avec PSET
20 X=127: Y=95: C=6: PSET(X,Y),7
30 IF NOT STRIG(0) THEN 30
35 ' Tracer un point avec VPOKE au même endroit
40 X=X\4: Y=Y\4: ' Calcul des coordonnées réelles
50 AD=(Y\8)*256+((X\2)*8)+(Y MOD 8) ' Calcul de l'adresse
60 IF X MOD 2=0 THEN VPOKE AD,C*16+(VPEEK(AD)AND15)
70 IF X MOD 2=1 THEN VPOKE AD,(VPEEK(AD)AND&HF0)+C
80 GOTO 80

```

Ce programme trace un point de couleur 7 avec l'instruction PSET puis, un autre par dessus de couleur 6 en pressant la barre d'espace. Vous pouvez changer les coordonnées X et Y de la ligne 20 pour vérifier le calcul de l'adresse.

Vous avez noté que, bien que la résolution du mode SCREEN 3 est de 64 x 48, les coordonnées des instructions graphiques du BASIC travaillent comme si résolution est de 256 x 192. L'interpréteur divise les coordonnées par 4 en interne.

Table des positions de motif à l'initialisation :

| Y\X | 0     | 2     | 4     | 6     | 8     | 10    | ... | ... | 52    | 54    | 56    | 58    | 60    | 62    |
|-----|-------|-------|-------|-------|-------|-------|-----|-----|-------|-------|-------|-------|-------|-------|
| 0   | 0,0   | 1,0   | 2,0   | 3,0   | 4,0   | 5,0   | ... | ... | 26,0  | 27,0  | 28,0  | 29,0  | 30,0  | 31,0  |
| 2   | 0,1   | 1,1   | 2,1   | 3,1   | 4,1   |       |     |     |       | 27,1  | 28,1  | 29,1  | 30,1  | 31,1  |
| 4   | 0,2   | 1,2   | 2,2   | 3,2   | ...   |       |     |     |       | ...   | 28,2  | 29,2  | 30,2  | 31,2  |
| 6   | 0,3   | 1,3   | 2,3   | 3,3   |       |       |     |     |       |       | 28,3  | 29,3  | 30,3  | 31,3  |
| 8   | 32,0  | 33,0  |       |       |       |       |     |     |       |       | 60,0  | 61,0  | 62,0  | 63,0  |
| 10  | 32,1  | 33,1  |       |       |       |       |     |     |       |       |       |       | 62,1  | 63,1  |
| 12  | 32,2  |       |       |       |       |       |     |     |       |       |       |       |       | 63,2  |
| 14  | 32,3  |       |       |       |       |       |     |     |       |       |       |       |       | 63,3  |
| 16  | 64,0  |       |       |       |       |       |     |     |       |       |       |       |       | 95,0  |
| ⋮   | 64,1  |       |       |       |       |       |     |     |       |       |       |       |       | 95,1  |
|     | ⋮     |       |       |       |       |       |     |     |       |       |       |       |       | ⋮     |
|     | ⋮     |       |       |       |       |       |     |     |       |       |       |       |       | ⋮     |
| ⋮   | 128,3 |       |       |       |       |       |     |     |       |       |       |       |       | 159,3 |
| 40  | 160,0 |       |       |       |       |       |     |     |       |       |       |       | ...   | 191,0 |
| 42  | 160,1 | 161,1 | ...   |       |       |       |     |     |       |       |       | ...   | 190,1 | 191,1 |
| 44  | 160,2 | 161,2 | 162,2 | 163,2 | ...   |       |     |     |       | ...   | 188,2 | 189,2 | 190,2 | 191,2 |
| 46  | 160,3 | 161,3 | 162,3 | 163,3 | 164,3 | 165,3 | ... | ... | 186,3 | 187,3 | 188,3 | 189,3 | 190,3 | 191,3 |

Voici un exemple Basic utilisant des motifs :

```

10 SCREEN 3: COLOR 10,1,1: CLS
20 ' Table des formes
30 ' Definis les motifs 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 ' Table des positions de motif
60 ' Fait un joli cadre en brique avec les motifs 0
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 GOTO 130

```

### Registres directement impliqués pour définir le mode SCREEN 3 :

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 0 :</b> | 0     | DG    | IE2   | IE1   | M5    | M4    | M3    | EV    | (Tous les VDP) |
| <b>Registre 1 :</b> | 4/16k | BL    | IE0   | M1    | M2    | 0     | SI    | MAG   | (Tous les VDP) |

M1, M2, M4 et M5 doivent être mis à 0. M3 doit être à 1 pour enclencher le mode SCREEN 3.

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 2 :</b> | 0     | N16   | N15   | N14   | N13   | N12   | N11   | N10   | (Tous les VDP) |

N10 à N16 codent les 7 bits de poids fort de l'adresse de la table des positions des motifs. L'adresse véritable s'obtient donc en multipliant la valeur de ces 7 bits par 400h. Sur MSX1 seuls les bits N10 à N13 sont pris en compte. L'adresse ne peut donc varier qu'entre 0 et 3C00h.

A l'initialisation du mode SCREEN 3, ce registre contient 02h.

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 4 :</b> | 0     | 0     | F16   | F15   | F14   | F13   | F12   | F11   | (Tous les VDP) |

F13 à F16 codent les 6 bits de poids fort de l'adresse de la table des formes de motif. L'adresse véritable s'obtient en multipliant la valeur de ces 6 bits par 800h. Sur MSX1, seuls les bits F11 à F13 sont pris en compte.

A l'initialisation du mode SCREEN 3, ce registre contient 00h.

## SCREEN 4 (mode graphique 3)

Résolution : 256 × 192.

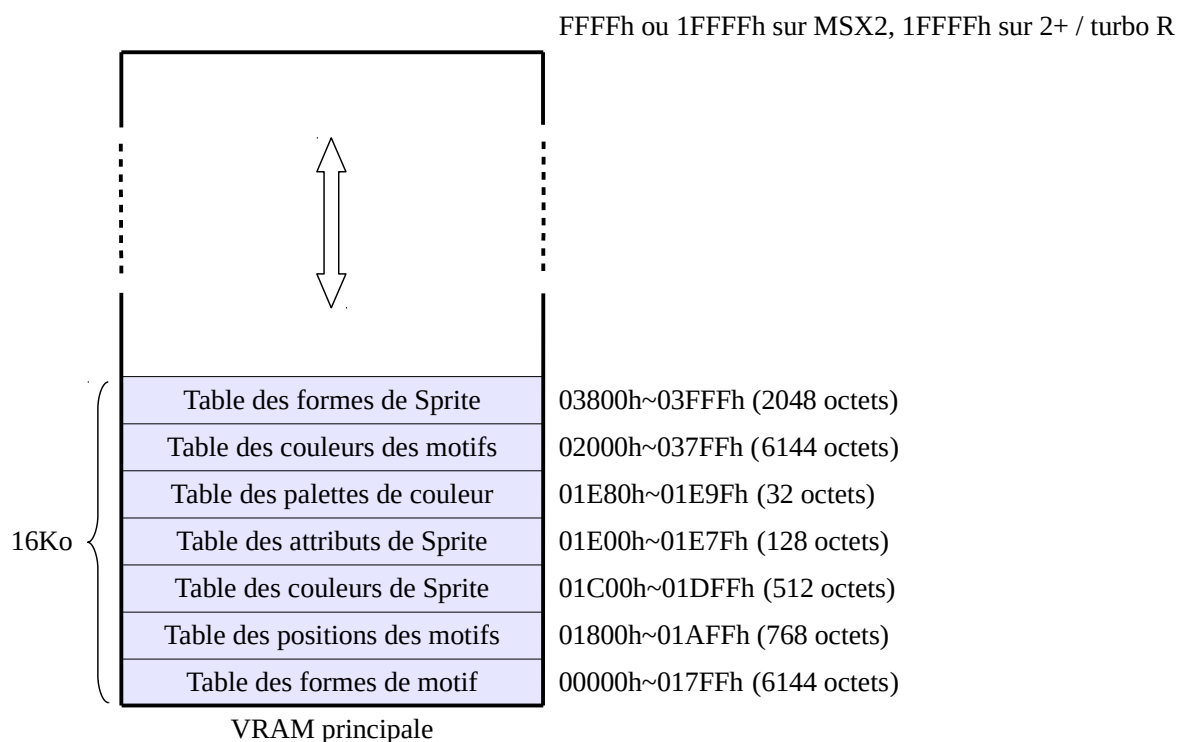
Type : Motifs 8 × 8 pixels sur 32 colonnes et 8 x 3 lignes.

Couleurs : 16 couleurs parmi 512 dont 2 par ligne horizontale de motif.

Sprite : Type 2.

VDP : V9938 et V9958

La carte de la VRAM se présente ainsi à l'initialisation :



Le SCREEN 4 est en tous points identique au SCREEN 2 à l'exception de plusieurs tables qui commencent à une autre adresse et des Sprite 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 Sprite, voir le paragraphe 7.9 « [Les Sprite et leur fonctionnement](#) », page 360.

### Registres directement impliqués pour définir le mode SCREEN 4 :

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2     | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-----------|-------|-------|----------------|
| <b>Registre 0 :</b> | 0     | DG    | IE2   | IE1   | M5    | <b>M4</b> | M3    | EV    | (Tous les VDP) |
| <b>Registre 1 :</b> | 4/16k | BL    | IE0   | M1    | M2    | 0         | SI    | MAG   | (Tous les VDP) |

M1 à M3 et M5 doivent être mis à 0. M4 doivent être mis à 1 pour enclencher le mode SCREEN 4.

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 2 :</b> | 0     | N16   | N15   | N14   | N13   | N12   | N11   | N10   | (Tous les VDP) |

N10 à N16 codent les 7 bits de poids fort de l'adresse de la table des positions des motifs. L'adresse véritable s'obtient donc en multipliant la valeur de ces 7 bits par 400h.

A l'initialisation du mode SCREEN 4, ce registre contient 06h.

|                      | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                  |
|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| <b>Registre 3 :</b>  | C13   | C12   | C11   | C10   | C9    | C8    | C7    | C6    | (Tous les VDP)   |
| <b>Registre 10 :</b> | 0     | 0     | 0     | 0     | 0     | C16   | C15   | C14   | (V9938 et V9958) |

Les bits C6 à C12 doivent normalement être mis à 1. Voici la description de chacun d'eux lorsque mis à 0 :

- C6 - La partie de la table des couleurs des motifs qui correspond aux lignes verticales 0 à 63 est réutilisée pour les lignes 64 à 127, et celle pour les lignes verticales 128 à 191 est réutilisée pour les lignes 128 à 255.
- C7 - La partie de la table des couleurs des motifs qui correspond aux lignes verticales 0 à 127 est réutilisée pour les lignes 128 à 255.
- C8 - La partie de la table des couleurs des motifs qui correspond aux lignes horizontales 0 à 7 est réutilisée pour les lignes 8 à 15, celle pour les lignes 15 à 23 est réutilisée pour les lignes 24 à 31, celle pour les lignes 32 à 39 est réutilisée pour les lignes 40 à 47, etc.
- C9 - La partie de la table des couleurs des motifs qui correspond aux lignes horizontales 0 à 15 est réutilisée pour les lignes 16 à 31, celle pour les lignes 32 à 47 est réutilisée pour les lignes 48 à 63, celle pour les lignes 64 à 79 est réutilisée pour les lignes 80 à 96, etc.
- C10 - La partie de la table des couleurs des motifs qui correspond aux lignes horizontales 0 à 31 est réutilisée pour les lignes 32 à 63, celle pour les lignes 32 à 47 est réutilisée pour les lignes 48 à 63, celle pour les lignes 64 à 95 est réutilisée pour les lignes 96 à 128, etc.
- C11 - La partie de la table des couleurs des motifs qui correspond aux lignes horizontales 0 à 63 est réutilisée pour les lignes 64 à 127.
- C12 - La partie de la table des couleurs des motifs qui correspond aux lignes horizontales 0 à 63 est réutilisée pour les lignes 128 à 255.

Les bits C13 à C16 codent les 4 bits de poids fort de l'adresse de la table des couleurs des motifs. L'adresse réelle en VRAM s'obtient donc en multipliant cette valeur sur 4 bits par 2000h.

A l'initialisation du mode SCREEN 4, ces registres contiennent FFh et 00h.

|                     |       |       |       |       |       |       |            |            |                |
|---------------------|-------|-------|-------|-------|-------|-------|------------|------------|----------------|
|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1      | bit 0      |                |
| <b>Registre 4 :</b> | 0     | 0     | F16   | F15   | F14   | F13   | <i>F12</i> | <i>F11</i> | (Tous les VDP) |

Les bits F11 et F12 doivent normalement être mis à 1. Si F12 est à 0, le premier tiers de la table des formes de motif sera réutilisé pour afficher le second tiers de l'écran. Si le bit F11 est à 0, le premier tiers de la table des formes de motif sera réutilisé pour afficher le troisième tiers de l'écran.

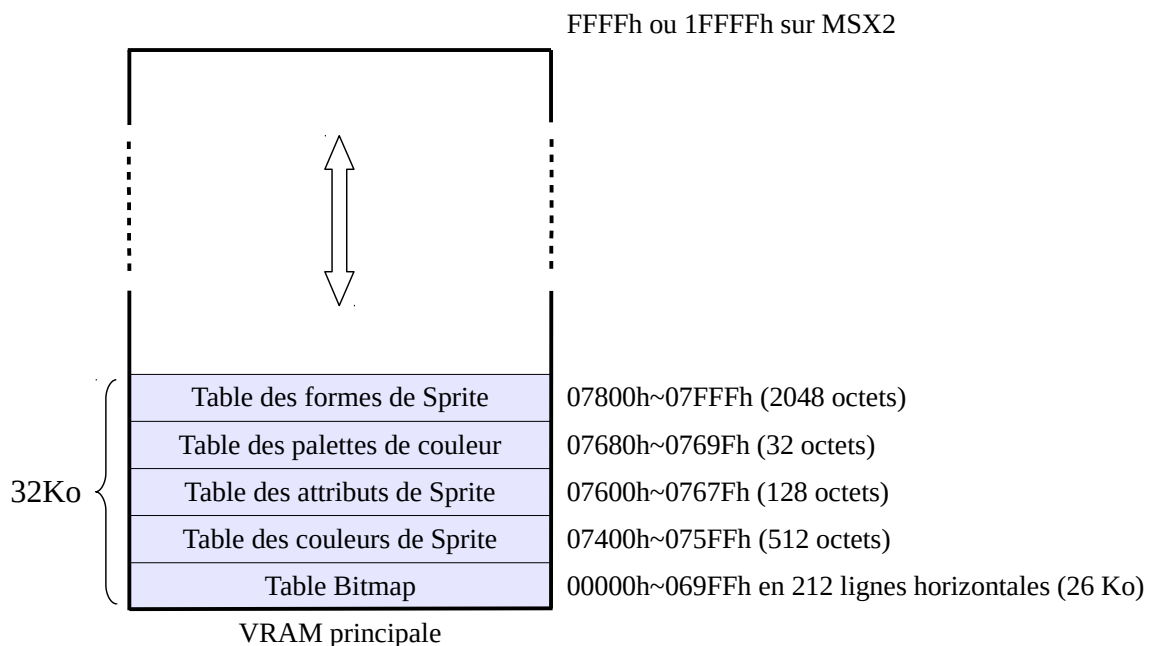
F13 à F16 codent les 4 bits de poids fort de l'adresse de la table des formes de motif. L'adresse véritable s'obtient donc en multipliant la valeur de ces 4 bits par 2000h. Sur MSX1, seuls le bit F13 est pris en compte. L'adresse ne peut donc être que 0000h ou 4000h.

A l'initialisation du mode SCREEN 4, ce registre contient 03h.

## SCREEN 5 (mode graphique 4)

Résolution : 256 × 212 / 192.  
 Type : Bitmap.  
 Couleurs : 16 couleurs parmi 512.  
 Sprite : Type 2.  
 VDP : V9938 et V9958.

La carte de la VRAM se présente ainsi à l'initialisation :



La table des palettes de couleur est juste une zone où le système y stocke les valeurs de la palette par défaut.



Ce mode d'écran est disponible à partir du MSX2. Dans ce mode, chaque pixel de l'avant plan est codé directement par sa couleur, et ce, indépendamment des autres pixels. Il n'existe donc plus de contrainte puisque les table de couleur ou de formes sont remplacées par la table Bitmap.

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.

La table Bitmap du mode SCREEN 5 est codée de la façon suivante :

|        | bit 7         | bit 6 | bit 5 | bit 4 | bit 3         | bit 2 | bit 1 | bit 0 |
|--------|---------------|-------|-------|-------|---------------|-------|-------|-------|
| 00000h | pixel (0,0)   |       |       |       | pixel (1,0)   |       |       |       |
| 00001h | pixel (2,0)   |       |       |       | pixel (3,0)   |       |       |       |
| ⋮      | ⋮             |       |       |       | ⋮             |       |       |       |
| 0007Fh | pixel (254,0) |       |       |       | pixel (255,0) |       |       |       |
| 00080h | pixel (0,1)   |       |       |       | pixel (1,1)   |       |       |       |
| ⋮      | ⋮             |       |       |       | ⋮             |       |       |       |

#### Registres directement impliqués pour définir le mode SCREEN 5 :

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2     | bit 1     | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-----------|-----------|-------|----------------|
| <b>Registre 0 :</b> | 0     | DG    | IE2   | IE1   | M5    | <b>M4</b> | <b>M3</b> | EV    | (Tous les VDP) |
| <b>Registre 1 :</b> | 4/16k | BL    | IE0   | M1    | M2    | 0         | SI        | MAG   | (Tous les VDP) |

M1, M2 et M5 doivent être mis à 0. M3 et M4 doivent être à 1 pour activer le mode SCREEN 5.

|                     |       |       |       |       |       |       |       |       |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
| <b>Registre 2 :</b> | 0     | N16   | N15   | N14   | N13   | N12   | N11   | N10   | (Tous les VDP) |

Les bits N10 à N14 doivent être normalement mis à 1. Ils sont mis à 1 à l'initialisation. Ces bits servent à répéter l'affichage de lignes de l'avant-plan de la façon suivante.

- N14 – Mettre ce bit à 0 pour répéter l'affichage des 128 premières lignes à la suite.
- N13 – Mettre ce bit à 0 pour répéter l'affichage des 64 premières lignes à la place des lignes 64 à 127 et répéter l'affichage des lignes 128 à 191 à la place des lignes 192 à 255.
- N12 – Mettre ce bit à 0 pour répéter l'affichage des 32 premières lignes à la place des lignes 32 à 63, répéter l'affichage des lignes 64 à 95 à la place des lignes 96 à 127, répéter l'affichage des lignes 128 à 159 à la place des lignes 160 à 191 et répéter l'affichage des lignes 192 à 231 à la place des lignes 232 à 255.
- N11 – Mettre ce bit à 0 pour répéter l'affichage des 16 premières lignes à la place des lignes 16 à 31, répéter les lignes 32 à 47 à la place des lignes 48 à 63, répéter les lignes 64 à 79 à la place des lignes 80 à 95, répéter les lignes 96 à 111 à la place des lignes 112 à 127, et ainsi de suite...
- N10 – Mettre ce bit à 0 pour répéter l'affichage des 8 premières lignes à la place des lignes 8 à 15, répéter les lignes 16 à 23 à la place des lignes 24 à 31, répéter les lignes 32 à 39 à la place des lignes 40 à 47, répéter les lignes 48 à 57 à la place des lignes 56 à 63, et ainsi de suite...

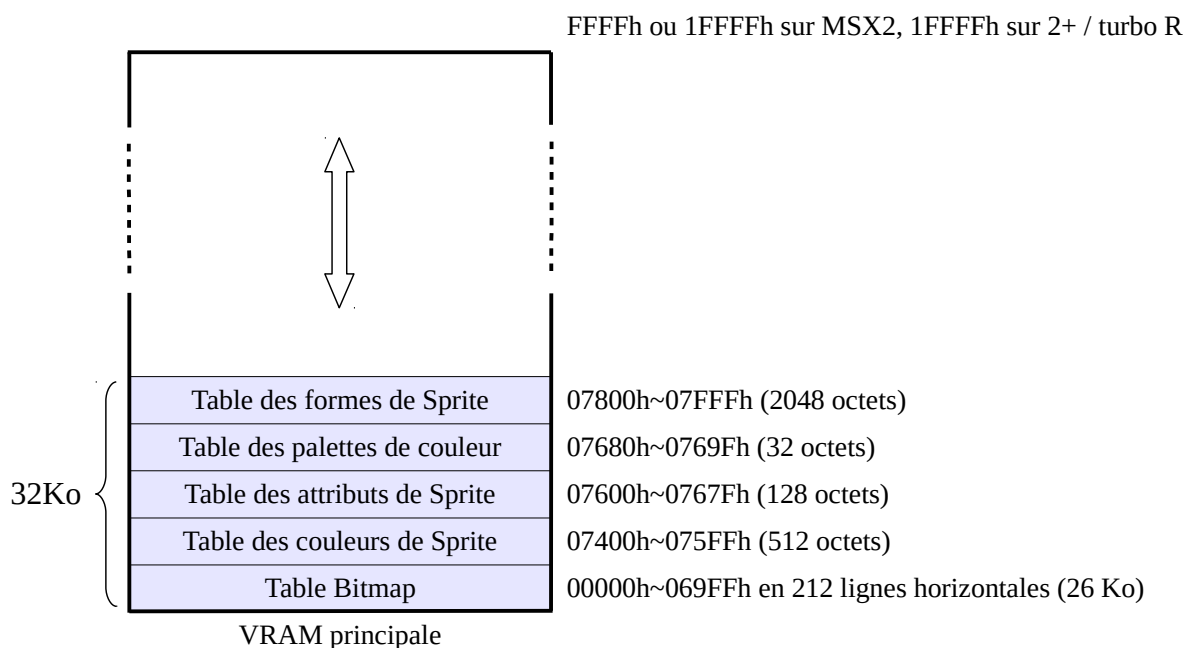
N16 et N15 codent les deux bits de poids fort de l'adresse de la table Bitmap à afficher. Une table Bitmap peut donc se trouver seulement aux adresses 00000h, 08000h, 10000h et 18000h lorsqu'il y a 128kB de VRAM. Pour simplifier, ces 2 bits sont assimilés à un numéro de page (0 à 3) par le MSX-Basic.

Le bit 7 (LN) du registre 9 est mis à 1 à l'initialisation du mode par le système pour afficher un écran sur 212 lignes horizontales. Vous pouvez le mettre à 0 pour réduire l'affichage à 192 lignes comme dans les modes d'écran MSX1. Ainsi, la table Bitmap a une taille 23 Ko (00000h~05FFFh), ce qui laisse plus de place pour stocker des décors ou des Sprite par exemple.

## SCREEN 6 (mode graphique 5)

Résolution : 512 × 212 / 192.  
Type : Bitmap.  
Couleurs : 4 couleurs parmi 512.  
Sprite : Type 2.  
VDP : V9938 et V9958.

La carte de la VRAM se présente ainsi à l'initialisation :



La table des palettes de couleur est juste une zone où le système y stocke les valeurs de la palette par défaut.

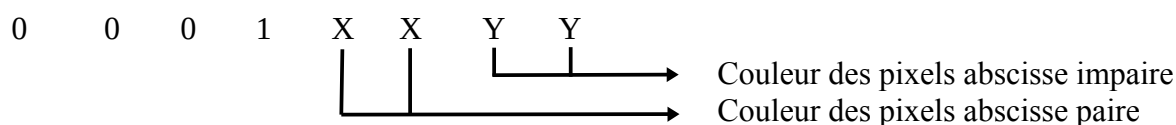
Ce mode d'écran est disponible à partir du MSX2. Dans ce mode, chaque octet de la table Bitmap code 4 pixels. Les bits 6 et 7 (poids fort) définissent la couleur (0 ~ 3) du pixel dont l'abscisse est multiple de 4. Les bits 4 et 5 codent la couleur du pixel immédiatement à la droite de celui dont l'abscisse est multiple de 4, et ainsi de suite...

La table Bitmap du mode SCREEN 6 est codée de la façon suivante :

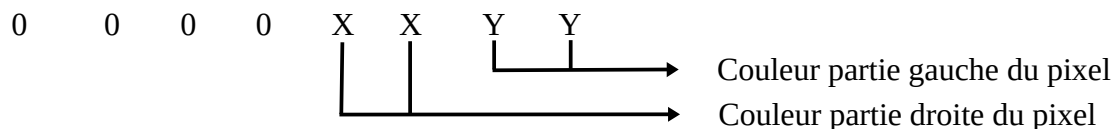
|        | bit 7         | bit 6 | bit 5         | bit 4 | bit 3         | bit 2 | bit 1         | bit 0 |
|--------|---------------|-------|---------------|-------|---------------|-------|---------------|-------|
| 00000h | pixel (0,0)   |       | pixel (1,0)   |       | pixel (2,0)   |       | pixel (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 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 Sprite de la manière suivante :

- pour la marge, préciser la couleur comme suit :



- pour les Sprite, préciser la couleur comme suit :



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

```

## Registres directement impliqués pour définir le mode SCREEN 6 :

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 0 :</b> | 0     | DG    | IE2   | IE1   | M5    | M4    | M3    | EV    | (Tous les VDP) |
| <b>Registre 1 :</b> | 4/16k | BL    | IE0   | M1    | M2    | 0     | SI    | MAG   | (Tous les VDP) |

Les bits M1 à M4 doivent être mis à 0. M5 doit être à 1 pour enclencher le mode SCREEN 6.

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 2 :</b> | 0     | N16   | N15   | N14   | N13   | N12   | N11   | N10   | (Tous les VDP) |

Les bits N10 à N14 doivent être normalement mis à 1. Ils sont mis à 1 à l'initialisation. Ces bits servent à répéter l'affichage de lignes de l'avant-plan de la façon suivante.

- N14 – Mettre ce bit à 0 pour répéter l'affichage des 128 premières lignes à la suite.
- N13 – Mettre ce bit à 0 pour répéter l'affichage des 64 premières lignes à la place des lignes 64 à 127 et répéter l'affichage des lignes 128 à 191 à la place des lignes 192 à 255.
- N12 – Mettre ce bit à 0 pour répéter l'affichage des 32 premières lignes à la place des lignes 32 à 63, répéter l'affichage des lignes 64 à 95 à la place des lignes 96 à 127, répéter l'affichage des lignes 128 à 159 à la place des lignes 160 à 191 et répéter l'affichage des lignes 192 à 231 à la place des lignes 232 à 255.
- N11 – Mettre ce bit à 0 pour répéter l'affichage des 16 premières lignes à la place des lignes 16 à 31, répéter les lignes 32 à 47 à la place des lignes 48 à 63, répéter les lignes 64 à 79 à la place des lignes 80 à 95, répéter les lignes 96 à 111 à la place des lignes 112 à 127, et ainsi de suite...
- N10 – Mettre ce bit à 0 pour répéter l'affichage des 8 premières lignes à la place des lignes 8 à 15, répéter les lignes 16 à 23 à la place des lignes 24 à 31, répéter les lignes 32 à 39 à la place des lignes 40 à 47, répéter les lignes 48 à 57 à la place des lignes 56 à 63, et ainsi de suite...

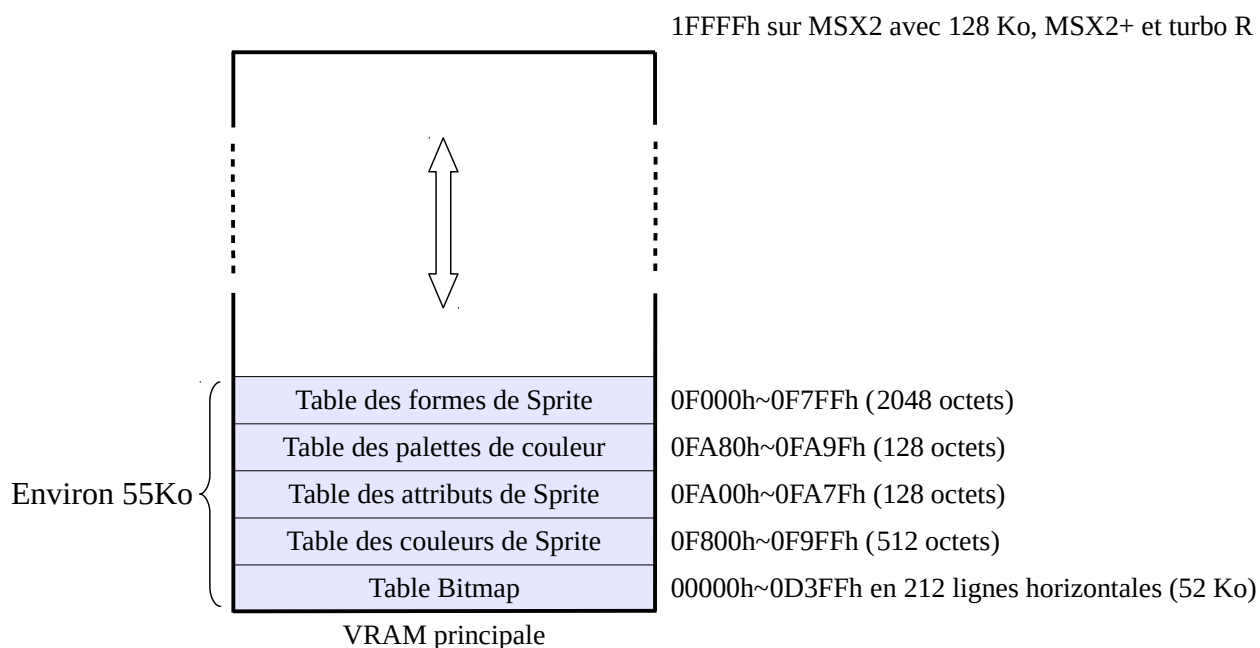
N16 et N15 codent les deux bits de poids fort de l'adresse de la table Bitmap à afficher. Une table Bitmap peut donc se trouver seulement aux adresses 00000h, 08000h, 10000h et 18000h lorsqu'il y a 128kB de VRAM. Pour simplifier, ces 2 bits sont assimilés à un numéro de page (0 à 3) par le MSX-Basic.

Le bit 7 (LN) du registre 9 est mis à 1 à l'initialisation du mode par le système pour afficher un écran sur 212 lignes horizontales. Vous pouvez le mettre à 0 pour réduire l'affichage à 192 lignes comme dans les modes d'écran MSX1. Ainsi, la table Bitmap a une taille 23 Ko (00000h~05FFFh), ce qui laisse plus de place pour stocker des décors ou des Sprite par exemple.

## SCREEN 7 (mode graphique 6)

Résolution : 512 × 192 / 212.  
Type : Bitmap.  
Couleurs : 16 couleurs parmi 512.  
Sprite : Type 2.  
VDP : V9938 avec 128 Ko de VRAM et V9958.

La carte de la VRAM à l'initialisation se présente ainsi :



La table des palettes de couleur est juste une zone où le système y stocke les valeurs de la palette par défaut.

Ce mode d'écran est disponible à partir du MSX2 avec 128Ko de VRAM. 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.

La table Bitmap du SCREEN7 est codée de la façon suivante :

|        | bit 7         | bit 6 | bit 5 | bit 4 | bit 3         | bit 2 | bit 1 | bit 0 |
|--------|---------------|-------|-------|-------|---------------|-------|-------|-------|
| 00000h | pixel (0,0)   |       |       |       | pixel (1,0)   |       |       |       |
| 00001h | pixel (2,0)   |       |       |       | pixel (3,0)   |       |       |       |
| ⋮      | ⋮             |       |       |       | ⋮             |       |       |       |
| 0009Fh | pixel (510,0) |       |       |       | pixel (511,0) |       |       |       |
| 00100h | pixel (0,1)   |       |       |       | pixel (1,1)   |       |       |       |
| ⋮      | ⋮             |       |       |       | ⋮             |       |       |       |

**Registres directement impliqués pour définir le mode SCREEN 7 :**

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 0 :</b> | 0     | DG    | IE2   | IE1   | M5    | M4    | M3    | EV    | (Tous les VDP) |
| <b>Registre 1 :</b> | 4/16k | BL    | IE0   | M1    | M2    | 0     | SI    | MAG   | (Tous les VDP) |

M1, M2 et M4 doivent être mis à 0. M3 et M5 doivent être à 1 pour enclencher le mode SCREEN 7.

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 2 :</b> | 0     | N16   | 0     | N14   | N13   | N12   | N11   | N10   | (Tous les VDP) |

Les bits N10 à N14 doivent être normalement mis à 1. Ils sont mis à 1 à l'initialisation. Ces bits servent à répéter l'affichage de lignes de l'avant-plan de la façon suivante.

- N14 – Mettre ce bit à 0 pour répéter l'affichage des 128 premières lignes à la suite.
- N13 – Mettre ce bit à 0 pour répéter l'affichage des 64 premières lignes à la place des lignes 64 à 127 et répéter l'affichage des lignes 128 à 191 à la place des lignes 192 à 255.
- N12 – Mettre ce bit à 0 pour répéter l'affichage des 32 premières lignes à la place des lignes 32 à 63, répéter l'affichage des lignes 64 à 95 à la place des lignes 96 à 127, répéter l'affichage des lignes 128 à 159 à la place des lignes 160 à 191 et répéter l'affichage des lignes 192 à 231 à la place des lignes 232 à 255.
- N11 – Mettre ce bit à 0 pour répéter l'affichage des 16 premières lignes à la place des lignes 16 à 31, répéter les lignes 32 à 47 à la place des lignes 48 à 63, répéter les lignes 64 à 79 à la place des lignes 80 à 95, répéter les lignes 96 à 111 à la place des lignes 112 à 127, et ainsi de suite...
- N10 – Mettre ce bit à 0 pour répéter l'affichage des 8 premières lignes à la place des lignes 8 à 15, répéter les lignes 16 à 23 à la place des lignes 24 à 31, répéter les lignes 32 à 39 à la place des lignes 40 à 47, répéter les lignes 48 à 57 à la place des lignes 56 à 63, et ainsi de suite...

N16 codent le bit de poids fort de l'adresse de la table Bitmap à afficher. Une table Bitmap peut donc se trouver seulement aux adresses 00000h et 10000h (table 0 et 1). Pour simplifier, ce bit est assimilé à un numéro de page par le MSX-Basic.

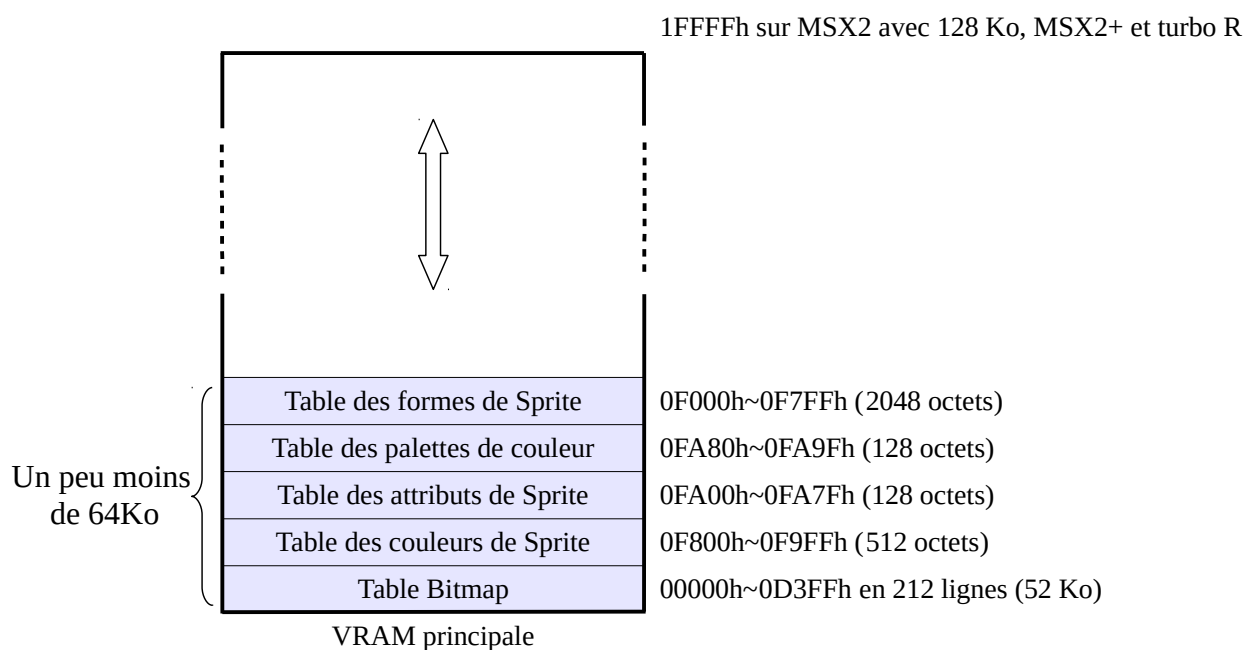
Le bit 7 (LN) du registre 9 est mis à 1 à l'initialisation du mode par le système pour afficher un écran sur 212 lignes horizontales. Vous pouvez le mettre à 0 pour réduire l'affichage à 192 lignes comme dans les modes d'écran MSX1. Ainsi, la table Bitmap a une taille 47 Ko (00000h~0BFFFh), ce qui laisse plus de place pour stocker des décors ou des Sprite par exemple.



## SCREEN 8 (mode graphique 7 avec couleurs indexées)

Résolution : 256 × 192 / 212 bitmap.  
Type : Bitmap.  
Couleurs : 256 couleurs.  
Sprite : Type 2.  
VDP : V9938 avec 128 Ko de VRAM et V9958.

La carte de la VRAM à l'initialisation se présente ainsi :



La table des palettes de couleur est juste une zone où le système y stocke les valeurs de la palette par défaut.

Ce mode d'écran est disponible à partir du MSX2 avec 128Ko de VRAM. Dans ce mode, chaque octet de la table Bitmap définit la couleur d'un pixel (0-255).

La couleur du mode SCREEN 8 est déterminée de la manière suivante :

|         | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---------|-------|-------|-------|-------|-------|-------|-------|-------|
| 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).

Le numéro de la couleur se calcule avec la formule :  $C = 32 \times G + 4 \times R + B$

### Registres directement impliqués pour définir le mode SCREEN 8 :

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 0 :</b> | 0     | DG    | IE2   | IE1   | M5    | M4    | M3    | EV    | (Tous les VDP) |
| <b>Registre 1 :</b> | 4/16k | BL    | IE0   | M1    | M2    | 0     | SI    | MAG   | (Tous les VDP) |

M1 et M2 doivent être mis à 0. M3 à M5 doivent être à 1 pour enclencher le mode SCREEN 8.

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 2 :</b> | 0     | N16   | 0     | N14   | N13   | N12   | N11   | N10   | (Tous les VDP) |

Les bits N10 à N14 doivent être normalement mis à 1. Ils sont mis à 1 à l'initialisation. Ces bits servent à répéter l'affichage de lignes de l'avant-plan de la façon suivante.

- N14 – Mettre ce bit à 0 pour répéter l'affichage des 128 premières lignes à la suite.
- N13 – Mettre ce bit à 0 pour répéter l'affichage des 64 premières lignes à la place des lignes 64 à 127 et répéter l'affichage des lignes 128 à 191 à la place des lignes 192 à 255.
- N12 – Mettre ce bit à 0 pour répéter l'affichage des 32 premières lignes à la place des lignes 32 à 63, répéter l'affichage des lignes 64 à 95 à la place des lignes 96 à 127, répéter l'affichage des lignes 128 à 159 à la place des lignes 160 à 191 et répéter l'affichage des lignes 192 à 231 à la place des lignes 232 à 255.
- N11 – Mettre ce bit à 0 pour répéter l'affichage des 16 premières lignes à la place des lignes 16 à 31, répéter les lignes 32 à 47 à la place des lignes 48 à 63, répéter les lignes 64 à 79 à la place des lignes 80 à 95, répéter les lignes 96 à 111 à la place des lignes 112 à 127, et ainsi de suite...
- N10 – Mettre ce bit à 0 pour répéter l'affichage des 8 premières lignes à la place des lignes 8 à 15, répéter les lignes 16 à 23 à la place des lignes 24 à 31, répéter les lignes 32 à 39 à la place des lignes 40 à 47, répéter les lignes 48 à 57 à la place des lignes 56 à 63, et ainsi de suite...

N16 codent le bit de poids fort de l'adresse de la table Bitmap à afficher. Une table Bitmap peut donc se trouver seulement aux adresses 00000h et 10000h (table 0 et 1). Pour simplifier, ce bit est assimilé à un numéro de page par le MSX-Basic.

Le bit 7 (LN) du [registre 9](#) est mis à 1 à l'initialisation du mode par le système pour afficher un écran sur 212 lignes horizontales. Vous pouvez le mettre à 0 pour réduire l'affichage à 192 lignes comme dans les modes d'écran MSX1. Ainsi, la table Bitmap a une taille 47 Ko (00000h~0BFFFh), ce qui laisse plus de place pour stocker des décors ou des Sprite par exemple.

## SCREEN 9 (mode graphique 4 ou 5)

Ce mode d'écran est utilisé par le mode « Hangul » des MSX2 coréens. Il permet d'afficher les caractères coréens. Ce mode est en fait le SCREEN 5 ou 6 selon le nombre de colonne affichable.

## SCREEN 10 et 11 (mode graphique 7 en YJK et RGB)

Résolution : 256 × 192 / 212.  
Type : Semi-bitmap.  
Couleurs : 12499 couleurs codées en YJK sur 4 octets et 16 couleurs codées en RGB.  
Sprite : Type 2.  
VDP : V9958.

La carte de la VRAM est la même que celle des modes SCREEN 7 et 8.

Ces modes d'écran sont disponibles à partir du MSX2+. Ces modes d'écran sont en fait les mêmes. La seule différence sont les instructions graphiques du basic qui travaillent en 16 couleurs RGB pour le SCREEN 10 et en 12499 couleurs YJK pour le SCREEN 11.

La couleur de chacun des 4 pixels alignés horizontalement est déterminée de la manière suivante :

|                   | bit 7          | bit 6 | bit 5 | bit 4 | bit 3          | bit 2                     | bit 1 | bit 0 |
|-------------------|----------------|-------|-------|-------|----------------|---------------------------|-------|-------|
| Premier octet :   | Y <sub>1</sub> |       |       |       | A <sub>1</sub> | bits de poids faible de K |       |       |
| Second octet :    | Y <sub>2</sub> |       |       |       | A <sub>2</sub> | bits de poids fort de K   |       |       |
| Troisième octet : | Y <sub>3</sub> |       |       |       | A <sub>3</sub> | bits de poids faible de J |       |       |
| Quatrième octet : | Y <sub>4</sub> |       |       |       | A <sub>4</sub> | bits de poids fort de J   |       |       |

La couleur du premier pixel est codée par Y<sub>1</sub>, J et K. Le second par Y<sub>2</sub>, J et K, le troisième par Y<sub>3</sub>, J et K puis le quatrième par Y<sub>4</sub>, J et K. Et ainsi de suite... Lorsque l'attribut d'un octet est à 1, J et K sont ignorés. Y<sub>1</sub> ~ Y<sub>4</sub> devient le numéro de couleur du pixel correspondant comme en SCREEN 5.

La seule différence entre le SCREEN 10 et le 11, c'est la gestion des attributs de la table YAE. En SCREEN 10, les attributs correspondants aux pixels tracés sont mis à 1. Il faut passer de l'un à l'autre pour tracer des pixels codés en RGB ou en YJK. Le passage du SCREEN 10 à 11 n'efface pas l'écran.

## Registres directement impliqués pour définir le mode SCREEN 10 /11 :

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 0 :</b> | 0     | DG    | IE2   | IE1   | M5    | M4    | M3    | EV    | (Tous les VDP) |
| <b>Registre 1 :</b> | 4/16k | BL    | IE0   | M1    | M2    | 0     | SI    | MAG   | (Tous les VDP) |

Tout comme pour le SCREEN 8, les bits M3 à M5 du registres 0 doivent être mis à 1 et M1 et M2 à 0.

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 2 :</b> | 0     | N16   | 0     | N14   | N13   | N12   | N11   | N10   | (Tous les VDP) |

Les bits N10 à N14 doivent être normalement mis à 1. Ils sont mis à 1 à l'initialisation. Ces bits servent à répéter l'affichage de lignes de l'avant-plan de la façon suivante.

- N14 – Mettre ce bit à 0 pour répéter l'affichage des 128 premières lignes à la suite.
- N13 – Mettre ce bit à 0 pour répéter l'affichage des 64 premières lignes à la place des lignes 64 à 127 et répéter l'affichage des lignes 128 à 191 à la place des lignes 192 à 255.
- N12 – Mettre ce bit à 0 pour répéter l'affichage des 32 premières lignes à la place des lignes 32 à 63, répéter l'affichage des lignes 64 à 95 à la place des lignes 96 à 127, répéter l'affichage des lignes 128 à 159 à la place des lignes 160 à 191 et répéter l'affichage des lignes 192 à 231 à la place des lignes 232 à 255.
- N11 – Mettre ce bit à 0 pour répéter l'affichage des 16 premières lignes à la place des lignes 16 à 31, répéter les lignes 32 à 47 à la place des lignes 48 à 63, répéter les lignes 64 à 79 à la place des lignes 80 à 95, répéter les lignes 96 à 111 à la place des lignes 112 à 127, et ainsi de suite...
- N10 – Mettre ce bit à 0 pour répéter l'affichage des 8 premières lignes à la place des lignes 8 à 15, répéter les lignes 16 à 23 à la place des lignes 24 à 31, répéter les lignes 32 à 39 à la place des lignes 40 à 47, répéter les lignes 48 à 57 à la place des lignes 56 à 63, et ainsi de suite...

N16 codent le bit de poids fort de l'adresse de la table YEA à afficher. Une table Bitmap peut donc se trouver seulement aux adresses 00000h et 10000h (table 0 et 1). Pour simplifier, ce bit est assimilé à un numéro de page par le MSX-Basic.

|                      | bit 7 | bit 6 | bit 5                   | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 | (Tous les VDP) |
|----------------------|-------|-------|-------------------------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 25 :</b> | 0     | CMD   | $\overline{\text{VDS}}$ | YAE   | YJK   | WTE   | MSK   | SP2   | (V9958)        |

Les bits YJK et YAE doivent être mis à 1 pour passer de la table Bitmap du SCREEN 8 à la table YAE du SCREEN 10/11.

Le bit 7 (LN) du [registre 9](#) est mis à 1 à l'initialisation du mode par le système pour afficher un écran sur 212 lignes horizontales. Vous pouvez le mettre à 0 pour réduire l'affichage à 192 lignes comme dans les modes d'écran MSX1. Ainsi, la table YEA a une taille 47 Ko (00000h~0BFFFh), ce qui laisse plus de place pour stocker des décors ou des Sprite par exemple.

Résolution : 256 × 192 / 212.  
 Type : YJK.  
 Couleurs : 19268 couleurs codées en YJK sur 4 octets pour 4 pixels.  
 Sprite : Type 2.  
 VDP : V9958.

La carte de la VRAM est la même que celle des modes SCREEN 7 et 8.

Ce mode d'écran est disponible à partir du MSX2+. Dans ce mode d'écran les couleurs sont codées en YJK. Ce qui permet d'afficher en 19268 couleurs simultanément au prix de contraintes sur chaque 4 pixels alignés horizontalement.

La couleur de chacun des 4 pixels alignés horizontalement est déterminée de la manière suivante :

|                   | bit 7          | bit 6 | bit 5 | bit 4 | bit 3 | bit 2                     | bit 1 | bit 0 |
|-------------------|----------------|-------|-------|-------|-------|---------------------------|-------|-------|
| Premier octet :   | Y <sub>1</sub> |       |       |       |       | bits de poids faible de K |       |       |
| Second octet :    | Y <sub>2</sub> |       |       |       |       | bits de poids fort de K   |       |       |
| Troisième octet : | Y <sub>3</sub> |       |       |       |       | bits de poids faible de J |       |       |
| Quatrième octet : | Y <sub>4</sub> |       |       |       |       | bits de poids fort de J   |       |       |

La couleur du premier octet est codée par Y<sub>1</sub>, J et K. Le second par Y<sub>2</sub>, J et K, le troisième par Y<sub>3</sub>, J et K puis le quatrième par Y<sub>4</sub>, J et K. Et ainsi de suite...

Formules de conversion :

$$\begin{aligned}
 R &= Y + JG = Y + K & B &= 1,25Y - 0,5J - 0,25K \\
 Y &= (2R + G + 4B) / 8 & J &= (6R - G + 4B) / 8 & K &= (-2R + 7G - 4B) / 8
 \end{aligned}$$

## Registres directement impliqués pour définir le mode SCREEN 12 :

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 0 :</b> | 0     | DG    | IE2   | IE1   | M5    | M4    | M3    | EV    | (Tous les VDP) |
| <b>Registre 1 :</b> | 4/16k | BL    | IE0   | M1    | M2    | 0     | SI    | MAG   | (Tous les VDP) |

Tout comme pour le SCREEN 8, les bits M3 à M5 du registres 0 doivent être mis à 1 et M1 et M2 à 0.

|                     | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                |
|---------------------|-------|-------|-------|-------|-------|-------|-------|-------|----------------|
| <b>Registre 2 :</b> | 0     | N16   | 0     | N14   | N13   | N12   | N11   | N10   | (Tous les VDP) |

Les bits N10 à N14 doivent être normalement mis à 1. Ils sont mis à 1 à l'initialisation. Ces bits servent à répéter l'affichage de lignes de l'avant-plan de la façon suivante.

N14 – Mettre ce bit à 0 pour répéter l'affichage des 128 premières lignes à la suite.

N13 – Mettre ce bit à 0 pour répéter l'affichage des 64 premières lignes à la place des lignes 64 à 127 et répéter l'affichage des lignes 128 à 191 à la place des lignes 192 à 255.

N12 – Mettre ce bit à 0 pour répéter l'affichage des 32 premières lignes à la place des lignes 32 à 63, répéter l'affichage des lignes 64 à 95 à la place des lignes 96 à 127, répéter l'affichage des lignes 128 à 159 à la place des lignes 160 à 191 et répéter l'affichage des lignes 192 à 231 à la place des lignes 232 à 255.

N11 – Mettre ce bit à 0 pour répéter l'affichage des 16 premières lignes à la place des lignes 16 à 31, répéter les lignes 32 à 47 à la place des lignes 48 à 63, répéter les lignes 64 à 79 à la place des lignes 80 à 95, répéter les lignes 96 à 111 à la place des lignes 112 à 127, et ainsi de suite...

N10 – Mettre ce bit à 0 pour répéter l'affichage des 8 premières lignes à la place des lignes 8 à 15, répéter les lignes 16 à 23 à la place des lignes 24 à 31, répéter les lignes 32 à 39 à la place des lignes 40 à 47, répéter les lignes 48 à 57 à la place des lignes 56 à 63, et ainsi de suite...

N16 codent le bit de poids fort de l'adresse de la table YJK à afficher. Une table Bitmap peut donc se trouver seulement aux adresses 00000h et 10000h (table 0 et 1). Pour simplifier, ce bit est assimilé à un numéro de page par le MSX-Basic.

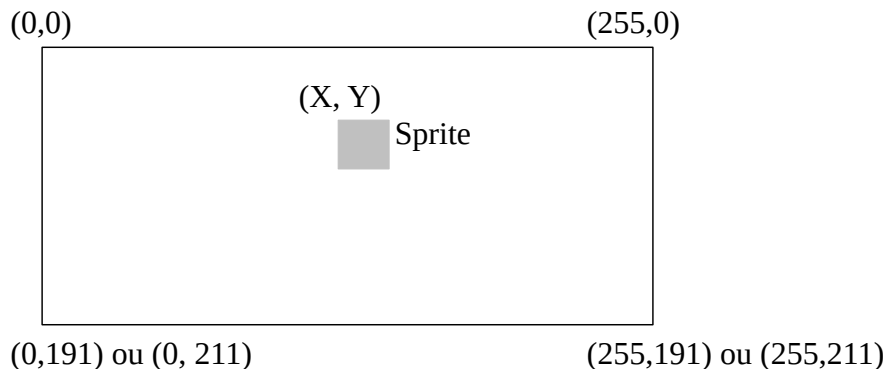
|                      | bit 7 | bit 6 | bit 5                   | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |                                             |
|----------------------|-------|-------|-------------------------|-------|-------|-------|-------|-------|---------------------------------------------|
| <b>Registre 25 :</b> | 0     | CMD   | $\overline{\text{VDS}}$ | YAE   | YJK   | WTE   | MSK   | SP2   | (Tous les VDP)<br>( <a href="#">V9958</a> ) |

Le bit YJK doit être mis à 1 et le bit YAE à 0 pour passer de la table Bitmap du SCREEN 8 à la table YJK du SCREEN 12.

Le bit 7 (LN) du [registre 9](#) est mis à 1 à l'initialisation du mode par le système pour afficher un écran sur 212 lignes horizontales. Vous pouvez le mettre à 0 pour réduire l'affichage à 192 lignes comme dans les modes d'écran MSX1. Ainsi, la table YJK a une taille 47 Ko (00000h~0BFFFh), ce qui laisse plus de place pour stocker des décors ou des Sprite par exemple.

## 7.9 Les Sprite et leur fonctionnement

Les Sprite, ou lutins graphiques, sont des motifs graphiques, des sortes de petits calques, pouvant être affichés à n'importe quel pixel en se superposant à l'avant-plan de l'écran, sans l'effacer. Ces facilités les prédestinent au mouvement puisqu'il suffit de modifier deux octets (les coordonnées du Sprite) pour les faire se déplacer sans toucher au reste de l'écran. Le VDP peut mémoriser jusqu'à 256 formes de Sprite, il peut en afficher simultanément au maximum 32 à l'écran avec une limite de 4 ou 8 par ligne selon le mode d'écran utilisé. Les Sprite se trouvent définis dans une matrice  $8 \times 8$  ou  $16 \times 16$ . 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 Sprite 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
```

Depuis les ordinateurs MSX2, il existe deux types de fonctionnement de Sprite. Le VDP choisit automatiquement le type adéquat suivant le mode écran. En SCREEN 1, 2 et 3, ce sont des Sprite de type 1 (qui correspond à ceux des ordinateurs MSX1) alors qu'en SCREEN 4 à 8 ou 12, ce sont des Sprite de type 2 de Sprite.



## Sprite de type 1

(MSX1~)

### Fonctionnement :

Il peut y avoir 32 Sprite de type 1 à l'écran numérotés de 0 à 31. Chacun ont une couleur unique. Plus un Sprite a un numéro faible, plus son affichage est prioritaire. Ceci signifie que si le Sprite n°3 croise le Sprite n°16, on verra le Sprite n°3 passer devant (lorsque des pixels des deux Sprite se trouveront aux mêmes coordonnées, c'est les pixels du Sprite n°3 qui seront affichés).

Quatre Sprite, au maximum, peuvent être affichés sur une même ligne de l'écran. A partir du 5<sup>e</sup> Sprite, toutes les portions communes aux Sprite sur une même ligne disparaissent sur le(s) Sprite de plus faible priorité.

### Paramètres :

|                     |                                                                                                                                                       |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| Matrice des Sprite. | Le bit 1 (SI) du registre 1 du VDP définit la taille de la matrice.<br>SI à 0 pour Sprite de 8 × 8 pixels.<br>SI à 1 pour Sprite de 16 × 16 pixels.   |
| Taille des Sprite.  | Le bit 0 (MAG) du registre 1 du VDP permet de doubler la taille.<br>MAG à 0 pour taille des pixels normale.<br>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 Sprite entrent en collision (deux pixels ou plus avec les mêmes coordonnées), le bit 5 du registre de statut 0 passe à 1.

### 5 Sprite sur une même ligne :

Lorsque plus de 4 Sprite se trouvent sur la même ligne de l'écran, 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 5e Sprite.

### Utilisation :

Pour afficher un Sprite, il faut d'abord le définir dans la table des formes de Sprite.

Pour les Sprite 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 Sprite pourrait être :

|        |          | huit octets pour le<br>Sprite n°0<br>qui aura cette forme |  |  |  |  |  |  |  |
|--------|----------|-----------------------------------------------------------|--|--|--|--|--|--|--|
|        | 7 ~ 0    |                                                           |  |  |  |  |  |  |  |
| 03800h | 11111111 |                                                           |  |  |  |  |  |  |  |
| 03801h | 10100001 |                                                           |  |  |  |  |  |  |  |
| 03802h | 10010001 |                                                           |  |  |  |  |  |  |  |
| 03803h | 10001001 |                                                           |  |  |  |  |  |  |  |
| 03804h | 10000101 |                                                           |  |  |  |  |  |  |  |
| 03805h | 10000101 |                                                           |  |  |  |  |  |  |  |
| 03806h | 10001001 |                                                           |  |  |  |  |  |  |  |
| 03807h | 11111111 |                                                           |  |  |  |  |  |  |  |

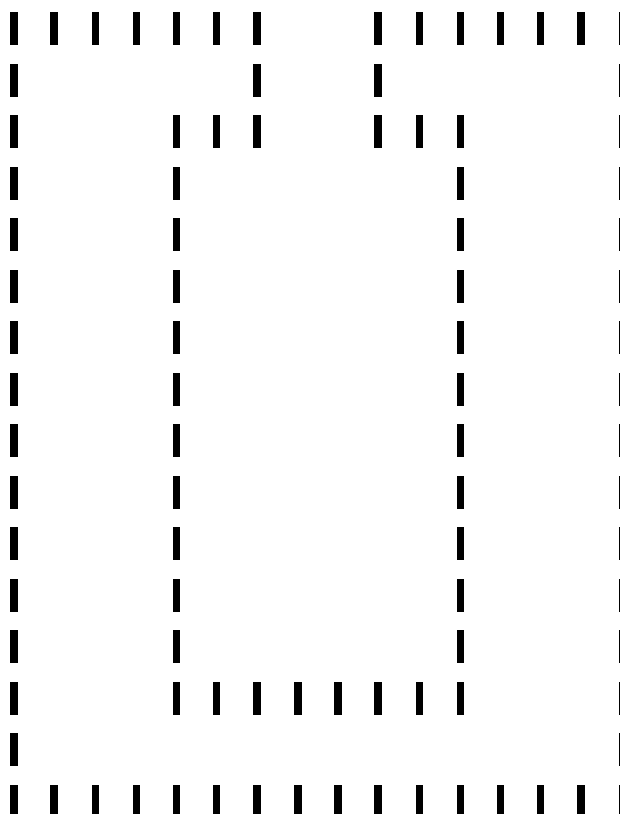
Dans le cas de Sprite de taille 16 x 16, les choses deviennent un tout petit peu plus compliquées. Le Sprite se décompose en 4 parties :

|   |   |
|---|---|
| 0 | 2 |
| 1 | 3 |

|        | 7 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |                                                      |
|--------|---|---|---|---|---|---|---|---|---|---|------------------------------------------------------|
| 03800h | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |   |   |                                                      |
| 03801h | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |   |   |                                                      |
| 03802h | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |   |   |                                                      |
| 03803h | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |   |   |                                                      |
| 03804h | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |   |   | huit octets pour le premier<br>quart du Sprite n°0   |
| 03805h | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |   |   |                                                      |
| 03806h | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |   |   |                                                      |
| 03807h | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |   |   |                                                      |
| 03808h | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |   |   |                                                      |
| 03809h | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |   |   |                                                      |
| 0380Ah | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |   |   |                                                      |
| 0380Bh | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |   |   | huit octets pour le second<br>quart du Sprite n°0    |
| 0380Ch | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |   |   |                                                      |
| 0380Dh | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |   |   |                                                      |
| 0380Eh | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |   |   |                                                      |
| 0380Fh | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |   |   |                                                      |
| 03810h | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |   |   |                                                      |
| 03811h | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |   |   |                                                      |
| 03812h | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |   |   |                                                      |
| 03813h | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |   |   | huit octets pour le troisième<br>quart du Sprite n°0 |
| 03814h | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |   |   |                                                      |
| 03815h | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |   |   |                                                      |
| 03816h | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |   |   |                                                      |
| 03817h | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |   |   |                                                      |
| 03818h | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |   |   |                                                      |
| 03819h | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |   |   |                                                      |
| 0381Ah | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |   |   |                                                      |
| 0381Bh | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |   |   | huit octets pour le dernier<br>quart du Sprite n°0   |
| 0381Ch | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |   |   |                                                      |
| 0381Dh | 1 | 1 | 1 | 0 | 0 | 0 | 1 |   |   |   |                                                      |
| 0381Eh | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |   |   |                                                      |
| 0381Fh | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |   |   |                                                      |

La réunion des quatre parties donne le résultat ci-dessous :

Première forme  
de Sprite (matrice 0)



Une fois le Sprite défini, on peut le manipuler à souhait grâce à la table des attributs de Sprite.

Cette table agit sur les plans graphiques, et compte 4 octets par plan. On appelle « plan graphique » un espace comprenant un élément graphique pouvant se superposer. Il y a 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 Sprite code la position et couleur de chaque plan sur 4 octets ainsi :

|        |                                     |   |   |   |                |   |   |   |                         |
|--------|-------------------------------------|---|---|---|----------------|---|---|---|-------------------------|
|        | 7                                   | 6 | 5 | 4 | 3              | 2 | 1 | 0 |                         |
| 01B00h | Ordonnée du Sprite 0 (0-255)        |   |   |   |                |   |   |   | } Attributs du Sprite 0 |
| 01B01h | Abscisse du Sprite 0 (0-255)        |   |   |   |                |   |   |   |                         |
| 01B02h | Numéro de matrice du Sprite (0-255) |   |   |   |                |   |   |   |                         |
| 01B03h | EC                                  | 0 | 0 | 0 | Couleur (0~15) |   |   |   |                         |
| 01B04h | Ordonnée du Sprite 1 (0-255)        |   |   |   |                |   |   |   |                         |
| ⋮      | ⋮                                   |   |   |   |                |   |   |   |                         |

La valeur de l'ordonnée peut varier entre 0 et 255 mais celles de 223 à 255 sont interprétées en complément à 2 pour permettre de faire entrer un Sprite dans l'écran (223 à 255 pour -32 à 0).

Notez que si l'ordonnée de n'importe quel Sprite est à 208 (0D0h), alors tous les Sprite dans les plans supérieurs deviennent invisibles. Par exemple si l'on met l'ordonnée du Sprite contenu dans 12 à 208, alors tous les Sprite dans les plans 13 à 31 disparaissent.

L'abscisse prend une valeur entre 0 et 255. Pour faire sortir le Sprite, pas de problème, 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 correspondant au Sprite.

Le numéro de matrice du Sprite indique la matrice à utiliser pour afficher le Sprite. On peut définir 256 matrices de 8 x 8, ou 64 de 16 x 16. En taille 8 x 8, on choisit simplement le numéro de celui que l'on désire afficher. Pour les Sprite 16 x 16, on met 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 STRIG(0)=0 THEN 55
```

## **Sprite de type 2** (MSX2~)

### Fonctionnement :

Il peut y avoir 32 Sprite de type 2 à l'écran numérotés de 0 à 31. Ces Sprite peuvent avoir une couleur différente par ligne. 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 devant (lorsque des pixels des deux Sprite se trouveront aux mêmes coordonnées, c'est les pixels du Sprite n°3 qui seront affichés).

Huit Sprite, au maximum, peuvent être affichés sur une même ligne. A partir du 9<sup>e</sup> Sprite, toutes les portions communes aux Sprite sur une même ligne disparaissent sur le(s) Sprite de plus faible priorité.

### Paramètres :

**Matrice des Sprite** Le bit 1 (SI) du registre 1 du VDP définit la taille de la matrice.

SI à 0 pour Sprite de 8 × 8 pixels.

SI à 1 pour Sprite de 16 × 16 pixels.

**Taille des Sprite** 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 Sprite** Le bit 1 (SPD) du registre 8 du VDP permet de désactiver des Sprite.

SPD à 0 = Les Sprite s'affichent. (Sprite désactivés.)

SPD à 1 = Le Sprite ne s'affichent pas. (Sprite activés.)

### Collisions :

Lorsque deux Sprite 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).

|                        | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 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 Sprite sur une même ligne :

Lorsque plus de 8 Sprite 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 9<sup>e</sup> Sprite (0 à 31).

### Utilisation :

Pour afficher un Sprite, il faut d'abord le définir dans la table des formes de Sprite.

Pour les Sprite 8 sur 8, il suffit de 8 octets pour définir la forme d'un Sprite, sachant qu'un bit représente un pixel, par exemple, en SCREEN 7, la table des formes de Sprite pourrait commencer par :

|        | Bits<br>7 ~ 0 | Huit octets pour définir<br>la forme de sprite n°0<br>qui aura celle-ci : |
|--------|---------------|---------------------------------------------------------------------------|
| 0F000h | 11111111      | █ █ █ █ █ █ █ █                                                           |
| 0F001h | 11000011      | █ █                    █ █                                                |
| 0F002h | 10100101      | █       █               █ █                                               |
| 0F003h | 10011001      | █               █ █       █ █                                             |
| 0F004h | 10011001      | █               █ █       █ █                                             |
| 0F005h | 10100101      | █       █               █ █                                               |
| 0F006h | 11000011      | █ █                    █ █                                                |
| 0F007h | 11111111      | █ █ █ █ █ █ █ █                                                           |

Dans le cas de Sprite 16 x 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 |   |   |   |   |   |   |   |   |
| 0F001h | 10000010 |   |   |   |   |   |   |   |   |
| 0F002h | 10001110 |   |   |   |   |   |   |   |   |
| 0F003h | 10001000 |   |   |   |   |   |   |   |   |
| 0F004h | 10001000 |   |   |   |   |   |   |   |   |
| 0F005h | 10001000 |   |   |   |   |   |   |   |   |
| 0F006h | 10001000 |   |   |   |   |   |   |   |   |
| 0F007h | 10001000 |   |   |   |   |   |   |   |   |
| 0F008h | 10001000 |   |   |   |   |   |   |   |   |
| 0F009h | 10001000 |   |   |   |   |   |   |   |   |
| 0F00Ah | 10001000 |   |   |   |   |   |   |   |   |
| 0F00Bh | 10001000 |   |   |   |   |   |   |   |   |
| 0F00Ch | 10001000 |   |   |   |   |   |   |   |   |
| 0F00Dh | 10001111 |   |   |   |   |   |   |   |   |
| 0F00Eh | 10000000 |   |   |   |   |   |   |   |   |
| 0F00Fh | 11111111 |   |   |   |   |   |   |   |   |
| 0F010h | 01111111 |   |   |   |   |   |   |   |   |
| 0F011h | 01000001 |   |   |   |   |   |   |   |   |
| 0F012h | 01110001 |   |   |   |   |   |   |   |   |
| 0F013h | 00010001 |   |   |   |   |   |   |   |   |
| 0F014h | 00010001 |   |   |   |   |   |   |   |   |
| 0F015h | 00010001 |   |   |   |   |   |   |   |   |
| 0F016h | 00010001 |   |   |   |   |   |   |   |   |
| 0F017h | 00010001 |   |   |   |   |   |   |   |   |
| 0F018h | 00010001 |   |   |   |   |   |   |   |   |
| 0F019h | 00010001 |   |   |   |   |   |   |   |   |
| 0F01Ah | 00010001 |   |   |   |   |   |   |   |   |
| 0F01Bh | 00010001 |   |   |   |   |   |   |   |   |
| 0F01Ch | 00010001 |   |   |   |   |   |   |   |   |
| 0F01Dh | 11110001 |   |   |   |   |   |   |   |   |
| 0F01Eh | 00000001 |   |   |   |   |   |   |   |   |
| 0F01Fh | 11111111 |   |   |   |   |   |   |   |   |

Huit octets pour le premier  
quart du Sprite n°0

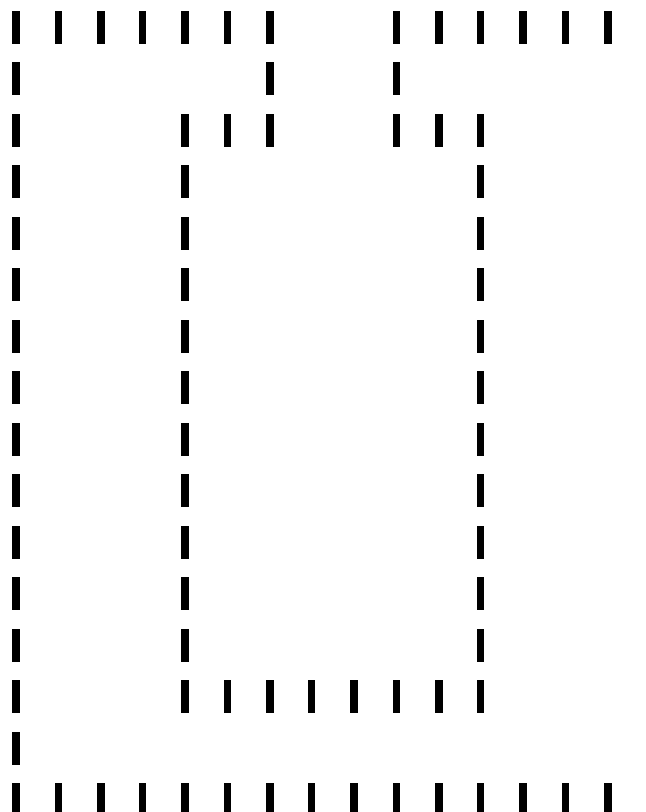
Huit octets pour le second  
quart du Sprite n°0

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 :

Première forme  
de Sprite (matrice 0)



Il faut ensuite remplir la table des attribues de Sprite supplémentaire :

|        | 7  | 6  | 5  | 4 | 3                | 2 | 1 | 0 |
|--------|----|----|----|---|------------------|---|---|---|
| 0F800h | EC | CC | IC | 0 | Couleur ligne 0  |   |   |   |
| 0F801h | EC | CC | IC | 0 | Couleur ligne 1  |   |   |   |
| 0F802h | EC | CC | IC | 0 | Couleur ligne 2  |   |   |   |
| 0F803h | EC | CC | IC | 0 | Couleur ligne 3  |   |   |   |
| ⋮      |    |    |    |   |                  |   |   |   |
| 0F80Eh | EC | CC | IC | 0 | Couleur ligne 14 |   |   |   |
| 0F80Fh | EC | CC | IC | 0 | Couleur ligne 15 |   |   |   |
| 0F810h | EC | CC | IC | 0 | Couleur ligne 0  |   |   |   |
| 0F811h | EC | CC | IC | 0 | Couleur ligne 1  |   |   |   |
| ⋮      |    |    |    |   | ⋮                |   |   |   |

Couleur de  
chaque ligne du  
Sprite numéro 0  
8 x 8 ou  
16 x 16

Idem pour les Sprite suivants

Le bit EC (« Early Clock ») sert à décaler la ligne du Sprite de 32 pixels vers la gauche. Ceci permet de faire « entrer » un Sprite à l'écran par la gauche.

Le bit CC permet de choisir le mode d'affichage des Sprite lors d'une collision :

CC à 0 pour afficher le Sprite comme défini par la table des attribues de Sprite.

CC à 1 pour effectuer un « ou » logique (OR) sur les couleurs des portions de Sprite superposés.



Le bit IC sert à activer la détection de collision :

IC à 0, active la détection.

IC à 1, désactive la détection.

La couleur est toujours codée sur 4 bits, même en mode d'écran 6 (SCREEN 6) (voir le mode d'écran 6 pour plus de précisions, fonction « hardware tiling »).

En SCREEN 8, les valeurs sont fixes et non re-définissables. En voici le détail :

| 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 Sprite 8 sur 8.

Une fois la forme d'un Sprite défini, on peut le manipuler à souhait grâce à la table des attributs de Sprite.

Cette table est composée de 4 octets par Sprite. Soit au total 128 octets pour 32 Sprite affichables (numérotés de 0 à 31) par dessus les plans graphiques, puisqu'il est parfois possible d'afficher un plan avec une entrée vidéo.

La table des attributs de Sprite code la position de chaque Sprite sur 4 octets ainsi :

|        | 7                                   | 6 | 5 | 4 | 3 | 2 | 1 | 0 |                            |
|--------|-------------------------------------|---|---|---|---|---|---|---|----------------------------|
| 01B00h | Ordonnée du Sprite 0 (0~255)        |   |   |   |   |   |   |   | } Attributs<br>du Sprite 0 |
| 01B01h | Abscisse du Sprite 0 (0~255)        |   |   |   |   |   |   |   |                            |
| 01B02h | Numéro de matrice du Sprite (0~255) |   |   |   |   |   |   |   |                            |
| 01B03h | ---- réservé au système ----        |   |   |   |   |   |   |   |                            |
| 01B04h | Ordonnée du Sprite 1 (0-255)        |   |   |   |   |   |   |   |                            |
| ⋮      | ⋮                                   |   |   |   |   |   |   |   |                            |

La valeur de l'ordonnée peut varier entre 0 et 255 mais celles de 223 à 255 sont interprétées en complément à 2 pour permettre de faire entrer un Sprite dans l'écran (223 à 255 pour -32 à 0).

Notez que si l'ordonnée d'un Sprite est à 216 (0D8h), alors tous les Sprite supérieurs deviennent invisibles. Par exemple si l'on place le Sprite 8 sur l'ordonnée 216, alors tous les Sprite de 9 à 31 disparaissent. Ceci est un point important lorsque vous utilisez le registre 23 pour décaler la zone visible de l'écran.

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 la matrice du Sprite donne la matrice à utiliser pour l'affichage. On peut définir 256 Sprite de 8 x 8, ou 64 de 16 x 16. En taille 8 sur 8, on choisit simplement le numéro de celle que l'on désire afficher. Pour les Sprite 16 x 16, on donne le numéro de Sprite multiplié par quatre.

## 7.10 *A propos de la souris et du crayon optique*

Bien que le V9938 soit capable de gérer la souris et le crayon optique, ces deux périphériques de pointage ne sont pas gérés par le processeur vidéo sur MSX (excepté par les ordinateurs MSX2 coréens Daewoo CPC-300, CPC-400 et CPC400S pour le crayon optique). Ces fonctions ont d'ailleurs été retirées du V9958. Pour gérer la souris et le crayon optique, il est fortement conseillé d'utiliser la routine du Bios **GTPAD** (000DBh en Main-ROM) ou **NEWPAD** (001ADh en Sub-ROM) ([voir le chapitre 3](#) page 130).

## 8 Le processeur sonore PSG

Le PSG est un générateur de son programmable fabriqué par la firme General Instrument (AY-3-8910) ou par Yamaha (YM2149). L'un ou l'autre équipe tous les MSX. Le PGG est souvent intégré dans le « MSX-Engine » sur les MSX récents.

### 8.1 Caractéristiques

Sur MSX, le PSG n'est pas seulement utilisé pour faire de la musique et des sons. Il sert aussi à contrôler les ports généraux (manette de jeu, souris, Trackball, molettes, tablette graphique, etc). Il donne aussi l'état de la lecture sur cassette et celui du mode Kana ou JIS des MSX japonais.

Pour la musique et les sons, le PSG dispose de trois canaux indépendants dotés chacun d'un générateur de son et d'un générateur de bruit « blanc » mixable et dont l'enveloppe est paramétrable. Notez que le PSG ne produit pas le cliquetis des touches du clavier ni le son Beep.

Le PSG est cadencé à une fréquence de 1,7897725Mhz en interne.

### 8.2 Descriptions du PSG et accès aux registres

Le PSG possède 16 registres dans lesquels on peut lire et écrire le contenu excepté le registre 14 qu'on ne peut que lire. Pour accéder aux registres du PSG, le Bios en Main-ROM dispose de deux routines. La première est **WRTPSG** (00093h) qui permet de paramétrer le PSG. La seconde, **RDPSG** (00096h), sert à lire contenu d'un registre.

Le Bios en Main-ROM dispose aussi d'une routine, appelée **GICINI** (00090h), qui sert à initialiser le PSG et les données de l'instruction PLAY.

Une fois initialisé, tous les registres de 0 à 13 seront mis à zéro sauf le registre 0 qui aura la valeur 01010101b (55h), le registre 7 qui aura 10111000b (B8h) et le registre 11 qui aura 1011b (0BH) pour la période de l'enveloppe.

Les registres du PSG sont aussi accessible directement par les ports E/S : 0A0h, 0A1h et 0A2h.

Pour écrire dans un registre directement par les ports E/S, il faut écrire le numéro de registre au port 0A0h puis la valeur à écrire au port 0A1h. Prenez soin de couper les interruptions pendant l'écriture du numéro de registre et de la valeur.

Pour lire un registre par les ports E/S, il faut écrire le numéro de registre à lire au port 0A0h puis lire la valeur au port 0A1h.

### 8.3 Registres de contrôle de fréquence (Registres 0~5)

Les six premiers registres servent à paramétrer la fréquence à générer pour produire un son.

|              | bit 7                                               | bit 6 | bit 5 | bit 4 | bit 3                               | bit 2 | bit 1 | bit 0 |
|--------------|-----------------------------------------------------|-------|-------|-------|-------------------------------------|-------|-------|-------|
| Registre 0 : | 8 bits de poids faible de la fréquence de la voix 1 |       |       |       |                                     |       |       |       |
| Registre 1 : | -                                                   | -     | -     | -     | 4 bits forts de la fréquence voix 1 |       |       |       |
| Registre 2 : | 8 bits de poids faible de la fréquence de la voix 2 |       |       |       |                                     |       |       |       |
| Registre 3 : | -                                                   | -     | -     | -     | 4 bits forts de la fréquence voix 2 |       |       |       |
| Registre 4 : | 8 bits de poids faible de la fréquence de la voix 3 |       |       |       |                                     |       |       |       |
| Registre 5 : | -                                                   | -     | -     | -     | 4 bits forts de la fréquence voix 3 |       |       |       |

La valeur indiquant la fréquence tient sur 12 bits. Elle est donc répartie sur deux registres. La valeur à écrire s'obtient à l'aide de la formule suivante.

$$\text{Valeur} = \frac{F_i}{16 \times F_n}$$

$F_i$  = Fréquence interne du PSG (1789772,5 Hz)

$F_n$  = Fréquence du son à produire (varie entre 27 et 111960 Hz)

Pour simplifier, voici un tableau de notes de musique obtenues en fonction de la valeur indiquée aux registres 0~1, 2~3 ou 4~5.

| Octave<br>Note | 1    | 2    | 3    | 4     | 5    | 6    | 7    | 8    |
|----------------|------|------|------|-------|------|------|------|------|
| Do (C)         | D5Dh | 6AFh | 357h | 1ACh  | 0D6h | 06Bh | 035h | 01Bh |
| Do# (C#)       | C9Ch | 64Eh | 327h | 194h  | 0CAh | 065h | 032h | 019h |
| Ré (D)         | BE7h | 5F4h | 2FAh | 17Dh  | 0BEh | 05Fh | 030h | 018h |
| Ré# (D#)       | B3Ch | 59Eh | 2CFh | 168h  | 0B4h | 05Ah | 02Dh | 016h |
| Mi (E)         | A9Bh | 54Eh | 2A7h | 153h  | 0AAh | 055h | 02Ah | 015h |
| Fa (F)         | A02h | 501h | 281h | 140h  | 0A0h | 050h | 028h | 014h |
| Fa# (F#)       | 973h | 4BAh | 25Dh | 12Eh  | 097h | 04Ch | 026h | 013h |
| Sol (G)        | 8EBh | 476h | 23Bh | 11Dh  | 08Fh | 047h | 024h | 012h |
| Sol# (G#)      | 86Bh | 436h | 21Bh | 10Dh  | 087h | 043h | 022h | 011h |
| La (A)         | 7F2h | 3F9h | 1FDh | 0FEh* | 05Fh | 040h | 020h | 010h |
| La# (A#)       | 780h | 3C0h | 1E0h | 0F0h  | 078h | 03Ch | 01Eh | 00Fh |
| Si (B)         | 714h | 38Ah | 1C5h | 0E3h  | 051h | 039h | 01Ch | 00Eh |

(\*) 0FEh est la note produite par les diapasons.

Par exemple, pour produire la note Do sur l'octave 4 par la voix 1, nous devons écrire 1h dans le registre 1 et ACh dans le registre 0. En pratique, ça donne ceci :

## En assembleur :

```
WRTPSG    equ    00093h

; == 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'exécution du programme

; ---
    org    0c000h
START:
    ld     b,13
PSGini:   ld     a,b      ;
    ld     e,0          ; 8 bits de poids faible
    cp     7
    jr     nz,NoR7      ; Saut si registre différent de 7
    ld     e,10111111b  ; Le bit 7 à 1 et le bit 6 à 0
NoR7:     call    WRTPSG
    djnz   PSGini       ; boucle pour initialiser les registres

    ld     a,0          ; Registre 0
    ld     e,0ach       ; 8 bits de poids faible
    call   WRTPSG
    ld     a,1          ; Registre 1
    ld     e,1          ; 4 bits de poids fort
    call   WRTPSG
    ld     a,8          ; Registre 8
    ld     e,1100b      ; Volume de la voix 1 à 12
    call   WRTPSG
    ld     a,7          ; Registre 7
    ld     e,10111110b  ; Active la voix 1
    call   WRTPSG
    ret

END:
```

Une fois assemblé et sauvegardé sous le nom « V1501C.BIN », exécuter la routine avec l'instruction suivante.

```
BLOAD"V1501C.BIN",R
```

## En Basic :

```
5 ' Initialise les registres sonores du PSG
10 FOR R=0 TO 13
20 IF R=7 THEN SOUND R,&B10111111 ELSE SOUND R,0
30 NEXT
40 ' Joue la note Do sur la voix 1 avec un volume à 12
50 SOUND 0,&hAC ' 8 bits de poids faible dans registre 0
60 SOUND 1,1 ' 4 bits de poids fort dans registre 1
70 SOUND 8,&b1100 ' Réglage du volume de la voix 1 à 12
80 SOUND 7,&b10111110 ' Active le générateur sonore sur la voix 1
```

## 8.4 Registre de contrôle de la fréquence de bruit blanc (Registre 6)

|              | bit 7 | bit 6 | bit 5 | bit 4                                  | bit 3 | bit 2 | bit 1 | bit 0 |
|--------------|-------|-------|-------|----------------------------------------|-------|-------|-------|-------|
| Registre 6 : | -     | -     | -     | Fréquence du générateur de bruit blanc |       |       |       |       |

La valeur indiquant la fréquence du générateur de bruit blanc tient sur 5 bits. La valeur à écrire s'obtient à l'aide de la formule suivante. Seuls les 5 bits de la valeur sont à écrire dans le registre 6.

$$\text{Valeur} = \frac{F_i}{16 \times F_b}$$

$F_i$  = Fréquence interne du PSG (1789772,5 Hz)

$F_b$  = Fréquence de base du bruit à produire (varie entre 27 et 3608 Hz)

## 8.5 Registre de contrôle des voix et des ports d'E/S du PSG (Registre 7)

Le registre sert à activer ou désactiver le générateur de son ainsi que le générateur de bruit. Il sert aussi à régler le sens des ports d'entrées / sorties du PSG.

|              | bit 7              | bit 6 | bit 5                  | bit 4  | bit 3  | bit 2                | bit 1  | bit 0  |
|--------------|--------------------|-------|------------------------|--------|--------|----------------------|--------|--------|
| Registre 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 |

Notes :

- Pour couper le son sur une voix, on peut mettre le volume de la voix à 0 (registres 8 à 10) ou désactiver le générateur sonore et de bruit de cette voix en mettant les bits correspondants du registre 7 à 1.
- Afin de garantir le bon fonctionnement des ports d'E/S du PSG, le bit 7 du registre 7 doit toujours rester à 1 (port B en mode sortie) et le bit 6 à 0 (port A en mode entrée).
- Il est donc possible d'activer le générateur sonore et le générateur de bruit en même temps sur chaque voix. C'est à dire mixer les deux.

## 8.6 Registres de contrôle de l'amplitude et du volume (Registres 8~10)

|               | bit 7 | bit 6 | bit 5 | bit 4 | bit 3                           | bit 2 | bit 1 | bit 0 |
|---------------|-------|-------|-------|-------|---------------------------------|-------|-------|-------|
| Registre 8 :  | -     | -     | -     | V/A   | Volume / Amplitude de la voix 1 |       |       |       |
| Registre 9 :  | -     | -     | -     | V/A   | Volume / Amplitude de la voix 2 |       |       |       |
| Registre 10 : | -     | -     | -     | V/A   | Volume / Amplitude de la voix 3 |       |       |       |

Mettre le bit 4 (V/A) à 0 pour régler le volume sonore de la voix correspondante.

Lorsque le bit 4 (V/A) est mis à 1, le volume variera proportionnellement à la forme de l'enveloppe définie par le registre 13.

## 8.7 Registres de contrôle la forme et de la période de l'enveloppe (Registres 11~13)

|               | bit 7                                                                           | bit 6 | bit 5 | bit 4 | bit 3                | bit 2 | bit 1 | bit 0 |
|---------------|---------------------------------------------------------------------------------|-------|-------|-------|----------------------|-------|-------|-------|
| Registre 11 : | 8 bits de poids faible de la valeur qui détermine la période de l'enveloppe (T) |       |       |       |                      |       |       |       |
| Registre 12 : | 8 bits de poids fort de la valeur qui détermine période de l'enveloppe (T)      |       |       |       |                      |       |       |       |
| Registre 13 : | -                                                                               | -     | -     | -     | Forme de l'enveloppe |       |       |       |

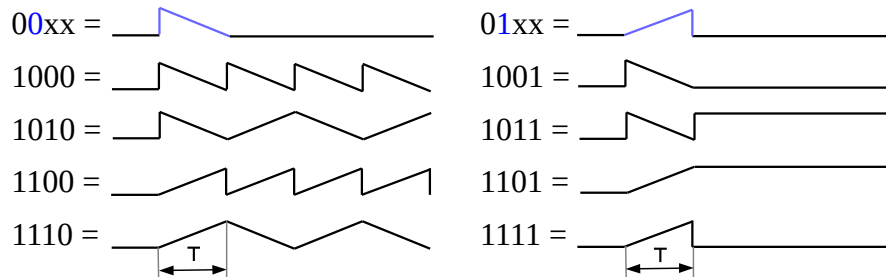
La valeur des registres 11 et 12 (de 1 à 65535) déterminent la période de l'enveloppe (T). La valeur à mettre se calcule à l'aide de la formule suivante. La période peut varier de 0,11 Hz à 6991 Hz.

$$\text{Valeur} = \frac{F_i}{256 \times T}$$

$F_i$  = Fréquence interne du PSG (1789772,5 Hz)

$T$  = Période de l'enveloppe (en  $\mu\text{s}$ )

Le registre 13 définit la forme de l'enveloppe. Voici les formes possibles :



Pour les détails :

- Le bit 0 (Hold) spécifie si la période doit être répétitif ou non.
- Le bit 1 (Alternate) spécifie si la forme de l'enveloppe doit être inversée ou pas à chaque répétition.
- Le bit 2 (Attack) spécifie l'inversion ou pas de la forme de l'enveloppe.
- Le bit 3 (Continue) spécifie que la forme de l'enveloppe doit rester au même niveau en fin de période.

## 8.8 Registres des ports parallèles d'entrées / sorties du PSG (Registres 14 et 15)

Les registres 14 et 15 servent à accéder aux ports parallèles du PSG. Sur MSX, le port A est connecté en tant qu'entrée et le B en sortie pour lire les ports généraux 1 ou 2 (aka : Ports Joystick) ainsi que deux autres signaux pour connaître le type de clavier japonais utilisé et pour savoir si une lecture est en cours sur cassette. Le registre 15 sert à contrôler les ports généraux, la LED de la touche « code » ou « Kana ».

|               | bit 7                                                                            | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---------------|----------------------------------------------------------------------------------|-------|-------|-------|-------|-------|-------|-------|
| Registre 14 : | Port parallèle A d'E/S du PSG (à régler toujours en entrée avec le bit 6 du R#7) |       |       |       |       |       |       |       |
| Registre 15 : | Port parallèle B d'E/S du PSG (à régler toujours en sortie avec le bit 7 du R#7) |       |       |       |       |       |       |       |

Détails:

- Registres 14 :

Ce registre permet de lire l'état des broches de 1 à 6 d'un port général du MSX (pour manettes de jeu, souris, tablette graphique, etc), l'état de la touche JIS/Kana des claviers japonais et l'état du signal de lecture sur cassette.

- Bit 0 = État de la broche 1 du port général sélectionné (Haut si joystick)
- Bit 1 = État de la broche 2 du port général sélectionné (Bas si joystick)
- Bit 2 = État de la broche 3 du port général sélectionné (Gauche si joystick)
- Bit 3 = État de la broche 4 du port général sélectionné (Droite si joystick)
- Bit 4 = État de la broche 6 du port général sélectionné (Bouton 1 si joystick)
- Bit 5 = État de la broche 7 du port général sélectionné (Bouton 2 si joystick)
- Bit 6 = 1 pour JIS, 0 pour Kana (valable seulement pour les MSX japonais )
- Bit 7 = CASRD (Signal de lecture sur cassette)

- Registre 15 :

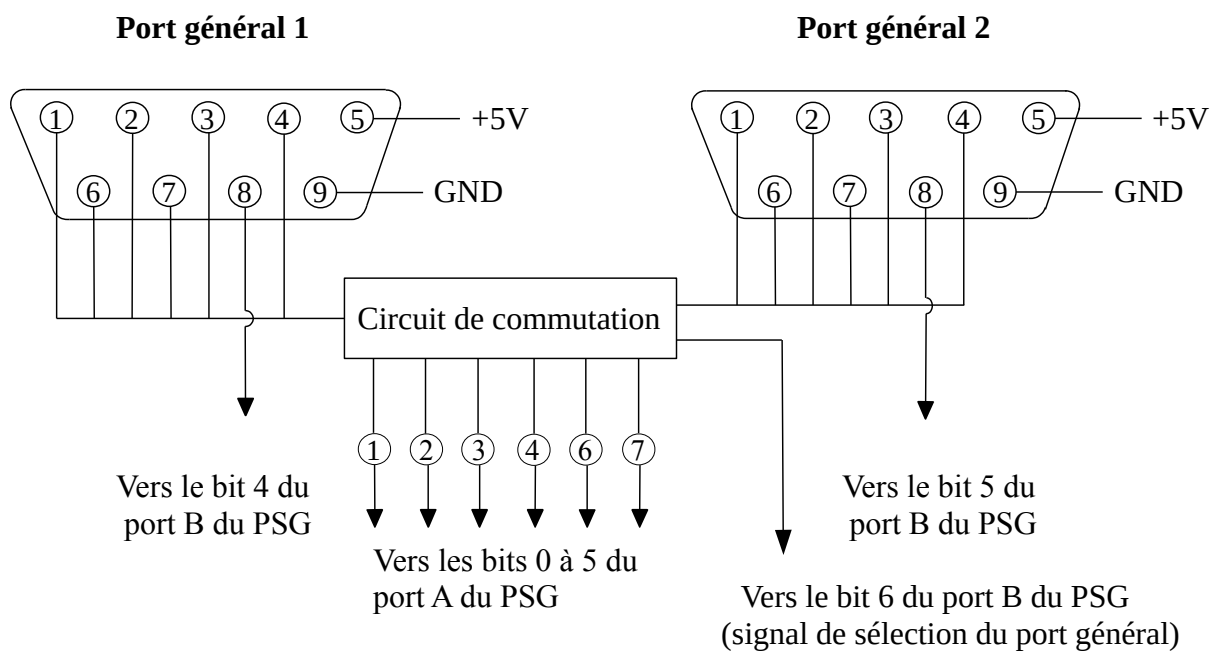
Ce registre permet de contrôler les broches des ports généraux à lire via le registre 14 ainsi que l'état de la LED « code » ou « Kana » selon le type de clavier.

- Bit 0 = Contrôle de la broche 6 du port général 1\*
- Bit 1 = Contrôle de la broche 7 du port général 1\*
- Bit 2 = Contrôle de la broche 6 du port général 2\*
- Bit 3 = Contrôle de la broche 7 du port général 2\*
- Bit 4 = Contrôle de la broche 8 du port général 1 (0 pour mode joystick standard)
- Bit 5 = Contrôle de la broche 8 du port général 2 (0 pour mode joystick standard)
- Bit 6 = Sélection du port général lisible via le registre 14 (1 pour port 2)
- Bit 7 = Contrôle de la LED de la touche « code » ou « Kana ». (1 pour éteindre)

\* Mettre à 1 si le port général est utilisé comme entrée (lecture).



Les broches des ports généraux sont connectées de la façon suivante.



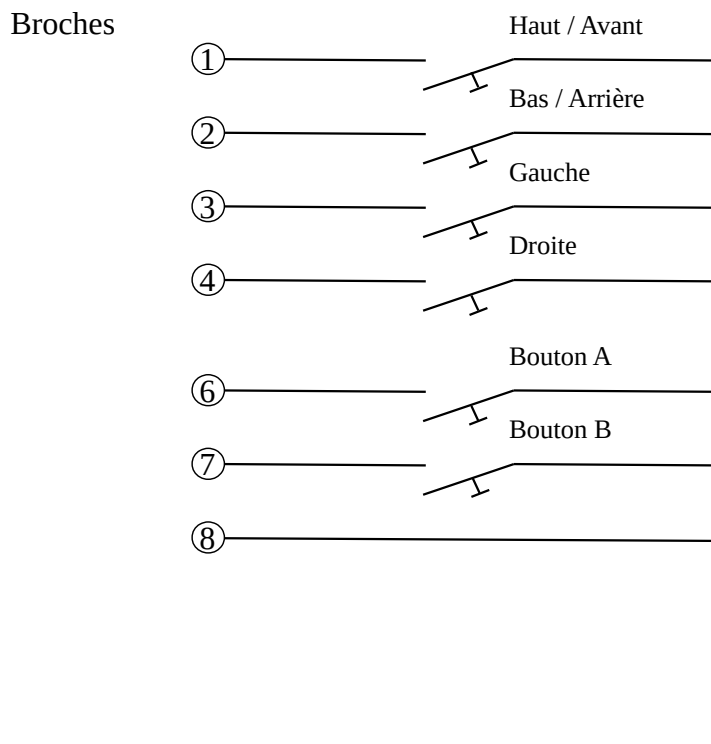
Exemples de fonctionnement :

Les exemples suivants ont pour but de montrer le fonctionnement des registres 14 et 15 du PSG. Il est préférable de se servir des routines du Bios autant que possible pour gérer les manettes de jeu, une souris, une tablette graphique ou les molettes ASCII.

## Manettes de jeu

Pour lire une manette de jeu, il faut d'abord lire le registre 15 pour connaître l'état de la LED du clavier afin d'écrire la même valeur au bit 7, mettre le bit 6 à 0 ou 1 selon la manette à lire, les bit 5 et 4 peuvent être tout les deux à 0 et mettre les autres bits (0 à 3) à 1.

Le schéma d'une manette standard (Type B) est le suivant.

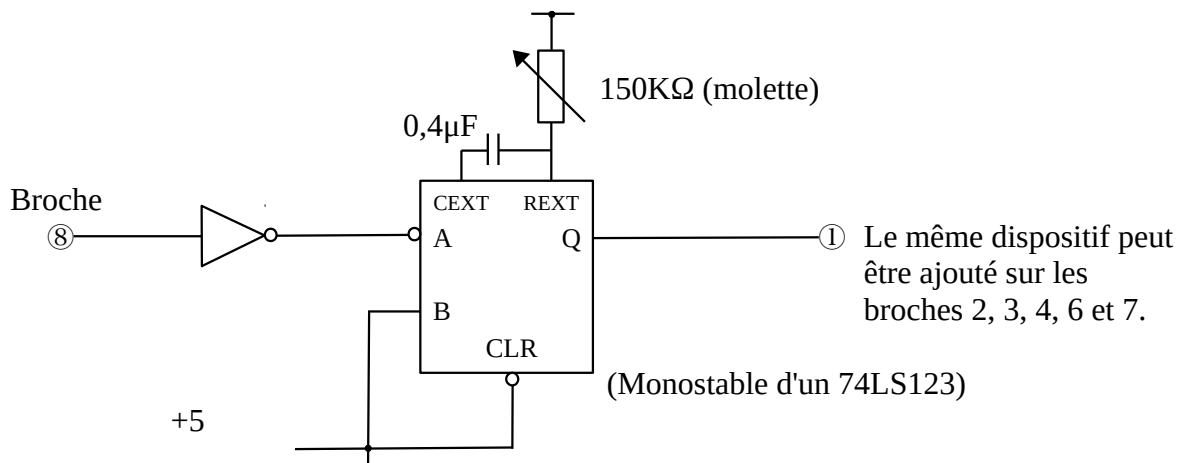


Lorsque l'on presse un bouton, la broche correspondante reçoit le signal zéro de la broche 8. Ainsi, le bit correspondant du registre 14 sera ainsi mis à 0. Notez que, dans ce cas, la broche 8 peut être remplacée par la masse. Il est possible d'ajouter des boutons à la manette en reliant la broche 8 à, par exemple, une puce 74LS157 pour commuter entre la manette standard et les boutons supplémentaires (non standard).

Les routines du Bios pour gérer les manettes de jeu sont **GTSTCK** (000D5h) et **GTTRIG** (000D8h) en Main-ROM. Voir aussi la variable système **TRGFLG** (0F3E8h).

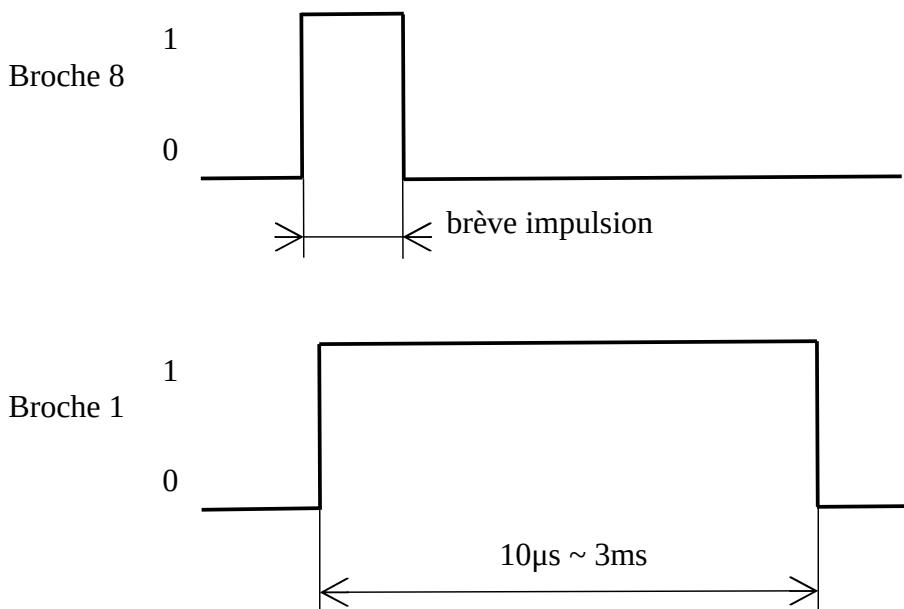
## Molettes de jeu ASCII

Les molettes ASCII n'ont été utilisées que par quelques jeux dont Break Out édité par ASCII au Japon. Le support de ses molettes a même été retiré du Bios des MSX turbo R. Le schéma est le suivant.



Le but de ce circuit est d'envoyer un bref signal sur la broche 8 pour obtenir un signal sur la broche 1 dont la longueur varie en fonction de la résistance variable contrôlée par la molette. La valeur de la position est ensuite à déduire en fonction de la longueur du signal. Il est possible de connecter jusqu'à 6 molettes par port général donc dans ce cas un bref signal sur la broche 8 déclenchera un signal de longueur différente sur chacune des broches 1, 2, 3, 4, 6 et 7.

Nature du signal :



La longueur du signal permet de déterminer la position.

La routine du Bios pour gérer les molettes ASCII est **GTPDL** (000DEh en Main-ROM). Cette routine n'étant pas implémentée dans les MSX Turbo R, voici une routine de remplacement :

```
; Entrée : A = Numéro de molette (0~12)
; Sortie : A = Position de la molette (0~255)

GTPDL:
    inc     a
    and     a
    rra                                ; Carry=1 pour Port 2
    push    af
    ld      b,a
    xor     a                                ; A=0 (et Carry=0)
    scf                                ; Carry=1
PDL_LOOP:
    rla
    djnz    PDL_LOOP
    ld      b,a                                ; B=bit correspondant à la broche à 1
    pop     af
    ld      c,10h                            ; Broche 8 à 1 sur port général 1
    ld      de,03afh                        ; D=11b (broche 6 & 7 port 1)
   ; E=10101111b (broche 8 port 1 à 0)
    jr      nc,PDL_PORT1
    ld      c,20h                            ; Broche 8 à 1 sur port général 2
    ld      de,4c9fh                        ; D=1001100b (broche 6 & 7 port 2)
   ; E=10011111 (broche 8 port 2 à 0)
PORT1:
    ld      a,0fh
    di
    out     (0a0h),a                        ; Registre 15
    in      a,(0a2h)
    and     e                                ; Garde le bit de sélection du port général
    or      d
    or      c
    out     (0a1h),a                        ; Envoie de l'impulsion (broche 8 = 1)
    xor     c
    out     (0a1h),a                        ; Fin de l'impulsion (broche 8 = 0)

    ld      a,0eh
    out     (0a0h),a                        ; Registre 14
    ld      c,0
WAIT:
    in      a,(0a2h)
    and     b
    jr      z,DONE                          ; Saut si broche de la molette est à 0
    inc     c                                ; Incrémente C
    jp      nz,WAIT                        ; Saut tant que C n'est pas à 0
    dec     c
DONE:
    ei
    ld      a,c                                ; A = 0~255
    ret
```

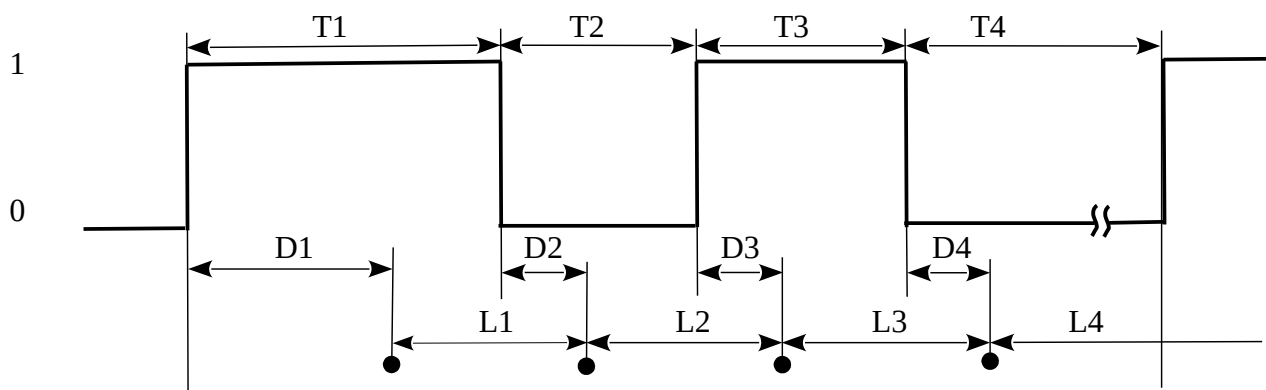
Note : Cette routine ne fonctionne qu'en mode Z80 (cadencé à 3,579545 MHz).

## La souris

Les boutons de la souris fonctionnent comme ceux d'une manette standard. Pour indiquer le déplacement, la souris dispose de 4 registres de 4 bits. La lecture des registres est commandée par la broche 8. Une première mise à 1 de cette broche commande la lecture du premier registre. La remise à 0 commande la lecture du second registre. Ensuite, la remise à 1 commande la lecture du troisième registre. Puis, la remise à 0 commande la lecture du quatrième registre. Et ainsi de suite. Le tout, en respectant les timings indiqués plus bas.

| Registre | Contenu (Coordonnée relative)        | Broches correspondantes |    |    |    |
|----------|--------------------------------------|-------------------------|----|----|----|
|          |                                      | 4                       | 3  | 2  | 1  |
| 1        | 4 bits de poids fort de l'abscisse   | X4                      | X6 | X5 | X4 |
| 2        | 4 bits de poids faible de l'abscisse | X3                      | X2 | X1 | X0 |
| 3        | 4 bits de poids fort de l'ordonnée   | Y4                      | Y6 | Y5 | Y4 |
| 4        | 4 bits de poids faible de l'ordonnée | Y3                      | Y2 | Y1 | Y0 |

Timings de lecture des coordonnées en fonction de l'état la broche 8 :



T1 = Intervalle entre 100 et 150  $\mu$ s

T2 = Intervalle entre 50 et 150  $\mu$ s (si T2 excède les 150  $\mu$ s, la souris initialisera ses registres.)

T4 = Intervalle de 300  $\mu$ s minimum durant lequel la souris scrute les coordonnées.

D1 = 100  $\mu$ s. Temps d'attente avant de pouvoir lire le premier registre.

D2~D4 = 50  $\mu$ s. Temps d'attente avant de pouvoir lire le registre suivant.

L1~L4 = Intervalle durant lequel il est possible de lire le registre correspondant.

Chaque lecture du registre 14 donnera les informations de la façon suivantes.

Bit 0 à 3 = 4 bits de poids fort/faible de l'abscisse/ordonnée relative.

Bit 4 = État du bouton gauche

Bit 5 = État du bouton droit

Bit 6 = 1 pour JIS, 0 pour Kana (valable seulement pour les MSX japonais)

Bit 7 = CASRD (Signal de lecture sur cassette)

La routine du Bios pour gérer la souris est **GTPAD** (000DBh en Main-ROM) ou **NEWPAD** (001ADh en Sub-ROM) et **GTRIG** (000D8h) en Main-ROM pour les boutons mais voici une routine de remplacement pour le principe (page suivante) :

Note : Les souris MSX (compatibles PC-8801 et FM-Towns) possèdent un mode « Joystick » que l'on peut activer en pressant le bouton gauche lors de la mise sous tension de la souris. Ce mode simule une manette de jeu. Il reste activé jusqu'à l'extinction de la souris.

```
; Routine pour lire la souris par accès direct au PSG
```

```
; Entrée :
; Mettre D à 10010011b et E à 00010000b (DE=9310h) si la souris est dans le port 1
; Mettre D à 11101100b et E à 00100000b (DE=EC20h) si la souris est dans le port 2
```

```
; Sortie :
; H = abscisse relative (-128~+127), L = ordonnée relative (-128~+127)
; H et L = 255 si la souris n'est pas trouvée
```

```
GTMOUS:
```

```
    ld    b,30          ; Long délai d'attente pour la première lecture
    call  GTOFS2        ; Lecture des 4 bits de poids fort de l'abscisse
relative
    and   0fh
    rlca
    rlca
    rlca
    rlca
    ld    c,a          ; Garde les bits de poids fort dans C
    call  GTOFST        ; Lecture des 4 bits de poids faible de l'abscisse
relative
    and   0fh
    or    c             ; Place les bits de poids fort
    ld    h,a          ; Recombine l'abscisse relative dans H

    call  GTOFST        ; Lecture des 4 bits de poids fort de l'ordonnée
relative
    and   0fh
    rlca
    rlca
    rlca
    rlca
    ld    c,a          ; Garde les bits de poids fort dans C
    call  GTOFST        ; Lecture des 4 bits de poids faible de l'ordonnée
relative
    and   0fh
    or    c             ; Place les bits de poids fort
    ld    l,a          ; Recombine l'ordonnée relative dans L
    ret
```

```
GTOFST:
```

```
    ld    b,10          ; Court délai d'attente
```

```
GTOFS2:
```

```
    ld    a,15
    out   (0a0h),a      ; Registre 15
    in    a,(0a1h)
    and   80h           ; Garde le bit de l'état de la LED Code/Kana
    or    d
    out   (0a1h),a
    xor   e             ; Inverse le bit de la broche 8
```

```

ld      d,a
WAIT:   djnz  WAIT      ; Délai d'attente pour lire la valeur au bon timing

ld      a,14
out     (0a0h),a      ; Registre 14
in      a,(0a2h)
ret

```

Note : Cette routine ne fonctionne qu'en mode Z80 (cadencé à 3,579545 MHz).

### Track-Ball

Le Track-Ball fonctionnent à peu près de la même manière que la souris. Les boutons fonctionnent comme ceux d'une manette standard. Quant à la broche 8, une première mise à 1 commande l'envoi l'abscisse relatif sur 4 bits. Ensuite, la mise à 0 de la broche 8 commandera l'envoi de l'ordonnée relatif sur 4 bits.

Chaque lecture du registre 14 donnera les informations de la façon suivantes.

Bit 0 à 3 = Abscisse/ordonnée relative sur 4 bits. La valeur lu doit être interprétée comme suit.

| Valeur lue | Coordonnée correspondante |
|------------|---------------------------|
| 0000b      | -8                        |
| 1          | 1                         |
| 0111b      | -1                        |
| 1000b      | 0                         |
| 1          | 1                         |
| 1111b      | 7                         |

Bit 4 = État du bouton 1

Bit 5 = État du bouton 2

Bit 6 = 1 pour JIS, 0 pour Kana (valable seulement pour les MSX japonais )

Bit 7 = CASRD (Signal de lecture sur cassette)

Note : Certains modèles ont trois boutons. Ce bouton à le même effet que les boutons 1 et 2 pressés en même temps.

La routine du Bios pour gérer le Track-Ball est **GTPAD** (000DBh en Main-ROM) ou **NEWPAD** (001ADh en Sub-ROM) et **GTTRIG** (000D8h) en Main-ROM

## 9 Le MSX-Music

Le MSX-Music est une option du standard pour jouer les musiques au moyen d'un générateur de son programmable à synthèse FM conçu et fabriqué par Yamaha dont la référence est YM2413 . Cette puce est appelée « OPLL » car c'est une version très simplifiée de l'OPL2 (YM3812). La cartouche Sound Blaster 1.0 utilise l'OPL2. Un matériel ou un logiciel compatible MSX-Music doit comporter le logo suivant.



### 9.1 Caractéristiques

L'OPLL du MSX-Music peut produire jusqu'à 9 voix FM ou 6 voix + boîte à rythme. Les sons sont produits par deux opérateurs. Les 24 registres de l'OPLL sont accessibles par écritures aux ports d'entrée/sortie 07Ch et 07Dh. L'MSX-Music comprend aussi une ROM de 16Ko qui contient un Bios et des instructions Basic étendues. L'option MSX-Music peut être ajoutée au MSX en interne ou en externe. Les registres d'une version externe sont accessibles aussi par écritures aux adresses 05FF4h et 7FF5h dans le Slot de la cartouche. Le but de ces adresses est de contrôler un OPLL externe indépendamment d'un autre en interne.

La sélection du registre à modifier se fait via le port 07Ch (ou l'adresse 05FF4h) et l'écriture des données via le port 07Dh (ou l'adresse 05FF5h). Il faut respecter un délai de 3,36 µs entre chaque sélection de registre et un délai de 23,52 µs entre chaque écriture de donnée.

La norme du MSX-Music stipule que les ports d'accès à un OPLL externe doivent être désactivés par défaut. Pour les activer, il faut écrire une valeur avec le bit 0 à 1 à l'adresse 7FF6h (dans le Slot correspondant). Attention, assurez-vous qu'il s'agit bien d'un MSX-Music externe avant d'écrire cette valeur. En effet, car si l'on écrit cette valeur entre 7FF0h et 7FFFh sur un MSX2+ Panasonic FS-A1WX ou FS-A1WSX, cela aura pour effet de désactiver la ROM. La méthode de détection se trouve à la page suivante.

Note : Selon la norme, l'accès aux registres par adressage mémoire devrait être effectif sur tous les MSX-Music externes mais, dans les faits, la plupart des MSX-Music qui ont été vendus hors du Japon sont des versions non officielles (FM-PAQ, FM-PAK, etc) qui n'ont pas cette fonctionnalité.



## 9.2 Méthode de détection du Slot du MSX-Music

Pour trouver un MSX-Music, vous devez d'abord effectuez une recherche du texte « **APRLOPLL** » dans tous les Slots à l'adresse 4018h (4018h~401Fh) pour déterminer si un MSX-Music est intégré dans l'ordinateur MSX utilisé ou pas.

Pour trouver un MSX-Music en cartouche, vous devez recherche du texte « **OPLL** » dans tous les Slots à l'adresse 401Ch (401Ch~401Fh). Les quatre caractères précédents seront « **PAC2** », il s'agit de l'FM-PAC ou autre que « **APRL** ».

Lorsque le MSX-Music est intégré au MSX, seules les ports E/S 7Ch et 7Dh sont utilisables pour accéder aux registres de l'OPPL. Ceux-ci sont toujours activés par défaut.

Si il s'agit d'un MSX-Music en cartouche, il est possible d'accéder aux registres de l'OPLL via les ports E/S 7Ch et 7Dh, ou les adresses 7FF4h et 7FF5h. Les ports E/S sont désactivés par défaut. Vous pouvez les activer en écrivant une valeur avec le bit 0 à 1 à l'adresse 7FF6h dans le Slot correspondant. Ne les activez pas si un MSX-Music interne est présent sinon *le volume sera deux fois plus fort puisque les deux sources sonores FM jouées en même temps s'additionnent.*

Attention, assurez-vous qu'il s'agit bien d'un MSX-Music externe avant d'écrire cette valeur. En effet, car si l'on écrit cette valeur avec le bit 0 à 1 entre 7FF0h et 7FFFh sur un MSX2+ Panasonic FS-A1WX ou FS-A1WSX, cela aura pour effet de désactiver la ROM.

### 9.3 Registres de l'OPLL

Pour accéder directement aux registres de l'OPLL faites très attention de respecter les temps d'attente entre chaque accès. La tâche peut être assez ardue si vous tenez compte des modes turbo de certains MSX car ils sont tous différents. Autrement, utilisez de préférence le [FM Bios](#).

Registres de contrôle sonore :

Ces huit registres permettent de créer son propre instrument.

|              | bit 7       | bit 6 | bit 5                 | bit 4 | bit 3    | bit 2    | bit 1 | bit 0 |               |
|--------------|-------------|-------|-----------------------|-------|----------|----------|-------|-------|---------------|
| Registre 0 : | AM          | VIB   | EGTYP                 | KSR   | MULTIPLE |          |       |       | (opérateur 0) |
| Registre 1 : | AM          | VIB   | EGTYP                 | KSR   | MULTIPLE |          |       |       | (opérateur 1) |
| Registre 2 : | KSL (op. 0) |       | Total Level Modulator |       |          |          |       |       |               |
| Registre 3 : | KSL (op. 1) |       | -                     | DC    | DM       | Feedback |       |       |               |
| Registre 4 : | AR          |       |                       |       | DR       |          |       |       | (opérateur 0) |
| Registre 5 : | AR          |       |                       |       | DR       |          |       |       | (opérateur 1) |
| Registre 6 : | SL          |       |                       |       | RR       |          |       |       | (opérateur 0) |
| Registre 7 : | SL          |       |                       |       | RR       |          |       |       | (opérateur 1) |

**MULTIPLE** = Contrôle de l'échantillon de l'onde ; Relation harmonique.

**KSR** (Key Scale Rate) = Taux de .

**EGTYP** = 0 pour une tonalité de percussion, 1 pour une tonalité soutenue.

**VIB** (Vibrato) = Active / Désactive le vibrato.

**AM** (Amplitude Modulation) = Active / Désactive la modulation d'amplitude.

**Total Level Modulator** = Niveau total de l'onde modulée. Contrôle de l'index de modulation.

**KSL** (Key Scale Level) = Niveau de « Key scale ».

**FR** (Feedback Rate) = Constante de retour FM.

**DM** (Distortion Module) = Distorsion de l'onde modulée de l'enveloppe. (Rectification par ondes planes.)

**DC** (Distortion Carrier) = Distorsion de l'onde porteuse de l'enveloppe. (Rectification par ondes planes.)

**DR** (Decay Rate) = Contrôle de la fréquence de changement d'enveloppe de décomposition.

**AR** (Attack Rate) = Contrôle du changement du taux d'enveloppe d'attaque.

**RR** (Release Rate) = Contrôle du niveau de changement d'enveloppe de sortie.

**SL** (Sustain Level) = Indication de décroissance ; niveau de maintien.

L'opérateur 0 est un opérateur modulateur et l'opérateur 1 est un opérateur de type porteur.

### Registre de contrôle de la boîte à rythme:

|               | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|
| Registre 14 : | -     |       | R     | BD    | SD    | TOM   | T-CT  | HH    |

HH = Active / Désactive la charleston (High-Hat) de la boîte à rythme.

T-CT = Active / Désactive la cymbale du haut (Top Cymbale) de la boîte à rythme.

TOM = Active / Désactive le Tom de la boîte à rythme.

SD = Active / Désactive la caisse claire de la boîte à rythme.

BD = Active / Désactive la grosse caisse de la boîte à rythme.

R = Active / Désactive la boîte à rythme. Désactive les voix 6 à 8 pour la boîte à rythme.

### Registre de test de l'OPLL :

|               | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|
| Registre 15 : | TEST  |       |       |       |       |       |       |       |

TEST = Donnée de test de l'OPLL.

### Registre de réglages :

|                    | bit 7               | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0          |              |
|--------------------|---------------------|-------|-------|-------|-------|-------|-------|----------------|--------------|
| Registre 16 à 24 : | Bits 0 ~ 7 de F-NUM |       |       |       |       |       |       |                | (voix 0 à 8) |
| Registre 32 à 40 : | -                   |       | SUS   | KEY   | BLOCK |       |       | Bit 8 de F-NUM | (voix 0 à 8) |
| Registre 48 à 56 : | INST                |       |       |       | VOL   |       |       |                | (voix 0 à 8) |

F-NUM = Indication de la fréquence.

BLOCK = Réglage de l'octave.

KEY = Touche maintenue ou pas.

SUS = Active / Désactive le Sustain.

VOL = Réglage du volume sonore.

INST = Sélection de l'instrument.

|               |                |                   |                            |
|---------------|----------------|-------------------|----------------------------|
| 0 = Original* | 4 = Flute      | 8 = Orgue         | 12 = Vibraphone            |
| 1 = Violon    | 5 = Clarinette | 9 = Corne         | 13 = Basse de synthétiseur |
| 2 = Guitare   | 6 = Hautbois   | 10 = Synthétiseur | 14 = Basse acoustique      |
| 3 = Piano     | 7 = Trompette  | 11 = Clavecin     | 15 = Guitare électrique    |

(\*) Instrument crée par l'utilisateur via les registres 0 à 7.

## 9.4 Le FM Bios

Le système n'a pas de variable pour le MSX-Music. Vous devez donc trouver le Slot vous-même pour appeler une routine du Bios d'un MSX-Music. Le standard conseille d'utiliser le Bios plutôt que les ports E/S bien que dans certains cas avancés, vous serez certainement obligé de passer par les ports E/S.

Le Bios du MSX-Music est appelé « FM Bios ». Il est séparé en deux zones, une qui contient des données pour créer les sons et une pour la table des sauts comme tous les autres Bios.

|             |                                       |
|-------------|---------------------------------------|
|             |                                       |
| 4100h~4BFFh | FM Bios                               |
| 4C00h~4DFFh | Données pour les sons des instruments |
|             |                                       |

### Table des sauts du FM Bios

#### 04110h (FM-ROM)      **WRTOPL**    Write to OPLl register

Rôle :            Ecriture d'un paramètre dans un registre de l'OPLL.  
Entrée :        A = Numéro du registre (0~7, 14~24, 32~40 et 48~56).  
                    E = Donnée à écrire.  
Sortie :        Rien.  
Modifie :      Rien.  
Note:          La routine ne coupe pas les interruptions. Veillez à que les interruptions soient coupées pour appeler cette routine.

#### 04113h (FM-ROM)      **INIOPL**    INItialises the OPLl

Rôle :            Prépare l'environnement du FM Bios et initialise tous les registres de l'OPLL.  
Entrée :        HL = Pointeur (Nombre pair) vers la zone de travail à utiliser pour le FM Bios.  
                    Cette zone doit être de la RAM. Sinon plusieurs routines ne fonctionneront pas.  
Sortie :        Rien.  
Modifie :      AF, BC, DE, HL, IX et IY.  
Notes:        - Il faut appeler cette routine avant l'utilisation de toutes autres routines du MSX-Music.  
                    - L'adresse de la zone de travail est stockée à SLTWRK (0FD09h).  
                    - La routine coupe et rétablit les interruptions avant de rendre la main.

### 04116h (FM-ROM)      **MSTART**    Music START

- Rôle :            Jouer la musique. (Le format des données est indiqué au paragraphe suivant)
- Entrée :          HL = Adresse des données de la musique.  
                    A = Indicateur pour jouer ou non la musique en boucle.  
                            0 pour jouer la musique en boucle,  
                            1~254 pour jouer la musique ce nombre de fois,  
                            255 ne doit pas être spécifié.  
                    (HL) = 0EH pour jouer en mode 6 voix FM + boîte à rythme.  
                            12H pour jouer en mode 9 voix FM.
- Sortie :          Rien.
- Modifie :        AF, BC, DE, HL, IX et IY.
- Notes:           - Avant d'appeler cette routine, il faut paramétrer la zone de travail du MSX-Music en fonction de l'entête des données de la musique pointé par HL. Assurez-vous aussi que OPLDRV soit bien appelé par le Hook H.TIMI (0FD9FH).  
                    - La routine coupe et rétablie les interruptions avant de rendre la main.

### 04119h (FM-ROM)      **MSTOP**      Music STOP

- Rôle :            Stopper la musique par l'OPLL et initialise la zone de travail du MSX-Music.
- Entrée :          Rien.
- Sortie :          Rien.
- Modifie :        AF, BC, DE, HL, IX et IY.
- Note:            - La routine coupe et rétablie les interruptions avant de rendre la main.

### 0411Ch (FM-ROM)      **RDATA**      Read DATA

- Rôle :            Récupérer les données en ROM de l'instrument spécifié.
- Entrée :          HL = Adresse la zone de travail (en RAM) qui recevra les données lues.  
                    A = Numéro du son d'un instrument. (0~63)
- Sortie :          Rien.
- Modifie :        F.
- Note :            Les données sonores de instrument sont copiées dans la zone de travail.

## 0411Fh (FM-ROM)      **OPLDRV**   OPLl DRiVer

Rôle :            Installe le pilote pour jouer la musique FM.  
Entrée :        Rien.  
Sortie :        Rien.  
Modifie :      Rien.  
Note:           Cette routine permet de jouer les musiques. Elle doit être appelée à chaque interruption via le Hook H.TIMI (0FD9Fh). Il faut donc modifier ce Hook en appelant une fois la routine INIOPL (04113h) avant tout appel aux autres routines du FM Bios.

## 04122h (FM-ROM)      **TSTBGM**   TeST BGM

Rôle :           Vérifie si la musique jouée avec MSTART est finie ou pas.  
Entrée :        Rien.  
Sortie :        A= 0 si la musique est finie, autrement la musique est en cours de lecture.  
Modifie :      AF.  
Note:           - Les interruptions ne doivent pas être coupées.

9.5 *Format des données d’une musique FM*

Il y a un format pour chaque mode.

**L’entête**

Le format de l’entête dans le mode 6 voix FM + boîte à rythme est le suivant.

|                          |           |     |                                                                                      |
|--------------------------|-----------|-----|--------------------------------------------------------------------------------------|
| Adresse pointée par HL : | 0Eh       | 00h | 0Eh pour des données en mode 6 voix FM + boîte à rythme                              |
| HL+02h :                 | Adresse 1 |     | Indique l’emplacement des données pour la voix 1 FM.                                 |
| HL+04h :                 | Adresse 2 |     | Indique l’emplacement des données pour la voix 2 FM.                                 |
| HL+06h :                 | Adresse 3 |     | Indique l’emplacement des données pour la voix 3 FM.                                 |
| HL+08h :                 | Adresse 4 |     | Indique l’emplacement des données pour la voix 4 FM.                                 |
| HL+0Ah :                 | Adresse 5 |     | Indique l’emplacement des données pour la voix 5 FM.                                 |
| HL+0Ch :                 | Adresse 6 |     | Indique l’emplacement des données pour la voix 6 FM.                                 |
| HL+0Eh :                 | .         |     | Emplacement des données pour la boîte à rythme. (voir plus bas pour plus de détail.) |
|                          | .         |     |                                                                                      |
|                          | .         |     |                                                                                      |

Le format de l’entête dans le mode 9 voix FM est le suivant.

|                          |           |     |                                                                                 |
|--------------------------|-----------|-----|---------------------------------------------------------------------------------|
| Adresse pointée par HL : | 12h       | 00h | 12h pour des données en mode 6 voix FM + boîte à rythme                         |
| HL+02h :                 | Adresse 1 |     | Indique l’emplacement des données pour la voix 2 FM.                            |
| HL+04h :                 | Adresse 2 |     | Indique l’emplacement des données pour la voix 3 FM.                            |
| HL+06h :                 | Adresse 3 |     | Indique l’emplacement des données pour la voix 4 FM.                            |
| HL+08h :                 | Adresse 4 |     | Indique l’emplacement des données pour la voix 5 FM.                            |
| HL+0Ah :                 | Adresse 5 |     | Indique l’emplacement des données pour la voix 6 FM.                            |
| HL+0Ch :                 | Adresse 6 |     | Indique l’emplacement des données pour la voix 7 FM.                            |
| HL+0Eh :                 | Adresse 7 |     | Indique l’emplacement des données pour la voix 8 FM.                            |
| HL+10h :                 | Adresse 8 |     | Indique l’emplacement des données pour la voix 9 FM.                            |
| HL+12h :                 | .         |     | Emplacement des données pour la voix 1 FM. (voir plus bas pour plus de détail.) |
|                          | .         |     |                                                                                 |
|                          | .         |     |                                                                                 |

Les données des voix sont placées les unes à la suite des autres à partir de l’adresse HL+0Eh en mode 6 voix FM + boîte à rythme ou, HL+12h en mode 9 voix FM. L’adresse des données de la première voix peut s’obtenir en additionnant HL avec la valeur du premier octet.

## Format des données pour la mélodie

### Données d'une voix FM :

Chacun des octets des données d'une voix FM est codé de la façon suivante.

**00h~5Fh** 00h correspond à un silence. Les autres correspondent aux notes de musiques suivantes.

| Octave<br>Note | 1   | 2   | 3   | 4    | 5   | 6   | 7   | 8   |
|----------------|-----|-----|-----|------|-----|-----|-----|-----|
| Do (C)         | 01h | 0Dh | 19h | 25h  | 31h | 3Dh | 49h | 55h |
| Do# (C#)       | 02h | 0Eh | 1Ah | 26h  | 32h | 3Eh | 4Ah | 56h |
| Ré (D)         | 03h | 0Fh | 1Bh | 27h  | 33h | 3Fh | 4Bh | 57h |
| Ré# (D#)       | 04h | 10h | 1Ch | 28h  | 34h | 40h | 4Ch | 58h |
| Mi (E)         | 05h | 11h | 1Dh | 29h  | 35h | 41h | 4Dh | 59h |
| Fa (F)         | 06h | 12h | 1Eh | 2Ah  | 36h | 42h | 4Eh | 5Ah |
| Fa# (F#)       | 07h | 13h | 1Fh | 2Bh  | 37h | 43h | 4Fh | 5Bh |
| Sol (G)        | 08h | 14h | 20h | 2Ch  | 38h | 44h | 50h | 5Ch |
| Sol# (G#)      | 09h | 15h | 21h | 2Dh  | 39h | 45h | 51h | 5Dh |
| La (A)         | 0Ah | 16h | 22h | 2Eh* | 3Ah | 46h | 52h | 5Eh |
| La# (A#)       | 0Bh | 17h | 23h | 2Fh  | 3Bh | 47h | 53h | 5Fh |
| Si (B)         | 0Ch | 18h | 24h | 30h  | 3Ch | 48h | 54h | -   |

(\*) 2Eh est la note produite par les diapasons.

L'octet qui suit une note indique la longueur de la note. Si cet octet est FFh, c'est que le prochain octet sera également un octet pour indiquer la longueur de la note. Et ainsi de suite jusqu'à ce que l'octet ne soient plus FFh. La longueur de la note s'obtient en multipliant la valeur de l'octet par 1/60 secondes.

Par exemple, si vous spécifiez : 19h FFh 10h, cela jouera la note Do de l'octave 3 pendant  $(255 + 16) * 1/60$  secondes.

**60h~6Fh** Volume sonore. 6FH est le volume maximum.

**70h~7Fh** Niveau du timbre.

**80h/81h** Désactive/Active le sustain.



- 82h** Sélection d'un instrument en ROM. L'octet suivant doit être une valeur de 0 à 63 pour indiquer l'instrument. (Voir le tableau des instruments plus bas.)
- 83h** Sélection de l'instrument que vous avez défini. Les deux octets suivants doivent être l'adresse des données de l'instrument.
- 84h** Désactive le Legato. (Les notes seront jouées avec un espace entre chacune.)
- 85h** Active le Legato.
- 86h** Réglage du Q. L'octet suivant doit être une valeur de 0 à 8.
- 87h~FEh** Inutilisé.
- FFh** Fin des données de la voix.

#### Données de la boîte à rythme :

L'octet de contrôle de la boîte à rythme a le format suivant.

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| V | 0 | 1 | B | S | T | C | H |
|---|---|---|---|---|---|---|---|

B est le bit pour sélectionner la grosse caisse

S permet de sélectionner la caisse claire

T permet de sélectionner le tom-tom

C permet de sélectionner la cymbale

H permet de sélectionner le charley

Si V est 1, l'octet suivant correspond à la longueur du son. Le format des données est le même pour le réglage de la longueur des notes.

Si V = 0, l'octet suivant (0 à 15) correspond au volume. Seuls les 4 bits inférieurs sont valides.

## Format des données pour les sons des instruments

|            | bit 7       | bit 6 | bit 5                 | bit 4 | bit 3    | bit 2    | bit 1 | bit 0 |               |
|------------|-------------|-------|-----------------------|-------|----------|----------|-------|-------|---------------|
| Pointeur   | AM          | VIB   | EGTYP                 | KSR   | MULTIPLE |          |       |       | (opérateur 0) |
| Pointeur+1 | AM          | VIB   | EGTYP                 | KSR   | MULTIPLE |          |       |       | (opérateur 1) |
| Pointeur+2 | KSL (op. 0) |       | Total Level Modulator |       |          |          |       |       |               |
| Pointeur+3 | KSL (op. 1) |       | -                     | DC    | DM       | Feedback |       |       |               |
| Pointeur+4 | AR          |       |                       |       | DR       |          |       |       | (opérateur 0) |
| Pointeur+5 | AR          |       |                       |       | DR       |          |       |       | (opérateur 1) |
| Pointeur+6 | SL          |       |                       |       | RR       |          |       |       | (opérateur 0) |
| Pointeur+7 | SL          |       |                       |       | RR       |          |       |       | (opérateur 1) |

**MULTIPLE** = Contrôle de l'échantillon de l'onde ; Relation harmonique.

**KSR** (Key Scale Rate) = Taux de .

**EGTYP** = 0 pour une tonalité de percussion, 1 pour une tonalité soutenue.

**VIB** (Vibrato) = Active / Désactive le vibrato.

**AM** (Amplitude Modulation) = Active / Désactive la modulation d'amplitude.

**Total Level Modulator** = Niveau total de l'onde modulée. Contrôle de l'index de modulation.

**KSL** (Key Scale Level) = Niveau de « Key scale ».

**FR** (Feedback Rate) = Constante de retour FM.

**DM** (Distortion Module) = Distorsion de l'onde modulée de l'enveloppe. (Rectification par ondes planes.)

**DC** (Distortion Carrier) = Distorsion de l'onde porteuse de l'enveloppe. (Rectification par ondes planes.)

**DR** (Decay Rate) = Contrôle de la fréquence de changement d'enveloppe de décomposition.

**AR** (Attack Rate) = Contrôle du changement du taux d'enveloppe d'attaque.

**RR** (Release Rate) = Contrôle du niveau de changement d'enveloppe de sortie.

**SL** (Sustain Level) = Indication de décroissance ; niveau de maintient.

L'opérateur 0 est un opérateur modulateur et l'opérateur 1 est un opérateur de type porteur.

## 10 Le MSX-Audio

Le MSX-Audio est une extension sonore optionnelle. Celle-ci est composée d'un générateur de sons FM optionnel sortie en 1986 qui comprend aussi des fonctions d'analyse et de synthèse vocale utilisant l'ADPCM/PCM et un convertisseur analogique-numérique et vice versa. Le MSX-Audio s'est relativement peu répandu sans doute à cause de son prix.

La puce principale du MSX-Audio est la YM8950 de Yamaha, appelée aussi OPL. Le fonctionnement est différent du MSX-Music ou des modules FM Yamaha.

La puce YM8950 est utilisée dans les cartouches FS-CA1 (Panasonic), HX-MU900 (Toshiba), NMS 1205 (Philips) et, bien plus récemment, elle est reproduite par le FPGA du GR8NET (Eugeny Brychkov).

Les cartouches NMS 1205 (Philips) et HX-MU900 (Toshiba) sont souvent considérés comme des MSX-Audio mais elles ne sont pas complètement compatibles MSX-Audio, car il leur manque le BIOS et les instructions BASIC étendues, et même la RAM de l'ADPCM/PCM pour la HX-MU900. De plus, le connecteur du clavier musical est différent sur la NMS 1205. Il existe cependant des kits à monter soit-même pour les rendre presque totalement compatible.

### 10.1 Caractéristiques

Le MSX-Audio a les caractéristiques suivantes.

- Générateur sonore LSI : Une ou deux puces Y8950 Yamaha. (La seconde utilisée en esclave.)
  - Générateurs sonores FM compatible avec la YM3526 pour produire un son réaliste.
  - Deux modes de fonctionnement : 9 voix FM, ou 6 + boîte à rythme sur 5 voix.
  - Oscillateur intégré pour le vibrato et la modulation d'amplitude.
  - Circuit intégré d'analyse et de jouer une voix ADPCM sur 4 bits.
  - Possibilité de jouer une voix avec ondes sinusoïdales complexes.
  - Circuit de contrôle du convertisseur numérique vers analogique (AD) / analogique vers numérique (DA) intégré.
  - De 32Ko à 256Ko de RAM pour le stockage de données ADPCM ou comme mémoire auxiliaire de la CPU.
  - Port d'entrée / sortie intégré sur 8 bits pour la scrutation du clavier.
  - Port d'entrée / sortie intégré sur 4 bits.
  - Deux compteurs (Timer) intégrés.
- 128Ko de ROM contenant le BIOS, les instructions BASIC étendues et un logiciel spécifique au fabricant (Firmware) de la cartouche.
- 4Ko de RAM pour le programme interne.
- DAC-LSI : Une ou deux puces YM-3014 Yamaha pour générer la sortie sonore analogique.
- Connecteur pour clavier musical. (Brochage non standardisé.)
- Prise(s) RCA pour la sortie sonore.
- Prise Jack pour l'entrée sonore (microphone externe).

## 10.2 Mappage de la mémoire du MSX-AUDIO

Le MSX-AUDIO possède une ROM de 128 Ko et une RAM de travail de 4 Ko. La ROM est divisée en 4 segments de 32 Ko. Ceux-ci sont mappés dans le même Slot comme décrit ci-dessous.

La carte mémoire est comme ça:

- Page 0000h~2FFFh = MBIOS (fixe)
- Page 3000h~3FFFh = 4 Ko de RAM de la cartouche MSX-AUDIO (fixe)
- Page 4000h~BFFFh = segment sélectionné (0 par défaut)

Les segments suivants sont sélectionnables par une écriture à 3FFFh (bits 0 et 1) dans le Slot du MSX-AUDIO. Cette adresse de commutation est reflétée à 7FFFh.

- Le segment 0 contient l'extension MSX-AUDIO BASIC (4000h~6FFFh) et le reflet de la RAM de 4 Ko (7000h~7FFFh).
- Le segment 1 est le firmware personnalisé (4000h~BFFFh)
- Le segment 2 correspond aux données ADPCM 1 (4000h~BFFFh)
- Le segment 3 correspond aux données ADPCM 2 (4000h~BFFFh)

## 10.3 Le MBIOS

Le MBIOS (Music Basic Input Output System) est un jeu de routines pour directement la puce Y8950 (MSX-Audio) afin de produire des sons FM, l'écriture ou la lecture ADPCM / PCM, le balayage du clavier musical, etc. Les routines disponibles sont expliquées dans les pages suivantes.

Sous MSX-DOS2 ou dans l'environnement du Disk Basic v.2.xx, utilisez le [BIOS étendu](#) (page 524) pour accéder aux routines du MBIOS en suivant la procédure ci-dessous.

1. Spécifiez l'adresse de la routine du MBIOS dans le registre HL.
2. Définissez les autres registres comme indiqué plus bas pour chaque MBIOS.
3. Spécifiez le contenu à mettre dans les registres IX et IY dans le cache BUF (0F55Eh).

| Adresse | Contenu                     |
|---------|-----------------------------|
| BUF + 0 | IX (8 bits de poids faible) |
| BUF + 1 | IX (8 bits de poids fort)   |
| BUF + 2 | IY (8 bits de poids faible) |
| BUF + 3 | IY (8 bits de poids fort)   |

Avant d'indiquer la liste des routines du MBOS, voici la description de sa zone de travail.

## Zone de travail du MBIOS

MBIOS permet de contrôler le canal du Y8950 maître et de l'esclave. Il contrôle également les 9 canaux sonore FM de chaque Y8950. Les programmes doivent spécifier le canal à utiliser au MBIOS en transmettant l'adresse de la zone de travail du canal correspondant. Celles-ci sont indiquées dans le tableau ci-dessous.

| Nom    | Adresse | Longueur | Utilisation                                        |
|--------|---------|----------|----------------------------------------------------|
| CHDB0  | 3000h   | 64       | Données du canal sonore FM 0                       |
| CHDB1  | 3040h   | 64       | Données du canal sonore FM 1                       |
| CHDB2  | 3080h   | 64       | Données du canal sonore FM 2                       |
| CHDB3  | 30C0h   | 64       | Données du canal sonore FM 3                       |
| CHDB4  | 3100h   | 64       | Données du canal sonore FM 4                       |
| CHDB5  | 3140h   | 64       | Données du canal sonore FM 5                       |
| CHDB6  | 3180h   | 64       | Données du canal sonore FM 6                       |
| CHDB7  | 31C0h   | 64       | Données du canal sonore FM 7                       |
| CHDB8  | 3200h   | 64       | Données du canal sonore FM 8                       |
| MIDB_M | 3280h   | 64       | Données des instruments de musique (canal maitre)  |
| MDIB_S | 32C0h   | 64       | Données des instruments de musique (canal esclave) |
| UVL    | 3700h   | 1024     | Données sonore FM de l'utilisateur                 |

## Description d'un MIDB

Il y a un MIDB (Music Instrument Data Block) de 64 octets pour chaque canal maître et chaque canal esclave (MIDB-M (3280H) et MIDB-S (32C0H)). Lorsque un programme appelle le MBIOS, il peut se référer au contenu du MIDB du canal correspondant, mais il ne doit pas en modifier le contenu. Le contenu d'un MIDB est le suivant.

| MIDB + | Contenu                             |
|--------|-------------------------------------|
| 0      | Inutilisé                           |
| 1      | Inutilisé                           |
| 2      | YM_TIM1                             |
| 3      | YM_TIM2                             |
| 4~17   | Inutilisé                           |
| 18     | YMA_BIAS                            |
| 19~24  | Inutilisé                           |
| 25     | YMA_AUDIO                           |
| 26~31  | Inutilisé                           |
| 32     | YMA_TRANS (8 bits de poids faible)  |
| 33     | YMA_TRANS (8 bits de poids fort)    |
| 34     | YMA_LFO                             |
| 35     | YMA_RAM                             |
| 36     | ZMA_FLAG                            |
| 37     | YMA_PDB (8 bits de poids faible)    |
| 38     | YMA_PDB (8 bits de poids fort)      |
| 39     | ZMA_PH_FILTER                       |
| 40     | ZMA_PH_TL                           |
| 41     | ZMA_PH_AR (8 bits de poids faible)  |
| 42     | ZMA_PH_AR (8 bits de poids fort)    |
| 43     | ZMA_PH_DIR (8 bits de poids faible) |
| 44     | ZMA_PH_DIR (8 bits de poids fort)   |
| 45     | ZMA_PH_SL                           |
| 46     | ZMA_PH_D2R (8 bits de poids faible) |
| 47     | ZMA_PH_D2R (8 bits de poids fort)   |
| 48     | ZMA_PH_RR (8 bits de poids faible)  |
| 49     | ZMA_PH_RR (8 bits de poids fort)    |
| 50     | ZMA_PH_EG (8 bits de poids faible)  |
| 51     | ZMA_PH_EG (8 bits de poids fort)    |
| 52     | ZMA_PH_STAT                         |
| 53~63  | Inutilisé                           |

## Description d'un CHDB

Un CHDB (CHannel Data Block) de 64 octets est créé pour chacun des neuf canaux sonore FM et sont utilisés par le MBIOS pour contrôler chaque canal.

Lorsque qu'un logiciel appelle une routine du MBIOS, l'adresse du CHDB du canal auquel la routine est destinée doit être spécifiée.

Étant donné que le contenu des registres du Y8950 sont pas lisible, le MBIOS copie leurs valeurs lors de l'écriture dans les registres dans le CHDB. Les programmes peuvent se référencer au contenu des CHDB, mais ne doivent pas modifier le contenu. Le contenu d'un CHDB est le suivant.

| CHDB + | Contenu                             |
|--------|-------------------------------------|
| 0      | YCAO0_MULTI                         |
| 1      | YCAO0_LS                            |
| 2      | YCAO0_AR                            |
| 3      | YCAO0_RR                            |
| 4      | YCAO0_VELS                          |
| 5      | YCAO0_VTL                           |
| 6      | Inutilisé                           |
| 7      | Inutilisé                           |
| 8      | YCAO1_MULTI                         |
| 9      | YCAO1_LS                            |
| 10     | YCAO1_AR                            |
| 11     | YCAO1_RR                            |
| 12     | YCAO1_VELS                          |
| 13     | YCAO1_VTL                           |
| 14     | Inutilisé                           |
| 15     | Inutilisé                           |
| 16     | YCA_VTRANS (8 bits de poids faible) |
| 17     | YCA_VTRANS (8 bits de poids fort)   |
| 18     | YCA_TRANS (8 bits de poids faible)  |
| 19     | YCA_TRANS (8 bits de poids fort)    |
| 20     | YCA_TRIG                            |
| 21     | YCA_VOL                             |
| 22     | YCA_FB                              |
| 23     | YCA_VEL                             |
| 24     | YCA_PITCH (8 bits de poids faible)  |
| 25     | YCA_PITCH (8 bits de poids fort)    |
| 26     | YCA_VOICE                           |
| 27     | ZCA_FLAG                            |
| 28     | ZC_CH                               |
| 29     | ZC_OP                               |
| 30     | ZC_COUNT (8 bits de poids faible)   |
| 31     | ZC_COUNT (8 bits de poids fort)     |

## Liste des routines du MBIOS

### 00090h (MSX-Audio ROM) **SV\_RESET**

Rôle : Initialiser le MSX-AUDIO.  
Entrée : Rien.  
Sortie : Rien.  
Modifie : Tous les registres.  
Note : La plupart des routines du MBIOS peuvent être utilisées en appelant SM\_AUDIO après avoir exécuté SV\_RESET.

### 00093h (MSX-Audio ROM) **SV\_DI**

Rôle : Désactiver le compteur d'interruption utilisateur.  
Entrée : Rien.  
Sortie : Rien.  
Modifie : Tous les registres.  
Note : Le compteur d'activation des interruptions utilisateur est incrémenté. Le traitement des interruptions utilisateur n'est effectué que lorsque ce compteur est à 0. Ce compteur étant composé de 8 bits, il ne peut être appelé plus de 255 fois de suite.

### 00096h (MSX-Audio ROM) **SV\_EI**

Rôle : Activer le compteur d'interruption utilisateur.  
Entrée : Rien.  
Sortie : Rien.  
Modifie : Tous les registres.  
Note : Si le compteur d'activation d'interruption utilisateur n'est pas 0, il est décrémenté de 1. Le traitement des interruptions utilisateur n'est effectué que lorsque ce compteur est à 0.

### 00099h (MSX-Audio ROM) **SV\_ADW**

Rôle : Écrire une donnée dans un registre du Y8950.  
Entrée : A = Donnée à écrire.  
C = Numéro du registre.  
IY = Adresse du MIDB sur le canal maître ou esclave.  
Sortie : F = L'indicateur CF sera mis à 1 si vous avez effectué une écriture sur le canal esclave du Y8950 alors qu'il est inexistant.  
Modifie : Tous les registres.  
Note : La routine désactive les interruptions pendant le traitement et les ré-active avant de rendre la main.



## 0009Ch (MSX-Audio ROM) **SV\_ADW\_DI**

Rôle : Écrire une donnée dans un registre du Y8950.

Entrée : A = Donnée à écrire.  
C = Numéro du registre.  
IY = Adresse du MIDB sur le canal maître ou esclave.

Sortie : F = L'indicateur CF sera mis à 1 si vous avez effectué une écriture sur le canal esclave du Y8950 alors qu'il est inexistant.

Modifie : Tous les registres.

Note : La routine désactive les interruptions.

## 000ABh (MSX-Audio ROM) **SV\_SETUP**

Rôle : Initialise les réglages de la fonction spécifiée.

Entrée : A = Code de la fonction. Les autres paramètres dépendent de la fonction choisie.

| Code | Nom      | Fonction                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0    | SM_AUDIO | Initialisation du musical.<br>C = Mode<br>Bit 7~3 = doit être mis à 0<br>Bit 2 = 9 voix FM (0), 6 voix FM + boîte à rythme (1)<br>Bit 1 = Canal maître : mode FM (0), mode CSM (1)<br>Bit 0 = Canal esclave : mode FM (0), mode CSM (1)<br>DE = Les bits du registre DE correspondant au numéro de canal sonore FM doivent être mis à 1 pour affecter ces canaux au clavier musical.<br>Note : Appelle les routines SC_CHDB et SM_INST. |
| 1    | SC_CHDB  | Initialisation d'une zone de travail CHDB. Les données de musicales sont initialisées avec des 0.<br>IY = Adresse du CHDB                                                                                                                                                                                                                                                                                                               |
| 2    | SM_INST  | Initialisation de la routine du son du clavier musical.                                                                                                                                                                                                                                                                                                                                                                                 |
| 3    | SM_MK    | Initialisation de la scrutation du clavier musical.<br>B = Connecter le clavier musical à un instrument (1)<br>C = Plage de vélocité lorsque vous appuyez sur le clavier musical. (0~15)                                                                                                                                                                                                                                                |

Sortie : Rien.

Modifie : Tous les registres.

## 000AEh (MSX-Audio ROM) **SV\_REAL**

Rôle : Effectue une opération en temps réel (sur les voix en cours de lecture, etc).

Entrée : A = Code de la fonction. Les autres paramètres dépendent de la fonction choisie.

Fonctions pour opérer sur un canal :

| Code | Nom          | Fonction                                                                                                                                                                                                                                                                                                                                                                                  |
|------|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 32   | RC_NOTE      | Key-on, Key-off du canal sonore FM spécifié.<br>Entrée : IX = Adresse du CHDB pour spécifier le canal FM.<br>DE = Hauteur (0~32767).<br>B = Temps de pression d'une touche (1~255)<br>C = Vitesse de pression d'une touche (0~15)<br>Sortie : rien                                                                                                                                        |
| 33   | RC_LEGATO    | Legato-on, Legato-Off du canal sonore FM spécifié.<br>Entrée : IX = Adresse du CHDB pour spécifier le canal FM.<br>DE = Hauteur (0~32767).<br>B = Temps de pression d'une touche (1~255)<br>Sortie : rien<br>Note : Contrairement au key-on, le legato-on ne démarre pas l'enveloppe, il est donc utilisé pour changer la hauteur et l'intensité du son une fois que vous appuyez dessus. |
| 34   | RC_DAMP      | Stopper la lecture du canal sonore FM en cours.<br>Entrée : IX = Adresse du CHDB pour spécifier le canal FM.                                                                                                                                                                                                                                                                              |
| 35   | RC_KON       | Key-on du canal sonore FM spécifié.<br>Entrée : IX = Adresse du CHDB pour spécifier le canal FM.<br>DE = Hauteur (0~32767).<br>C = Vitesse de pression d'une touche (0~15)<br>Sortie : rien                                                                                                                                                                                               |
| 36   | RC_LEGATO_ON | Legato-on du canal sonore FM spécifié.<br>Entrée : IX = Adresse du CHDB pour spécifier le canal FM.<br>DE = Hauteur (0~32767).<br>Sortie : rien                                                                                                                                                                                                                                           |
| 37   | RC_KOFF      | Key-off du canal sonore FM spécifié.<br>Entrée : IX = Adresse du CHDB pour spécifier le canal FM.<br>Sortie : rien                                                                                                                                                                                                                                                                        |
| 38   | RCA_PARAM    | Réglage du canal sonore FM en cours.<br>Entrée : IX = Adresse du CHDB pour spécifier le canal FM.<br>DE = Donnée de réglage.<br>C = Décalage à partir de l'adresse du CHDB des paramètres en temps réel.<br>Sortie : rien                                                                                                                                                                 |

|    |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 39 | RCA_VOICE  | Réglage de la tonalité de la voix du canal sonore FM.<br>Entrée : IX = Adresse du CHDB pour spécifier le canal FM.<br>C = Tonalité (0~63)<br>Sortie : rien                                                                                                                                                                                                                                                                                                                                                                |
| 40 | RCA_VPARAM | Paramétrer une voix du canal sonore FM.<br>Entrée : IX = Adresse du CHDB pour spécifier le canal FM.<br>DE = Décalage à partir de l'adresse du CHDB des paramètres en temps réel.<br>C = Tonalité (0~63)<br>Sortie : rien<br>Note : Les données suivantes peuvent être définies avec cette fonction.<br>YCAO0_MULTI      CAO1_MULTI<br>CAO0_LS            CAO1_LS<br>CAO0_AR            YCAO1_AR<br>CAO0_RR            YCAO1_RR<br>CAO0_VELS        YCAO1_VELS<br>CAO0_VTL          YCAO1_VTL<br>YCA_VTRANS        YCA_FB |
| 41 | RCA_VOICEP | Charger les données de la voix du canal sonore FM.<br>Entrée : IX = Adresse du CHDB pour spécifier le canal FM.<br>BC = Adresse des données du timbre. Ces données sont sur 32 octets comme suit.<br>Octets 0~7 = Nom du timbre (V_NAME).<br>Octet 8~9 = Valeur de transposition lors du calcul de la hauteur (V_TRANS).<br>Octet 10 = Arguments (V_ARG)<br>Bit 0 = Connexion des deux opérateurs. (0 = mode de modulation de fréquence en série, 1 = Sinus parallèle en mode combiné)<br>Sortie : rien                   |

Fonctions pour les instruments maitres :

| Code | Nom       | Fonction                                                                |
|------|-----------|-------------------------------------------------------------------------|
| 16   | RM_TIMER  | Activer ou désactiver le compteur d'interruption.                       |
| 17   | RM_TIM1   | Régler le Timer 1.                                                      |
| 18   | RM_TIM2   | Régler le Timer 2.                                                      |
| 19   | RM_TEMPO  | Timer 2 : Régler le tempo.                                              |
| 20   | RM_DAMP   | Forcer l'arrêt de toutes les voix FM en cours de lecture.               |
| 21   | RM_PERC   | Jouer les voix rythmiques.                                              |
| 22   | RMA_MK    | Scruter le clavier musical et indiquer son état.                        |
| 23   | RMA_LFO   | Régler la profondeur du vibrato et de la modulation d'amplitude.        |
| 24   | RMA_TRANS | Transposer le réglage du son et les notes suivantes.                    |
| 28   | RM_UTEMPR | Convertir la hauteur du tempérament à celle du tempérament égal défini. |
| 44   | RM_PVEL   | Définir la vitesse de chaque voix rythmique.                            |

## Fonctions pour les Instruments :

| Code | Nom        | Fonction                                                                   |
|------|------------|----------------------------------------------------------------------------|
| 48   | RI_DAMP    | Forcer l'arrêt de tous les canaux sonore FM attribués.                     |
| 49   | RI_ALLOFF  | Key-off de tous les canaux de sonore FM attribués.                         |
| 50   | RI_EVENT   | Key-on et key-off en convertissant le tempérament de la hauteur spécifiée. |
| 51   | RI_PCHB    | Régler la courbure tonal (pitch bend).                                     |
| 52   | RI_PCHBR   | Régler le degré qui influe sur la courbure tonal (pitch bend).             |
| 53   | RIA_PARAM  | Régler en temps réel les paramètres des canaux sonore FM.                  |
| 54   | RIA_VOICE  | Régler la voix avec le numéro de tonalité du canal sonore FM.              |
| 55   | RIA_VPARAM | Régler les paramètres de voix pour les canaux sonore FM.                   |
| 56   | RIA_VOICEP | Régler la voix avec les données audio du canal sonore FM.                  |

## Fonctions ADPCM / PCM :

| Code | Nom              | Fonction                                                                                                                       |
|------|------------------|--------------------------------------------------------------------------------------------------------------------------------|
| 0    | RM_MOVE_DI       | Transférer des données ADPCM/PCM d'un appareil à l'autre.                                                                      |
| 1    | RM_TRACE_DI      | Tracer des données ADPCM en fonction de la prédiction initiale et de la largeur de quantification                              |
| 26   | RM_READ_DI       | Transférer 256 octets de données ADPCM/PCM d'un périphérique vers la RAM principale                                            |
| 27   | RM_WRITE_DI      | Transférer 256 octets de données ADPCM/PCM de la RAM principale vers le périphérique                                           |
| 2    | RM_CONV_PCM_DI   | Convertir des données ADPCM en données PCM sur la base des valeurs de prédiction initiales et de la largeur de quantification. |
| 3    | RM_CONV_ADPCM_DI | Convertir des données PCM en données ADPCM sur la base des valeurs de prédiction initiales et de la largeur de quantification. |
| 4    | RMA_DAC_BIAS     | Régler le volume (registre 17h du Y8950) pour la lecture PCM.                                                                  |
| 5    | RMA_DAC_DI       | Jouer des données PCM.                                                                                                         |
| 6    | RMA_ADC_DI       | Enregistrer des données PCM.                                                                                                   |
| 7    | RMA_ADPCM_BIAS   | Régler le volume pour la lecture ADPCM.                                                                                        |
| 8    | RMA_ADPLAY_DI    | Lire des données ADPCM en mode non local.                                                                                      |
| 9    | RMA_ADREC_DI     | Enregistrer des données ADPCM en mode non local.                                                                               |
| 43   | RMA_ADPLY_SAMPLE | Modifier la fréquence d'échantillonnage pendant la lecture en mode local.                                                      |
| 10   | RMA_BREAK        | Interruption de la lecture/enregistrement en mode local.                                                                       |
| 11   | RMA_ADPLAY       | Lire des données ADPCM en mode local                                                                                           |

|    |                  |                                                                                         |
|----|------------------|-----------------------------------------------------------------------------------------|
| 12 | RMA_ADREC        | Enregistrer des données ADPCM en mode local.                                            |
| 42 | RMA_ADPLAYLP     | Lire des données ADPCM en mode local (en boucle)                                        |
| 13 | RMA_PHASE_SET_DI | Convertir des données PCM de 256 octets en données ADPCM dans la RAM principale.        |
| 14 | RMA_PHASE_EG     | Paramétrer les données de l'enveloppe.                                                  |
| 15 | RMA_PHASE_EVENT  | Key-on, key-off de la hauteur spécifiée par la simulation du clavier d'échantillonnage. |

Fonctions pour le Play-back CSM (Composite Sine Wave) :

| Code | Nom        | Fonction               |
|------|------------|------------------------|
| 25   | RMA_CSM_DI | Jouer les données CSM. |

Sortie : F = L'indicateur CF sera mis à 1 si des paramètres entrées sont incorrects.

Modifie : Tous les registres.

## 10.4 Les registres du Y8950

Les registres du Y8950 sont accessibles via les ports d'entrée/sorties suivants.

| Ports      | Rôle                                                                                                                                                                                                   |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| C0h et C1h | Accès au MSX Audio (canal maitre)<br>Port C0h = Numéro de registre (écriture seule) ;<br>Port C1h = Donnée du canal maitre (écriture et lecture ou écriture seule selon le registre que l'on accède).  |
| C2h et C3h | Accès au MSX Audio (canal esclave)<br>Port C2h = Numéro de registre (écriture seule) ;<br>Port C3h = Donnée du canal maitre (écriture et lecture ou écriture seule selon le registre que l'on accède). |

Vous devez respecter un délais de 12 cycles entre chaque écriture de numéro de registre, et 84 cycles entre chaque écriture de donnée (12 pour les registres de contrôle).

Ces ports E/S sont paramétrables en écrivant le type de configuration à l'adresse 3FFEh (bits 1 et 0) comme suit.

- 00 = Désactiver les ports C0h~C3h. (Valeur initiale)
- 01 = Attribuer les ports C0h~C1h pour accéder aux registres du Y8950.
- 10 = Attribuer les ports C2h~C3h pour accéder aux registres du Y8950.
- 11 = Attribuer les ports C0h~C1h et C2h~C3h (pour les cartouches avec deux Y8950).

### Les registres de contrôle

Les registres de contrôle sont accessibles en écriture mais aussi en lecture pour quelques uns.

|              |       |       |       |       |       |       |       |       |
|--------------|-------|-------|-------|-------|-------|-------|-------|-------|
|              | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
| Registre 1 : | TEST  |       |       |       |       |       |       |       |

Ce registre est utilisée lors du test du fonctionnement interne du LSI. Il contient 0 lorsque ça fonctionne correctement.

|              |         |       |       |       |       |       |       |       |
|--------------|---------|-------|-------|-------|-------|-------|-------|-------|
|              | bit 7   | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
| Registre 2 : | TIMER 1 |       |       |       |       |       |       |       |
| Registre 3 : | TIMER 2 |       |       |       |       |       |       |       |

**TIMER 1** est incrémenté toutes les 80 µs et **TIMER 2** toutes les 320 µs. En cas de dépassement de la valeur spécifiée, l'indicateur correspondant du registre de statut est mis à 1 et une demande d'interruption est envoyée à la CPU.

|              | bit 7      | bit 6     | bit 5     | bit 4      | bit 3     | bit 2 | bit 1 | bit 0 |
|--------------|------------|-----------|-----------|------------|-----------|-------|-------|-------|
| Registre 4 : | IRQ<br>RST | T1<br>MSK | T2<br>MSK | EOS<br>MSK | BR<br>MSK | -     | ST2   | ST1   |

**ST1** (Start Timer 1) = Mettre ce bit à 1 pour démarrer le TIMER 1, 0 pour le stopper.

**ST2** (Start Timer 2) = Mettre ce bit à 1 pour démarrer le TIMER 2, 0 pour le stopper.

**BR MSK** (Buffer Ready Mask) = Mettre ce bit à 1 pour masquer les requêtes de lecture/d'écriture de donnée pendant le transfert de données entre le CPU et l'ADPCM ou le stockage externe.

**EOS MSK** (End Mask) = Mettre ce bit à 1 pour masquer l'indicateur de fin de lecture/d'écriture ADPCM ou de stockage externe ou d'une numérisation sonore.

**T2 MSK** (Timer 2 Mask) = Mettre ce bit à 1 pour mettre TIMER 2 à 0.

**T1 MSK** (Timer 1 Mask) = Mettre ce bit à 1 pour mettre TIMER 1 à 0.

**IRQ RST** (IRQ Reset) = Mettre ce bit à 1 pour initialiser tous les indicateurs des timers (0) et IRQ (1) du registre d'état. Les bits 0~6 sont ignorés lorsque ce bit est à 1.

|              | bit 7        | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|--------------|--------------|-------|-------|-------|-------|-------|-------|-------|
| Registre 5 : | Keyboard IN  |       |       |       |       |       |       |       |
| Registre 6 : | Keyboard OUT |       |       |       |       |       |       |       |

**Keyboard IN / OUT** permet de lire / d'écrire les données en provenance du clavier musical.

|              | bit 7 | bit 6 | bit 5       | bit 4 | bit 3     | bit 2 | bit 1 | bit 0 |
|--------------|-------|-------|-------------|-------|-----------|-------|-------|-------|
| Registre 7 : | START | REC   | MEM<br>DATA | REPT  | SP<br>OFF | -     | -     | RST   |

**RST** (Reset) = Si ce bit est mis à 1, en tant que source de données mémoire externe comme

**SP OFF** (Speaker OFF) = Ce bit est contrôle la broche SP-OFF. Lorsqu'il est mis à 1, la broche SP-OFF est mise à 0, afin de protéger le haut-parleur pendant une analyse ADPCM ou un conversion analogique-numérique.

**REPT** (Repeat) = Mettre ce bit à 1 pour répéter la synthèse vocale ADPCM en cours.

**MEM DATA** (Memory Data) = Mettez ce bit à 1 pour accéder à la mémoire externe.

**REC** (Record) = Mettre ce bit à 1 pour la analyse vocale ADPCM ou l'écriture de données de la CPU à la mémoire externe.

**START** = Mettre ce bit à 1 pour analyser / jouer les données ADPCM. Ainsi, le temps de démarrage dépendra de l'emplacement les données (CPU ou mémoire externe). Lorsque l'emplacement des données est le CPU, il démarre lorsque le registre 15 est lu ou que l'on y écrit. Lorsqu'il s'agit de la mémoire externe, il démarre lorsque le bit START est mis à 1. Par conséquent, lors de l'accès à la mémoire externe, toutes les autres conditions doivent être remplies avant de mettre le bit START à 1. Lorsque vous mettez le bit START à 0, mettez d'abord le bit START à 0 pour réinitialiser les données restantes.

|              | bit 7 | bit 6    | bit 5 | bit 4 | bit 3  | bit 2 | bit 1 | bit 0 |
|--------------|-------|----------|-------|-------|--------|-------|-------|-------|
| Registre 8 : | CSM   | NOTE SEL | -     | -     | SAMPLE | DA/AD | 64K   | ROM   |

**ROM** = Mettre ce bit à 1 pour sélectionner la ROM externe, 0 pour la RAM.

**64K** = Ce bit est utilisé pour spécifier le type de mémoire externe (0 pour DRAM de type 256kbit, 1 pour 64Kbit)

**DA/AD** = Ce bit est utilisé en combinaison avec le bit SAMPLE ci-dessous. Il permet d'utiliser la sortie MO pour envoyer des données envoyées via &15~\$17 lorsqu'il est à 1. Si le bit DA/AD est à 0 et SAMPLE à 1, la conversion analogique-numérique commencera. Mettez les deux à 0 pour la sortie MUSIC.

**SAMPLE** = Ce bit sert à activer le timer pour la conversion analogique-numérique / numérique-analogique.

**NOTE SEL** = Ce bit contrôle le point de séparation de la division du clavier dans une octave. Si 0, le contrôle des bits qui indiquent la séparation se fait avec les 8 bits de poids fort de F-Number. (voir les tableaux ci-dessous et les explications sur F-Number/BLOCK pour plus de précision)

Lorsque NOTE SEL = 0

|                               |   |   |   |   |   |   |   |   |
|-------------------------------|---|---|---|---|---|---|---|---|
| Octave                        | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| BLOCK Data                    | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Bit de poids fort de F-Number | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Second F-Number               | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| Nombre de séparation          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Lorsque NOTE SEL = 1

|                               |   |   |   |   |   |   |   |   |
|-------------------------------|---|---|---|---|---|---|---|---|
| Octave                        | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| BLOCK Data                    | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Bit de poids fort de F-Number | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| Second F-Number               | - | - | - | - | - | - | - | - |
| Nombre de séparation          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

**CSM** (Composite Sine Wave) = Mettre à 1 pour le mode CSM (onde sinusoïdale composite). Dans ce cas, tous les canaux doivent être à l'état Key-OFF.

|               | bit 7                            | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---------------|----------------------------------|-------|-------|-------|-------|-------|-------|-------|
| Registre 9 :  | START ADD (bits de poids faible) |       |       |       |       |       |       |       |
| Registre 10 : | START ADD (bits de poids fort)   |       |       |       |       |       |       |       |

Ces deux registres servent à spécifier l'adresse de début à accéder dans la mémoire externe (par l'ADPCM / la CPU). Notez que la façon de spécifier est légèrement différente entre ROM et RAM.



|               | bit 7                           | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---------------|---------------------------------|-------|-------|-------|-------|-------|-------|-------|
| Registre 11 : | STOP ADD (bits de poids faible) |       |       |       |       |       |       |       |
| Registre 12 : | STOP ADD (bits de poids fort)   |       |       |       |       |       |       |       |

Ces deux registres servent à spécifier l'adresse de fin à accéder dans la mémoire externe (par l'ADPCM / la CPU). Notez que la façon de spécifier est légèrement différente entre ROM et RAM.

|               | bit 7                           | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---------------|---------------------------------|-------|-------|-------|-------|-------|-------|-------|
| Registre 13 : | PRESCALE (bits de poids faible) |       |       |       |       |       |       |       |
| Registre 14 : | PRESCALE (bits de poids fort)   |       |       |       |       |       |       |       |

|               | bit 7      | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---------------|------------|-------|-------|-------|-------|-------|-------|-------|
| Registre 15 : | ADPCM-DATA |       |       |       |       |       |       |       |

Ce registre peut être lu.

|               | bit 7                          | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---------------|--------------------------------|-------|-------|-------|-------|-------|-------|-------|
| Registre 16 : | DELTA-N (bits de poids faible) |       |       |       |       |       |       |       |
| Registre 17 : | DELTA-N (bits de poids fort)   |       |       |       |       |       |       |       |

|               | bit 7   | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---------------|---------|-------|-------|-------|-------|-------|-------|-------|
| Registre 18 : | EG-CTRL |       |       |       |       |       |       |       |

|               | bit 7                              | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---------------|------------------------------------|-------|-------|-------|-------|-------|-------|-------|
| Registre 21 : | DAC DATA (bits de poids fort)      |       |       |       |       |       |       |       |
| Registre 22 : | DAC DATA<br>(bits de poids faible) | -     | -     | -     | -     | -     | -     | -     |

|               | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2   | bit 1   | bit 0   |
|---------------|-------|-------|-------|-------|-------|---------|---------|---------|
| Registre 23 : | -     | -     | -     | -     | -     | SHIFT 2 | SHIFT 1 | SHIFT 0 |

|               | bit 7 | bit 6 | bit 5 | bit 4 | bit 3    | bit 2 | bit 1 | bit 0 |
|---------------|-------|-------|-------|-------|----------|-------|-------|-------|
| Registre 24 : | -     | -     | -     | -     | I/O CTRL |       |       |       |

Bit 0 = Application interne (sortie)

Bit 1 = Application interne (sortie)

Bit 2 = Sélection du MSX-AUDIO BASIC (0) ou du firmware (1)

Bit 3 = Utilisation du filtre pour l'ADPCM/PCM (0) ou le son FM (1).

|               | bit 7 | bit 6 | bit 5 | bit 4 | bit 3    | bit 2 | bit 1 | bit 0 |
|---------------|-------|-------|-------|-------|----------|-------|-------|-------|
| Registre 25 : | -     | -     | -     | -     | I/O DATA |       |       |       |

Ce registre peut être lu.

|               | bit 7    | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---------------|----------|-------|-------|-------|-------|-------|-------|-------|
| Registre 26 : | PCM-DATA |       |       |       |       |       |       |       |

Ce registre peut être lu.

## 11 Le MSX-MIDI

Le MSX-MIDI apparaît dans le standard à partir du MSX turbo R. Il ne peut pas être utilisé sur MSX, MSX2 et MSX2+. Celui-ci apporte des fonctionnalités MIDI et des instructions BASIC étendues qui permettent d'utiliser des appareils MIDI.

Une prise midi envoi ou reçoit les données en série bit par bit, un peu comme une interface RS-232. Le signal est transmis par modulation d'amplitude à une vitesse de 31250 bauds. Chaque octet à transmettre doivent être précédé d'un bit qui signal le départ et suivi d'un bit d'arrêt. Comme il n'y a pas de bit de parité, cela fait au total 10 bits pour chaque octet transmis.

Contrairement au MSX-MUSIC, il n'y a pas de BIOS. Les programmes pour MSX-MIDI, hormis ceux en Basic, doivent accéder directement au matériel via des ports E/S.

Lorsque le MSX-MIDI n'est pas intégré au MSX, il peut être ajouté via une cartouche externe. Un MSX-MIDI externe peut s'utiliser en parallèle avec un MSX-MIDI interne.

MSX-MIDI est composé des éléments suivants :

- Une Interface MIDI qui est composée d'une puce « 8251 » qui s'occupe des transferts de donnée, et d'une autre « 8253 » ou « 8254 » pour générer un signal en bauds et les timings.
- Une ROM MSX-MIDI (16K bytes) qui contient uniquement des instructions Basic étendues pour le MIDI. Dans le cas d'un MSX-MIDI interne, ces instructions se trouvent dans la ROM du MSX-Music (4000h~7FFFh) puisque le MSX-MIDI est une extension de celle-ci.

### 11.1 Accéder au MSX-MIDI

Selon que le MSX-MIDI soit interne ou externe, la configuration du matériel et la méthode d'accès sont différents.

## Accéder au MSX-MIDI interne

Envoi / Réception de donnée

| Port            | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|-----------------|-------|-------|-------|-------|-------|-------|-------|-------|
| E8h en écriture | RXD7  | RXD6  | RXD5  | RXD4  | RXD3  | RXD2  | RXD1  | RXD0  |
| E8h en lecture  | TXD7  | TXD6  | TXD5  | TXD4  | TXD3  | TXD2  | TXD1  | TXD0  |

### Description des bits :

RXD7~RXD0 = Donnée à envoyer à la 8251

TXD7~TXD0 = Donnée de la 8251 reçue

Accès au registre de mode / commande

| Port            | bit 7   | bit 6   | bit 5    | bit 4    | bit 3     | bit 2   | bit 1    | bit 0    |
|-----------------|---------|---------|----------|----------|-----------|---------|----------|----------|
| E9h en écriture | S2 / EH | S1 / IR | EP / RIE | PEN / ER | L2 / SBRK | L1 / RE | B2 / TIE | B1 / TEN |
| E9h en lecture  | DSR     | BRK     | FE       | OE       | PE        | EMPT    | RRDY     | TRDY     |

### Description des bits en écriture :

TEN = Activation de la transmission MIDI OUT (1 pour activer, 0 pour désactiver)

TIE = Activation d'interruption du Timer 2 (8253) (1 pour activer, 0 pour désactiver)

RE = Activation de la réception MIDI IN (1 pour activer, 0 pour désactiver)

SBRK = Mettre à 0

ER = Initialisation des indicateurs d'erreur (1 pour mettre à 0 les bits FE, OE et PE, 0 ne fait rien)

RIE = Activation de l'interruption MIDI IN (1 pour activer, 0 pour désactiver)

IR = Mettre à 0

EH = Mettre à 0

B2~B1 = Niveau du débit (500KHz/8\*B)

L2~L1 = Longueur des caractères (00 pour 5 bits, 01 pour 6 bits, 10 pour 7 bits, 11 pour 8 bits)

PEN = Activation de la parité (1 pour activer, 0 pour désactiver)

EP = Contrôle de parité (0 pour impaire, 1 pour paire)

S2~S1 = Nombre de bit d'arrêt (mettez 01 pour 1 bit !) (10 pour 1,5 bit et 11 pour 2 bits)

### Description des bits en lecture :

TRDY = État de transmission (1 lorsque la transmission est prête)

RRDY = État du cache de réception (1 lorsqu'une donnée est disponible)

EMPT = État du cache de transmission (1 lorsque le cache de transmission est vide)

PE = Indicateur d'erreur de parité (1 lorsqu'une erreur s'est produite)

OE = Indicateur d'erreur de dépassement de vitesse (1 lorsqu'une erreur s'est produite)

FE = Indicateur d'erreur de trame (1 lorsqu'une erreur s'est produite)

BRK = Détection de code de rupture (code « Break ») (1 = Détecté)

DSR = Indicateur d'interruption du Timer (1 lorsqu'une interruption s'est produite) (8253)

### Signal de de verrouillage de la broche OUT2 de la 8253

| Port            | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|-----------------|-------|-------|-------|-------|-------|-------|-------|-------|
| EAh en écriture | -     | -     | -     | -     | -     | -     | -     | -     |
| EBh en écriture | -     | -     | -     | -     | -     | -     | -     | -     |

Une écriture quelconque sur l'un de ces deux ports initialise le Timer 2.

### Compteurs (8253)

| Port                   | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|------------------------|-------|-------|-------|-------|-------|-------|-------|-------|
| ECh (écriture/lecture) | CT07  | CT06  | CT05  | CT04  | CT03  | CT02  | CT01  | CT00  |
| EDh (écriture/lecture) | CT17  | CT16  | CT15  | CT14  | CT13  | CT12  | CT11  | CT10  |
| EEh (écriture/lecture) | CT27  | CT26  | CT25  | CT24  | CT23  | CT22  | CT21  | CT20  |

#### Description des bits :

CT07~CT00 = Timer 0

CT17~CT10 = Timer 1

CT27~CT20 = Timer 2

Les compteurs ont les fonctions suivantes:

- Le Timer 0 de la 8251 est utilisé comme générateur de signal en bauds. Un signal d'horloge de 4 MHz entre par la broche CLK. Le 8251 doit être configuré pour transmettre une fréquence de 500 KHz (8 fois moins). Pour cela, utilisez le mode 3 (le mode onde carrée divisée par N).
- Le Timer 1 est utilisé comme compteur à usage général. La sortie du compteur 2 entre par la broche CLK.
- Le Timer 2 est utilisé pour les demandes d'interruption au CPU (En BASIC, une interruption est provoquée toutes les 5 ms). Normalement, le mode 2 est utilisé (le mode diviser par N). Lorsque la broche OUT2 passe à 0, une demande d'interruption est envoyée au CPU via le circuit de verrouillage. Le signal d'horloge de 4 MHz entre par la broche CLK.

### Registre de commande (8253)

| Port            | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|-----------------|-------|-------|-------|-------|-------|-------|-------|-------|
| EFh en écriture | -     | -     | -     | -     | -     | -     | -     | -     |
| EFh en lecture  | SC1   | SC0   | RW1   | RW0   | M2    | M1    | M0    | BCD   |

#### Description des bits :

BCD = Sélection du Timer en binaire / BCD

SC1~SC0 = Numéro du Timer ou de la commande à sélectionner

M2~M0 = Mode du Timer

RW1~RW0 = Timer en mode lecture / écriture

## Accéder au MSX-MIDI externe

Les ports d'entrée/sortie d'une interface MSX-MIDI externe peut être modifiée en écrivant une valeur sur le port E2h.

Sélection des ports de la 8251

| Port            | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|-----------------|-------|-------|-------|-------|-------|-------|-------|-------|
| E2h en écriture | EN    | -     | -     | -     | -     | -     | -     | E8    |

### Description des bits :

E8 = Définition du numéro des ports d'accès (0 pour E8h et E9h, 1 pour E0h et E1h (valeur par défaut).  
Lorsque le bit E8 est à 0, l'accès aux ports de ECh à EFh est ignoré, et le Timer des interruptions est désactivé).

EN = Désactivation de l'interface MIDI (0 pour activer, 1 (valeur par défaut) pour désactiver).

Envoie / Réception de donnée

| Port                  | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|-----------------------|-------|-------|-------|-------|-------|-------|-------|-------|
| E0h / E8h en écriture | RXD7  | RXD6  | RXD5  | RXD4  | RXD3  | RXD2  | RXD1  | RXD0  |
| E0h / E8h en lecture  | TXD7  | TXD6  | TXD5  | TXD4  | TXD3  | TXD2  | TXD1  | TXD0  |

### Description des bits :

RXD7~RXD0 = Donnée à envoyer à la 8251

TXD7~TXD0 = Donnée de la 8251 reçue

### Registre de commande / mode

| Port                  | bit 7   | bit 6   | bit 5    | bit 4    | bit 3     | bit 2   | bit 1    | bit 0    |
|-----------------------|---------|---------|----------|----------|-----------|---------|----------|----------|
| E1h / E9h en écriture | S2 / EH | S1 / IR | EP / RIE | PEN / ER | L2 / SBRK | L1 / RE | B2 / TIE | B1 / TEN |
| E1h / E9h en lecture  | DSR     | BRK     | FE       | OE       | PE        | EMPT    | RRDY     | TRDY     |

#### Description des bits en écriture :

TEN = Activation de la transmission MIDI OUT (1 pour activer, 0 pour désactiver)

TIE = Activation d'interruption du Timer 2 (8253) (1 pour activer, 0 pour désactiver)

RE = Activation de la réception MIDI IN (1 pour activer, 0 pour désactiver)

SBRK = Mettre à 0

ER = Initialisation des indicateurs d'erreur (1 pour mettre à 0 les bits FE, OE et PE, 0 ne fait rien)

RIE = Activation de l'interruption MIDI IN (1 pour activer, 0 pour désactiver)

IR = Mettre à 0

EH = Mettre à 0

B2~B1 = Niveau du débit (500KHz/8\*B)

L2~L1 = Longueur des caractères (00 pour 5 bits, 01 pour 6 bits, 10 pour 7 bits, 11 pour 8 bits)

PEN = Activation de la parité (1 pour activer, 0 pour désactiver)

EP = Contrôle de parité (0 pour impaire, 1 pour paire)

S2~S1 = Nombre de bit d'arrêt (mettez 01 pour 1 bit !) (10 pour 1,5 bit et 11 pour 2 bits)

#### Description des bits en lecture :

TRDY = État de transmission (1 lorsque la transmission est prête)

RRDY = État du cache de réception (1 lorsqu'une donnée est disponible)

EMPT = État du cache de transmission (1 lorsque le cache de transmission est vide)

PE = Indicateur d'erreur de parité (1 lorsqu'une erreur s'est produite)

OE = Indicateur d'erreur de dépassement de vitesse (1 lorsqu'une erreur s'est produite)

FE = Indicateur d'erreur de trame (1 lorsqu'une erreur s'est produite)

BRK = Détection de code de rupture (code « Break ») (1 = Détecté)

DSR = Indicateur d'interruption du Timer (1 lorsqu'une interruption s'est produite) (8253)

### Signal de de verrouillage de la broche OUT2 de la 8253

| Port            | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|-----------------|-------|-------|-------|-------|-------|-------|-------|-------|
| EAh en écriture | -     | -     | -     | -     | -     | -     | -     | -     |
| EBh en écriture | -     | -     | -     | -     | -     | -     | -     | -     |

Une écriture quelconque sur l'un de ces deux ports initialise le Timer 2.

### Timers (8253)

| Port                   | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|------------------------|-------|-------|-------|-------|-------|-------|-------|-------|
| ECh (écriture/lecture) | CT07  | CT06  | CT05  | CT04  | CT03  | CT02  | CT01  | CT00  |
| EDh (écriture/lecture) | CT17  | CT16  | CT15  | CT14  | CT13  | CT12  | CT11  | CT10  |
| EEh (écriture/lecture) | CT27  | CT26  | CT25  | CT24  | CT23  | CT22  | CT21  | CT20  |

#### Description des bits :

CT00~CT07 = Timer 0

CT10~CT17 = Timer 1

CT20~CT27 = Timer 2

Les compteurs ont les fonctions suivantes:

- Le Timer 0 de la 8251 est utilisé comme générateur de signal en bauds. Un signal d'horloge de 4 MHz entre par la broche CLK. Le 8251 doit être configuré pour transmettre une fréquence de 500 KHz (8 fois moins). Pour cela, utilisez le mode 3 (le mode onde carrée divisée par N).
- Le Timer 1 est utilisé comme compteur à usage général. La sortie du compteur 2 entre par la broche CLK.
- Le Timer 2 est utilisé pour les demandes d'interruption au CPU (En BASIC, une interruption est provoquée toutes les 5 ms). Normalement, le mode 2 est utilisé (le mode diviser par N). Lorsque la broche OUT2 passe à 0, une demande d'interruption est envoyée au CPU via le circuit de verrouillage. Le signal d'horloge de 4 MHz entre par la broche CLK.

### Registre de commande du 8253

| Port            | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|-----------------|-------|-------|-------|-------|-------|-------|-------|-------|
| EFh en écriture | -     | -     | -     | -     | -     | -     | -     | -     |
| EFh en lecture  | SC1   | SC0   | RW1   | RW0   | M2    | M1    | M0    | BCD   |

#### Description des bits :

BCD = Sélection du Timer en binaire / BCD

SC1~SC0 = Numéro du Timer ou de la commande à sélectionner

M2~M0 = Mode du Timer

RW1~RW0 = Timer en mode lecture / écriture

## ***11.2 Les interruptions du MSX-MIDI***

Un MSX turbo R avec le MSX-MIDI en interne utilise les Hooks H.MDIN (0FF75h) et H.MDTM (0FF93h). Le Hook HMDIN permet de gérer les interruptions de l'entrée MIDI et H.MDTM les interruptions du Timer d'interruption (8253)

Lorsque le MSX-MIDI externe, seul le Hook H.KEYI (0FD9Ah) permet de gérer les interruptions du MSX-MIDI.

## 11.3 Programmation du MSX-MIDI

### Méthode pour distinguer le type interne ou externe

La présence du MSX-MIDI interne ou externe peut être déterminé de la façon suivante.

1. D'abord vérifiez que l'ordinateur n'est pas d'une génération antérieure au MSX turbo R. (si 002Dh en Main-ROM n'est pas inférieure à 3).
2. Si le bit 0 à l'adresse 002Eh en Main-ROM est à 1, c'est qu'il y a un MSX-MIDI interne.
3. La présence d'un MSX-MIDI externe peut être déterminé en cherchant le mot « MIDI » aux adresses 401Ch~401FH dans chacun des Slot.
4. Si une interface MSX-MIDI interne et une externe sont trouvées, celles-ci peuvent être utilisées en parallèle à condition que bit E8 du registre de sélection des ports de la 8251 reste toujours à 0 (valeur par défaut).

Voici un exemple de programme de détection en assembleur :

```
; Routine de recherche du MSX-MIDI
; Sortie: A = Le bit 0 indique la présence du MSX-MIDI interne
;           Le bit 1 indique la présence du MSX-MIDI externe

RDSLT    equ    000Ch        ;Lire l'octet d'un Slot
MSXVER    equ    002Dh        ;Version du MSX
MSXMIDI    equ    002Eh        ;Présence du MSX-MIDI interne
EXPTBL    equ    0FCC1h        ;Table des Slot étendus

MSXMDCHK:
    ld     a, (EXPTBL)
    ld     hl, MSXVER
    call   RDSLT              ;Lit l'indentifiant du MSX
    cp     3
    ld     a, 0
    ret    c                  ;Retour si MSX1, MSX2, MSX2+
    ld     a, (EXPTBL)
    ld     hl, MSXMIDI
    call   RDSLT              ;Lit l'indentifiant du MSX-MIDI interne
    and    1                  ;Garde le bit 1 de A
    ld     c, a               ;puis le place dans C

    ld     b, 4               ;Compteur

CHKLP1:
    push   bc                 ;Stocke le compteur
    ld     a, 4
    sub    b                  ;Numéro de Slot primaire
    ld     c, a               ;dans C
    ld     hl, EXPTBL
    ld     a, c
    add    a, 1               ;Définie EXPTBL
    ld     l, a
```



```

        ld    a,(hl)
        and   10000000b    ;Garde le bit 7 si Slot étendu
        jr    z,CHKLP3     ;Saut si Slot étendu
        ld    b,4          ;number of expanded Slot
CHKLP2:                                ;Recherche dans le Slot étendu
        push  bc
        ld    a,00100100b
        sub   b            ;001000pp
        rlc   a
        rlc   a            ;1000ss00
        or    c            ;1000sspp = Address du Slot
        call  CHKID        ;Chercher l'identifiant
        pop   bc
        jr    z,CHKLPP     ;Saute si MSX-MIDI externe trouvé
        djnz  CHKLP2       ;Slot secondaire suivant
        pop   bc
        jr    CHKLP4       ;Slot suivant

CHKLP3:                                ;Chercher dans le Slot primaire
        ld    a,c          ;Numéro de Slot
        call  CHKID        ;Chercher l'identifiant
        pop   bc
        jr    z,CHKLPF     ;Saute si MSX-MIDI externe trouvé
CHKLP4:
        djnz  CHKLP1       ;Chercher dans Slot suivant
        ld    a,c          ;Pas de MSX-MIDI externe
        ret

CHKLPP:
        pop   bc
CHKLPF:
        ld    a,2          ;MSX-MIDI externe trouvé
        or    c
        ret

; Effect : Recherche du MSX-MIDI externe
; Entrée : A = Numnéo de Slot
; Sortie : F = ZF à 1 si MSX-MIDI externe trouvé

ID_ADRS equ  401Ch          ;Adresse de l'identifiant

ID_TXT:
        db   'MIDI '
CHKID:
        push  bc
        ld    de,ID_TXT
        ld    hl,ID_ADRS
        ld    b,4          ;Longueur de l'identifiant
CHKID1:
        push  af          ;Stocke le numéro de Slot
        push  bc
        push  de
        call  RDSLT       ;Lecture d'un caractère
        pop   de
        pop   bc
        ld    c,a          ;Caractère actuel dans C
        ld    a,(de)       ;Récupère le caractère
        cp    c
        jr    nz,CHKID2    ;Saute si même caractère

```

```

    pop  af          ;Restitue le numéro de Slot
    inc  de          ;Caractère suivant
    inc  hl
    djnz CHKID1      ;Recherche du caractère suivant
    pop  bc          ;Restitue BC
    xor  a           ;Identifiant trouvé
    ret

CHKID2:
    pop  af          ;Restitue le numéro de Slot
    pop  bc          ;Restitue BC
    xor  a
    inc  a           ;Mauvais identifiant
    ret

```

## Temps d'attente entre l'envoi d'une commande et du mode de registre

Il est nécessaire de respecter un temps d'attente de 16 T-cycle (en mode Z80 à 3.579545MHz) entre l'envoi d'une commande et du mode de registre pour laisser le temps de traiter les données.

Exemple de routine en assembleur suivante pour effectuer le temps d'attente :

```

TIMERLSB equ 0E6h      ;Une écriture initialise le Timer du système
TIMERMSB equ 0E7h      ;Bits de poids fort du Timer du système

MID_WAIT:
    push af
    push bc
    ld    c,1
    out   (TIMERLSB),a  ;Initialiser le Timer
MID_WAITLP:
    in    a,(TIMERMSB)  ;Lire le Timer
    cp    c
    jr    c,MID_WAITLP  ;Boucle
    pop   bc
    pop   af
    ret

```

## Initialisation du MSX-MIDI

Par initialiser l'interface, réglez les paramètres avec les valeurs par défaut comme dans le programme d'exemple ci-dessous et initialiser le 8251 en écrivant 00h, 00h, 00h, 40h sur le port E9h en respectant le temps d'attente.

```
; Initialize MIDI Interface

TIMER0      equ    0ECh      ;Timer 0 du 8253
TIMER2      equ    0EEh      ;Timer 2 du 8253
TM_CMD      equ    0EFh      ;Registre de commande du 8253

INIMDP:
;
; MIDI baud rate generater
;
    ld      a,00010110b      ;Commande de controle du 8253
;          |||||++-----= Timer en mode binaire
;          |||++-----= Mode 3 : Générateur de signal (signal carré)
;          ||+-----= Lecture/Écriture de l'octet le moins significatif
;          ++-----= Timer 0 for Baud Rate Generater of 8251
    out     (TM_CMD),a
    ld      a,8              ;4MHz / 8 = 500KHz
    out     (TIMER0),a       ;Réglage des bits de poids fort du timer 0 (8253)
;
; Réglage du timer 2 à 5msec
;
    ld      a,10110100b      ;Commande de controle du 8253
;          |||||++-----= Timer en mode binaire
;          |||++-----= Mode 2 : Générateur de la cadence
;          ||+-----= Lecture/Écriture de 2 octets
;          ++-----= Timer 2 cadencé à 5msec
    out     (TM_CMD),a
    ld      hl,20000
    ld      a,l
    out     (TIMER2),a       ;Règle l'octet de poids faible du Timer du 8253
    ld      a,h
    out     (TIMER2),a       ;Règle l'octet de poids fort du Timer du 8253

;Initialise le 8251

UARTcmd     equ    0E9h      ;0E1h par défaut si MSX-MIDI externe

    xor     a
    out     (UARTcmd),a      ;Ecrit 0
    call    MID_WAIT
    out     (UARTcmd),a      ;Ecrit 0
    call    MID_WAIT
    out     (UARTcmd),a      ;Ecrit 0
    call    MID_WAIT
    ld      a,40h
    out     (UARTcmd),a      ;puis 40h
    call    MID_WAIT
```

## 12 Le MSX-JE

MSX-JE est le dispositif standard de saisie de texte japonais qui sert entre autre à convertir les Kana-Kanji sur MSX. Ce qui n'était pas normalisée avant l'apparition de MSX-JE. Plusieurs traitements de texte japonais possédaient leur propre système de saisie.

MSX-JE est apparu avec le traitement de texte "Japanese MSX WRITE" d'ASCII sortie en 1986 et depuis lors, la plupart des logiciels nécessitant une entrée japonaise sur MSX sont compatibles avec MSX-JE et sont conformes à cette norme.

Par exemple, le traitement de texte japonais "Bunshosaku Zaemon" de Sony (HBS-B012D) est vendu sans aucun système de saisie additionnel. Pour l'utiliser, il est donc nécessaire d'avoir une machine MSX avec le MSX-JE installé.

MSX-JE est installé dans la plupart des MSX2+. Le Kanji BASIC de la version 3 du MSX-BASIC est compatible avec MSX-JE donc, la saisie de texte japonais peut être faite sous BASIC.

## 13 Le système des disques et MSX-DOS

Dans ce livre, je présente uniquement les routines et la structure de la mémoire sous MSX-DOS ou Disk BASIC. Je ne parlerai pas de l'utilisation des commandes du MSX-DOS ou instructions du Disk-BASIC. Pour cela, veuillez vous référer à un manuel d'utilisation correspondant.

Les disques sont gérés par un contrôleur de disque dont les routines se trouvent dans une mémoire morte appelée « Disk-ROM ». Celle-ci est présente dans tous les MSX ayant un lecteur de disquette interne ou dans n'importe quelle interface de disque sauf le Quick-Disk.

Le système peut manipuler jusqu'à quatre Disk-ROM en même temps et un maximum de huit disques logiques. Ces ROM contiennent des routines spécifiques pour contrôler les disques, les instructions du Disk-BASIC, le noyau du MSX-DOS et un BIOS dont la table des sauts est décrite plus bas.

### 13.1 Le MSX-DOS 1 et le MSX-DOS 2

Bien que sous BASIC les disques peuvent fonctionner sur les MSX ayant au moins 16Ko de RAM, le MSX-DOS 1 nécessite un minimum de 64Ko. Quant au MSX-DOS 2, il a besoin d'un minimum de 128Ko (Memory Mapper) même sous BASIC.

Le MSX-DOS 2 est la seule mise à jour majeure officielle. Celle-ci apporte les principales nouveautés qui sont : la gestion de dossiers, de variables d'environnement, du Memory Mapper, de fichiers temporaires « Pipe » et la redirection des périphériques. La redirection peut-être désactivée avec la commande SET REDIR= OFF. Dès lors, le COMMAND2.COM fonctionnera de façon très similaire au MSX-DOS1. De plus, le MSX-DOS 2 gère mieux les erreurs.

Il y a trois versions officielles du MSX-DOS 2 :

#### 1. MSX-DOS 2 v2.20 :

Cette version a été développée par ASCII. Elle était vendue en cartouche avec ou sans Memory Mapper. Il existe aussi des versions non-officielles vendues en Europe et ailleurs. Les différences principales sont le Mapper de la ROM et les Kanji qui ne sont pas présents.

#### 2. MSX-DOS 2 v2.30 :

Elle était vendue avec le FS-A1ST. Les changements sont les suivants.

##### Noyau :

- Cette version est spécifique au MSX turbo R.
- Le Memory Mapper interne est sélectionné d'office en tant que mémoire principale.
- L'identifiant du volume est à nouveau vérifié à l'ouverture d'un fichier lorsqu'un fichier est déjà ouvert. Lorsque l'on change de disque entre temps, même si le lecteur, le chemin et le nom du fichier sont les mêmes, le fichier sera considéré comme différent. A condition, toutefois, que les disques soient au format MSX-DOS 2.

### COMMAND2.COM :

- Ajout de la variable d'environnement KHELP. En mode Kanji, le dossier /KHELP est pris en compte à la place de /HELP (si ils sont à la racine du disque et si la variable HELP n'a pas été modifié).
- Ajout de la variable d'environnement EXPERT qui permet d'exécuter les commandes MSX-DOS2 sur un disque formaté MSX-DOS1.

### Commandes externes :

- La commande CHKDSK prévient lorsque le type de média n'indique pas MSX-DOS2.
- Ajout de l'option /S à la commande DISKCOPY. Cette option permet de copier aussi le code de démarrage (code boot).

### Autres :

- Le R800 est sélectionné par défaut.
- Si l'on presse la touche [1] au démarrage jusqu'au bip sonore, le Z80 est sélectionnée et la Disk-ROM v1 est installée à la place de la Disk-ROM v2.
- Le Hook H.STKE (0FEDAh)

## 3. MSX-DOS 2 v2.31 :

Elle était vendue avec le FS-A1GT. Les changements par rapport à la v2.30 sont les suivants.

### Noyau :

- Cette version est toujours spécifique au MSX turbo R.
- Les caractères des noms de fichier sont traitées comme en Shift JIS même en mode ANK (sans Kanji). Auparavant, c'était le cas uniquement en mode Kanji. Les fonctions 5Bh (Parse Filename) et 5Dh (Check Character) ne tiennent plus compte si le mode Kanji est activé ou non.

### COMMAND2.COM :

- Certaines commandes renvoient leur paramètre dans la variable d'environnement correspondante afin d'être utilisable dans les paramètres des commandes entrées par la ligne de commande ou dans un fichier Batch en plaçant le caractère « % » juste devant et juste derrière le nom de la variable.

Par exemple, si vous entrez la commande suivante, vous pouvez voir le ou les chemins utilisés par défaut à place du mot « PATH » entré.

ECHO %PATH%

- Ajout de la commande IF pour exécuter des commandes sous condition.

Notez que les versions 2.30 et 2.31 du COMMAND2.COM et la version 2.30 du MSXDOS2.SYS sont utilisables avec le noyau v2.20 du MSX-DOS 2.20. Il n'y a pas de version 2.31 du MSXDOS2.SYS.

Il existe aussi une version améliorée du MSX-DOS 2 par [Konamiman](#) qui possède le code source du MSX-DOS 2 dont le COMMAMD2.COM v2.44 provient de [The New Image](#). Veuillez vous référer à la documentation pour toutes informations à son sujet.

Le MSX-DOS 2 nécessite un MSX2 ou plus récent avec un minimum de 128Ko de RAM. Sous MSX-DOS 2, la mémoire est répartie de la même façon qu'avec le MSX-DOS 1 à la différence que des pages du Memory Mapper sont réservées pour le stockage temporaire de données. Pour cela, la Disk-ROM v2.xx installe des routines pour manipuler le Memory Mapper. On y accède via le Bios étendu. (Voir au chapitre 13.13 sur le [Bios étendu](#) page 524).

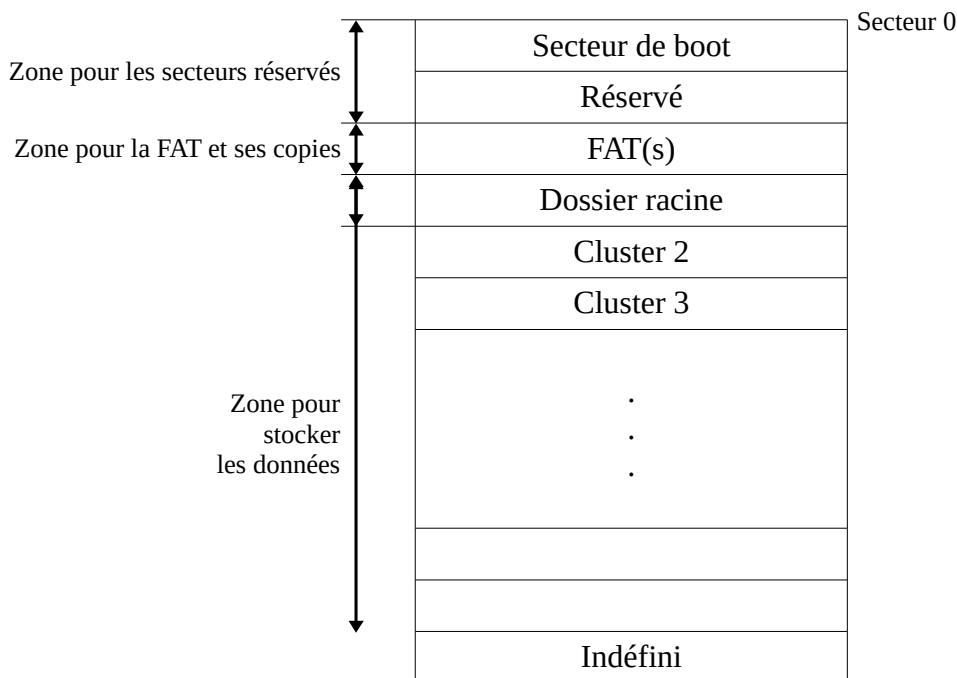
## 13.2 Format d'un disque

Le format des disques utilisés sur MSX en standard est appelé « FAT12 ». FAT est l'acronyme anglais de « File Allocation Table » et le 12 indique que les données de la FAT sont sur 12 bits. Le format FAT a évolué en différentes versions (FAT12, FAT16, FAT32, etc) afin de pouvoir stocker de plus gros fichiers et de plus nombreux. Sur MSX, il a évolué jusqu'au format FAT16 grâce à un patch sur le MSX-DOS2 fait par Okei puis, avec Nextor développé par Konamiman.

Seules les interfaces disques les plus récentes sont compatibles FAT16 car cela nécessite un logiciel de formatage spécifique. Les longs noms de fichiers ne sont pas gérés donc faites attention à ne pas en copier sur les disques MSX car bien qu'ils puissent être lus, un effacement ou un déplacement d'un fichier au long nom avec le MSX corrompra probablement la FAT.

Une fois formaté, le contenu des disques FAT12/16 pour MSX sont divisées en secteur de 512 octets. Un certain nombre de secteurs, qui varient selon la FAT utilisée et la taille du disque, constituent des Cluster.

Voici comment se présente un disque formaté en FAT12/16.



Les premiers octets du secteur de Boot contient les données suivantes que l'on appelle BPB (Bios Parameter Block).

| Position | Taille | Description                                                                                                     |
|----------|--------|-----------------------------------------------------------------------------------------------------------------|
| 0        | 3      | Codes de saut vers un programme de Boot.                                                                        |
| 3        | 8      | Nom du programme qui a formaté le disque.                                                                       |
| 11 (0Bh) | 2      | Nombre d'octets par secteur. (512 sur MSX)                                                                      |
| 13 (0Dh) | 1      | Nombre de secteurs par Cluster. (1, 2, 4, 8, 16, 32, 64 ou 128)                                                 |
| 14 (0Eh) | 2      | Nombre de secteurs pour la zone réservée. Cette zone inclue le secteur de Boot. (1 par défaut pour la FAT12/16) |
| 16 (10h) | 1      | Nombre de FAT sur le disque. (2 par défaut)                                                                     |
| 17 (11h) | 2      | Nombre de fichier / dossier maximum dans le dossier racine. (0 si FAT32)                                        |
| 19 (13h) | 2      | Nombre total de secteurs 16 bits (0 si FAT32).                                                                  |
| 21 (15h) | 1      | Type de disque. (0F8h pour les disques durs, 0F0h pour les disquettes)                                          |
| 22 (16h) | 2      | Taille d'une FAT en secteurs. (0 si FAT32)                                                                      |
| 24 (18h) | 2      | Nombre de secteurs par piste.                                                                                   |
| 26 (1Ah) | 2      | Nombre de têtes.                                                                                                |
| 28 (1Ch) | 4      | Secteurs cachés. (0 par défaut si le disque n'est pas partitionné)                                              |
| 32 (20h) | 4      | Nombre total de secteurs. (Contient une valeur si le nombre total de secteurs 16 bits est égal à 0)             |
| 36 (24h) | 1      | Identifiant du disque. (à partir de 00h pour les disques amovibles et à partir de 080h pour les disques fixes)  |
| 37 (25h) | 1      | Réservé.                                                                                                        |
| 38 (26h) | 1      | Signature (029h par défaut).                                                                                    |
| 39 (27h) | 4      | Numéro de série du disque.                                                                                      |
| 43 (2Bh) | 11     | Nom du disque sur 11 caractères.                                                                                |
| 54 (36h) | 8      | Type de système de fichiers (FAT, FAT12, FAT16).                                                                |

Le reste est décrit dans le [paragraphe suivant](#).

Après la zone réservée, vient la FAT qui est disposée comme suit.

| FAT12     | FAT16       | Description                                            |
|-----------|-------------|--------------------------------------------------------|
| 000h      | 0000h       | Nombre de Cluster inutilisés.                          |
| 001h      | 0001h       | Nombre de Cluster réservés.                            |
| 002h~FEFh | 0002h~FFEFh | Pointeur vers le cluster suivant du dossier / fichier. |
| FF0h~FF6h | FFF0h~FFF6h | Réservé.                                               |
| FF7h      | FFF7h       | Cluster erroné.                                        |
| FF8h~FFFh | FFF8h~FFFFh | Dernier cluster d'un dossier / fichier.                |



### 13.3 *Comment démarrer automatiquement un logiciel sur disque.*

Dans le [chapitre sur les Slot](#) (page 118), il y a une description de ce qui se passe lorsque l'ordinateur MSX est allumé ou réinitialisé. Au cours de la recherche des ROM exécutables, si le système trouve une Disk-ROM, le système sera reconfiguré pour gérer les disques. Juste après, le secteur de Boot du premier disque installé, c'est à dire le secteur 0, est chargé à l'adresse indiquée par la variable système DIRBUF (0F351h), ensuite les 256 premiers octets de ce secteur sont copiés à 0C000h, puis un appel est effectué à l'adresse 0C01Eh. Lors de cet appel, les registres contiennent ce qui suit.

A = REBOOT (0F340h)

F = Le bit CF est à 1 si une erreur s'est produite.

DE = SETROM (0F368h)

HL = ERRADR (0F323h)

Par défaut, selon que la disquette a été formatée pour MSX-DOS1 ou MSX-DOS2, cette routine de Boot installera en RAM le MSXDOS . SYS ou le MSXDOS2 . SYS qui chargera le COMMAND . COM ou le COMMAND2 . COM à l'adresse 0100h. Ce dernier chargera et exécutera automatiquement l'AUTOEXEC . BAT si il est présent sur le disque.

Cette routine de Boot commence par « RET NC », ce qui permet de ne pas être exécutée en cas d'erreur (« Read Error » par exemple) ou lorsque le premier octet du secteur n'est pas un code correspondant au type de disque géré par une des interfaces installées (0FBh ou 0E9h en général).

Vous pouvez modifier la routine de Boot afin de charger et exécuter autre chose. Par exemple, lire des secteurs sur le disque et exécuter ce que vous y avez mis.

L'instruction CALL SYSTEM du BASIC ne charge pas le secteur de Boot et sous MSX-DOS2, c'est le fichier REBOOT . BAT qui est exécuté à la place de l'AUTOEXEC . BAT.

### 13.4 Carte de la mémoire principale sous MSX-DOS

La mémoire principale sous MSX-DOS est répartie de la façon suivante.

|                                         |                                                                              |
|-----------------------------------------|------------------------------------------------------------------------------|
| Variables et zone de travail du système | FFFFh                                                                        |
| Zone de communication fixe des disques  | F380h                                                                        |
| Zone de travail dynamique des disques   | F1C9h                                                                        |
| Cache du secteur à lire/écrire          |                                                                              |
| E/S du secteur                          |                                                                              |
| Cache des fichiers                      |                                                                              |
| Cache des FAT                           |                                                                              |
| MSX-DOS.SYS                             | ← Adresse indiquée à 0006h pendant l'exécution d'une commande<br>Registre SP |
| Pile                                    |                                                                              |
| Zone libre (TPA)                        |                                                                              |
| Base du MSX-DOS                         | 0100h                                                                        |
|                                         | 0000h                                                                        |

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.

Lorsqu'on exécute un fichier de commande, il est chargé dans la TPA à l'adresse 0100h puis exécuté à la même adresse.

La zone du cache de secteur prendra la taille du plus grand secteur présent sur un des disques installés.

### 13.5 Base du MSX-DOS 1 et 2 (System Scratch Area)

Cette zone en mémoire, qui s'étend de 0 à FFh, se met en place lorsque le système démarre et s'actualise lorsqu'on lance un fichier de commande du MSX-DOS. Cette zone contient des données utilisées pour communiquer avec le programme utilisateur par l'intermédiaire de paramètres entrées par la ligne de commande. Elle contient aussi des adresses de saut pour les appels système ou dans le programme de l'utilisateur. Voici cette zone en détail.

| Adresse | Nom    | Longueur | Fonction                                                                                                                                                                                                                                                                     |
|---------|--------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0000h   |        | 3        | Saut vers la routine de réinitialisation du MSX-DOS (WBOOT). Permet de quitter votre programme                                                                                                                                                                               |
| 0003h   | IOBYTE | 1        | IOByte. (À prendre en compte que pour les programmes pour MSX-DOS compatibles CP/M. Inutilisé par le MSX-DOS2.)                                                                                                                                                              |
| 0004h   |        | 1        | Lecteur / Utilisateur par défaut. (À prendre en compte que pour les programmes pour MSX-DOS compatibles CP/M. Inutilisé par le MSX-DOS2.)                                                                                                                                    |
| 0005h   | BDOS   | 3        | Appel à une routine BDOS qui se trouve au tout début du MSX-DOS.SYS résident en mémoire juste après la zone réservée à l'utilisateur pour la commande (TPA). Par conséquent, l'adresse de fin de la TPA sera indiquée par le contenu de l'adresse 0006h-1 (voir ci-dessous). |
| 0006h   |        | 2        | Adresse du premier octet derrière la TPA. 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 qui précède celle indiquée ici moins la taille de la pile.                                     |
| 0008h   | RST08  | 4        | Appel à une routine créée par l'utilisateur. (RST 8)                                                                                                                                                                                                                         |
| 000Ch   | RDSLT  | 4        | Appel à la routine RDSLT (Main-ROM) qui sert à lire un octet dans un Slot.                                                                                                                                                                                                   |
| 0010h   | RST10  | 4        | Appel à une routine créée par l'utilisateur. (RST 10)                                                                                                                                                                                                                        |
| 0014h   | WRSLT  |          | Appel à la routine WRSLT (Main-ROM) qui sert à écrire un octet dans un Slot.                                                                                                                                                                                                 |
| 0018h   | RST18  | 4        | Appel à une routine créée par l'utilisateur. (RST 18)                                                                                                                                                                                                                        |
| 001Ch   | CALSLT | 4        | Même routine que CALSLT (Main-ROM) qui sert à faire un appel inter-Slot.                                                                                                                                                                                                     |
| 0020h   | RST20  | 4        | Appel à une routine créée par l'utilisateur. (RST 20)                                                                                                                                                                                                                        |
| 0024h   | ENASLT | 4        | Appel à la routine ENASLT (Main-ROM) qui sert à sélectionner un Slot.                                                                                                                                                                                                        |
| 0028h   | RST28  | 4        | Appel à une routine créée par l'utilisateur. (RST 28)                                                                                                                                                                                                                        |
| 002Ch   | -      | 4        | Réservé.                                                                                                                                                                                                                                                                     |
| 0030h   | CALLF  | 4        | Appel à la routine CALLF (Main-ROM) qui sert à faire un appel inter-Slot via RST 30.                                                                                                                                                                                         |
| 0038h   |        | 3        | Routine d'interruption.                                                                                                                                                                                                                                                      |
| 003Bh   |        | 33       | Routine de changement de Slot utilisée par le système.                                                                                                                                                                                                                       |

|       |         |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------|---------|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 005Ch | FCB1    | 16  | <p>Cette zone contient une copie des 16 premiers octets du FCB correspondant au fichier dont le nom est indiqué en premier dans les paramètres de la commande entrée. Le premier octet indique le numéro de lecteur, les 11 suivants le nom du fichier, etc.</p> <p>Attention, si vous entrez une commande dont le premier paramètre n'est pas de nom de fichier, le MSX-DOS 1 placera à FCB1+1 la valeur du premier paramètre alors que le MSX-DOS 2 y placera que des espaces.</p>                                                                               |
| 006Ch | FCB2    | 16  | <p>Cette zone contient une copie des 16 premiers octets du FCB correspondant au fichier dont le nom est indiqué en second dans les paramètres de la commande entrée. Le premier octet indique le numéro de lecteur, les 11 suivants le nom du fichier, etc.</p> <p>Attention, si vous entrez une commande dont les options se trouvent avant le nom de fichier, le MSX-DOS 1 placera à FCB2+1 la valeur du second paramètre si il s'agit d'une option ou, le nom de fichier si il s'agit d'un nom de fichier alors que le MSX-DOS 2 y placera que des espaces.</p> |
| 007Ch |         | 4   | Contient des zéros.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 0080h | DMA/DTA | 127 | <p>Zone DMA (Disk Memory Access). Cette zone contient tous paramètres spécifiés derrière la commande entrée. Le premier octet indique le nombre de caractères utilisés pour les paramètres. L'octet en fin de chaine est 00h souvent précédé de 0dh.</p> <p>Lors de l'utilisation des fonctions BDOS, cette zone sert aussi de DTA (Disk Transfer Area) par défaut.</p>                                                                                                                                                                                            |

---

## 13.6 Variables d'environnement

Les variables d'environnement (à ne pas confondre avec les variables système) sont apparues avec le MSX-DOS 2. Celles-ci peuvent contenir un chemin dans les dossiers, nom de fichier ou une chaîne de caractères quelconque.

À partir du COMMAND2.COM v2.31, les variables d'environnement sont utilisables comme paramètre de commande ou de condition dans un fichier Batch.

La syntaxe pour définir une variable d'environnement est la suivante :

SET Variable=Chaîne

Exemple pour COMMAND2.COM v2.31 ou supérieure :

```
set chemin=A:\trash\  
copy readme.txt %chemin%  
del readme.txt
```

Cet exemple définit le chemin du dossier A : /TRASH, y copie le fichier README . TXT du disque et dossier actuel vers le A : /TRASH avant de l'effacer.

Pour afficher la liste des variables d'environnement, entrez la commande SET seule.

Liste des commandes qui servent aussi de variables d'environnement :

```
APPEND  
DATE  
ECHO  
EXPERT  
HELP  
KHELP  
PATH  
PROMPT  
REDIR  
SHELL  
TEMP  
TIME  
UPPER
```

## 13.7 Routines de la Disk-ROM v1.XX

### Table des sauts du pilote des disques

Il peut y avoir jusqu'à quatre Disk-ROM installées. Le numéro du Slot de chacune peut s'obtenir grâce au tableau de variables système DRVINF (0FB21h, 0FB23h, 0FB25h et 0FB27h). Parmi elles, il y a la Disk-ROM maitre qui est indiquée par la variable système MASTERS (0F348h). Celle-ci est celle dont la version est la plus récente.

#### 04010h (Disk-ROM v1.XX) **DSKIO** (« DiSK I/O »)

Rôle : Lire ou écrire de 1 à 255 secteur sur un disque. (Incompatible FAT16.)

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

F = Mettre l'indicateur CF à 0 pour lire ou 1 pour écrire.

B = Nombre de secteur à lire / écrire.

C = Descripteur du Média.

DE = Numéro du premier secteur.

HL = Adresse de destination en RAM.

Sortie : A= Code d'erreur (voir la note ci-dessous) si l'indicateur CF = 1

B = Nombre de secteurs lus / écrits.

Modifie : Tout.

Note : Voici les codes d'erreur possible.

0 = Disque est protégé contre l'écriture. (Write protected)

2 = 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 = Le secteur d'amorçage ne contient pas d'entête. Le disque nécessite un formatage. (Header not found)

0Ah = Il y a eu une anomalie pendant l'écriture. (Write fault / Write current error)

0Ch = Autres erreurs (Other error).

12h = Le disque n'est pas formaté pour le DOS. (Not a DOS disk) (MSX-DOS2)

14h = Le disque n'est pas compatible. (Incompatible disk) (MSX-DOS2)

16h = Le disque n'est pas formaté. (Unformatted disk) (MSX-DOS2)

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      ; Code pour fichier binaire
        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,0FDh ;
        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 90
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 v1.XX)    **DSKCHG** (« DiSK CHanGe »)

Rôle :     Vérifie si le disque a été changé.

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

          B = 0.

          C = Descripteur du Média.

          HL = Adresse du FCB.

Sortie :    A = Code d'erreur (voir la routine DSKIO ci-dessus) si l'indicateur CF = 1

          B = 1 (le disque est le même), 255 (disque différent) ou 0 (autre).

Modifie :   AF, BC, DE, HL, IX et IY.

Notes :    - Si le disque a été changé (B=255) ou probablement modifié (B=0), lire le secteur d'amorçage (Boot Sector) ou le premier octet du secteur de la FAT et transférer un nouveau DPB avec GETDPB comme décrit ci-dessous.  
          - Le DOS2 ne tient pas compte des changements du FCB.



#### 04016h (Disk-ROM v1.XX) **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 = Descripteur du Média.

HL = Adresse du DPB à obtenir.

Sortie : A = Code d'erreur (voir la routine DSKIO plus haut) si l'indicateur CF = 1

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

Modifie : AF, BC, DE, HL, IX et IY.

Description du DPB obtenu :

| Adresse | Longueur | Nom      | Description                                                                               |
|---------|----------|----------|-------------------------------------------------------------------------------------------|
| HL+0    | 1        | MEDIA    | Type de Média (0F8h~0FFh)                                                                 |
| HL+1    | 2        | SECSIZ   | Taille des secteurs (doit être 2^n)                                                       |
| HL+3    | 1        | DIRMSK   | (SECSIZE/32)-1                                                                            |
| HL+4    | 1        | DIRSHFT  | Nombre de bits dans DIRMSK                                                                |
| HL+5    | 1        | CLUSMSK  | (Secteurs par cluster)-1                                                                  |
| HL+6    | 1        | CLUSSHFT | (Nombre de bits dans CLUSMSK)+1                                                           |
| HL+7    | 2        | FIRFAT   | Numéro du secteur logique de la première FAT                                              |
| HL+8    | 1        | FATCNT   | Nombre de FAT                                                                             |
| HL+10   | 1        | MAXENT   | Nombre de fichier ou dossier possible (254 maxi.)                                         |
| HL+11   | 2        | FIRREC   | Numéro du premier secteur des données.                                                    |
| HL+13   | 2        | MAXCLUS  | Nombre de clusters libres +1 (sans inclure les secteurs réservés pour la FAT et dossiers) |
| HL+15   | 1        | FATSIZ   | Nombre de secteur par FAT                                                                 |
| HL+16   | 2        | FIRDIR   | Numéro de secteur logique du début du dossier de la FAT                                   |

#### 04019h (Disk-ROM v1.XX) **CHOICE** (« CHOICE »)

Rôle : Indique l'emplacement du texte des types de format possible qui sont affichés pour formater un disque.

Entrée : Rien.

Sortie : HL = Adresse du texte des choix de formatage à afficher pour la routine DSKFMT.  
Cette zone se termine par zéro. S'il n'y a pas de choix possible (un seul type de formatage) alors HL = 0.

Modifie : AF, BC, DE, HL, IX et IY.

Notes :  
- Le texte est au format [ASCIIZ](#).  
- Routine appelée par

#### 0401Ch (Disk-ROM v1.XX) **DSKFMT** (« DiSK ForMaT »)

Rôle : Formater un disque.

Entrée : A = Type de formatage (1~8). (Sans effet sur les interfaces sans choix.)

Le numéro à mettre doit correspondre à ce qui est demandé dans le texte indiqué par la routine CHOICE (04019h) ci-dessus.

D = Numéro de lecteur physique (0 pour Premier lecteur)

HL = Adresse de la zone de travail.

BC = Longueur de la zone de travail.

Sortie : A = Code d'erreur (voir la note ci-dessous) si l'indicateur CF est à 1

Modifie : Tous les registres sauf SP.

Notes :

- Efface toutes les FAT (descripteur de média au premier octet, 0FFh au deuxième / troisième octet et reste à zéro), efface les dossiers (en le remplissant de zéros) puis, écrit un secteur d'amorçage pour l'MSX au secteur 0.
- Cette routine est toujours présente dans les ROM de lecteur de disquette. Les autres interfaces utilisent en général un logiciel externe pour formater leurs disques.
- Codes d'erreurs possibles :
  - 0 = Le disque est protégé contre l'écriture. (Write protected)
  - 2 = Le disque n'est pas inséré ou pas prêt. (Disk offline/Not ready)
  - 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/Record not found)
  - 10 = Il y a une anomalie pendant l'écriture. (Write fault / Write current error)
  - 12 = Un mauvais paramètre a été spécifié (Bad parameter)
  - 14 = La mémoire sur le disque est insuffisante (Insufficient memory)
  - 16 = Autres erreurs (Other error)

#### 0401Fh (Disk-ROM v1.XX) **DSKSTP** (« Disks STop »)

Rôle : Stopper le moteur des disques de l'interface correspondante.

Entrée : Rien.

Sortie : Rien.

Modifie : AF, BC, DE, HL, IX et IY. Note : Cette routine n'existe que si l'interface gère les disques amovibles. Dans le cas contraire, 0401Fh contiendra l'octet 00h (ou parfois C9h).

## 04022h (Disk-ROM v1.XX) **STBAS** (« cold STart BASic »)

Rôle : Démarrer l'environnement du Basic et exécuter un programme Basic à partir d'un programme en langage machine.

Entrée : Mettez la variable REBOOT (0F340h) à 0 pour exécuter le fichier AUTOEXEC.BAS (à la racine du disque). Sinon, il n'y aura qu'un retour au Basic.

Sous MSX-DOS, lorsque la variable REBOOT (0F340h) contient une valeur différente de 0, il est possible d'exécuter un programme Basic dont le nom du fichier et sa longueur est spécifié dans DTA (0080h).

Sortie : Rien. (Ne revient pas.)

Exemple en assembleur :

```
; Quitter votre programme MSX-DOS et lancer un
; programme BASIC sauvegardé sous le nom MAME.BAS

CALSLT equ 001Ch
DTA     equ 0080h
STBAS   equ 04022h
REBOOT  equ 0F340h
DRVINF  equ 0FB21h

        org 0100h

        ld a,1          ; 1 pour ignorer un
        ld (REBOOT),a    ; éventuel AUTOEXEC.BAS

        ld hl,NAME
        ld de,DTA
        ld bc,END-NAME
        ldir              ; Copie le nom du fichier

        ld ix,STBAS
        ld iy,(DRVINF)    ; Slot de la Disk ROM maitre
        jp CALSLT

NAME:
        db 9              ; Longueur
        db "NAME.BAS",0

END:
```

Second exemple en assembleur :

```
; Quitter votre programme MSX-DOS et aller sous BASIC

CALSLT equ 001Ch
DTA     equ 0080h
STBAS   equ 04022h
REBOOT  equ 0F340h
TEMPST  equ 0F67Ah
DRVINF  equ 0FB21h

        org 0100h

        ld a,1          ; 1 pour ignorer un
        ld (REBOOT),a    ; éventuel AUTOEXEC.BAS
```

```

ld    a,0
ld    (DTA),a          ; Pas de nom de fichier

ld    ix,(TEMPST)      ; Efface 3 octets
ld    (ix),0           ; de la
ld    (ix+1),0         ; zone réservée
ld    (ix+2),0         ; au programme BASIC

ld    ix,STBAS
ld    iy,(DRVINF)      ; Slot de la Disk ROM maitre
jp    CALSLT

```

Troisième exemple en assembleur :

```

; Quitter votre programme MSX-DOS et lancer
; l'AUTOEXEC.BAS sous BASIC.

```

```

CALSLT equ 001Ch
STBAS  equ 04022h
REBOOT equ 0F340h
TEMPST equ 0F67Ah
DRVINF equ 0FB21h

org 0100h

ld    a,0              ; 0 pour exécuter le
ld    (REBOOT),a       ; fichier AUTOEXEC.BAS

ld    ix,(TEMPST)      ; Efface 3 octets
ld    (ix),0           ; de la
ld    (ix+1),0         ; zone réservée
ld    (ix+2),0         ; au programme BASIC

ld    ix,STBAS
ld    iy,(DRVINF)      ; Slot de la Disk ROM maitre
jp    CALSLT

```

Quatrième exemple en assembleur :

```

; Quitter votre programme MSX-DOS et lancer un programme
; BASIC en mémoire.

```

```

CALSLT equ 001Ch
DTA     equ 0080h
STBAS  equ 04022h
REBOOT equ 0F340h
TXTTAB equ 0F676h
DRVINF equ 0FB21h

org 0100h

ld    a,1              ; 1 pour ignorer un
ld    (REBOOT),a       ; éventuel AUTOEXEC.BAS

ld    a,0
ld    (DTA),a          ; pas de nom

ld    hl,(TXTTAB)

```

```

        dec    hl
        ld     (hl),0
        ex     de,hl
        ld     hl,PRGBAS
        ld     bc,END-PRGBAS
        ldir                                ; Place le programme BASIC

        ld     ix,STBAS
        ld     iy,(DRVINF)                ; Slot de la Disk ROM maitre
        jp     CALSLT

PRGBAS:
        .phase 8000h                      ; Décale l'adresse des Labels
        db     0

LINE10:
        dw     LINE20                    ; Adresse de la ligne suivante
        dw     10                        ; Numéro de la ligne actuelle
        db     91h                       ; Code pour PRINT
        db     22h,"Hello",22h          ; 22h = code des guillemets
        db     0                         ; Fin de ligne

LINE20:
        dw     BASEND                    ; Adresse de la ligne suivante
        dw     20                        ; Numéro de la ligne actuelle
        db     91h                       ; Code pour PRINT
        db     22h,"World!",22h

BASEND:
        db     0,0,0                     ; Fin du programme
        .dephase                          ; Retour à l'adresse en cours

END:

```

#### 04025h (Disk-ROM v1.XX) **FORMTM** (« interactif Disk ForMaT »)

Rôle : Formater un disque sous BASIC en demandant les paramètres à l'utilisateur.

Cette routine met l'indicateur Carry à 1 puis appelle la routine FORMTK (04026h) ci-dessous.

Entrée : Rien.

Sortie : Rien.

Modifie : Tout.

#### 04026h (Disk-ROM v1.XX) **FORMTK** (« FORMaT with Keyboard choice »)

Rôle : Formater un disque sous MSX-DOS avec les paramètres spécifiés par l'utilisateur.

Entrée : F = Mettre l'indicateur CF à 0.

HL = Début de la zone de travail.

BC = taille de la zone de travail.

Sortie : Rien.

Modifie : Tout.

#### 04029h (Disk-ROM v1.XX) **MTOFF** (« MoTor OFF »)

Rôle : Stopper le moteur de tous les disques du système.

Entrée : Rien.

Sortie : Rien.

Modifie : AF, BC, DE, HL, IX et IY.

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.

```
RDSLIT      equ    000ch
CALSLT      equ    001ch
MTOFF       equ    4029h
MASTERS     equ    0f348h

DiskOFF:
            ld      a, (MASTERS)
            ld      hl, MTOFF
            call    RDSLIT
            and     a                ; Routine MTOFF présente?
            ret     z                ; Retour si pas de MTOFF
            ld      iy, (MASTERS-1)
            ld      ix, MTOFF
            jp      CALSLT
```

#### 0402Dh (Disk-ROM v1.XX) **GTSLT1** (« GET SLoT page 1 »)

Rôle : Donne le numéro de Slot sélectionné sur la plage 04000h~07FFFh.

Entrée : Rien.

Sortie : A = Numéro de Slot.

Modifie : Tout.

Note:

### Routines de base

#### 04030h (Disk-ROM v1.XX) **GTDSHM** (« GeT DoS Hi-Memory »)

Rôle : Donne l'adresse de la fin du noyau du MSX-DOS.

Entrée : Rien.

Sortie : HL = Adresse contenu dans DOSHIM (0F34BH).

Modifie : HL.

Note: L'adresse de début du noyau est stockée à l'adresse 0006h lorsqu'une commande est chargée. Le fichier MSX-DOS.SYS est chargé dans cette zone.

04034h (Disk-ROM v1.XX)    **KEYINP** (« Key INPut »)

Rôle :     Tester si CTRL+STOP est pressé sinon, si une touche est pressée.

Entrée :   Rien.

Sortie :    A = Code de la touche ou 3 si CTRL+STOP est pressé. 0F336h et 0F33h sont actualisés.

Modifie : Tout.

04078h (Disk-ROM v1.XX)

Rôle :     Tester si une touche est pressée.

Entrée :   Rien.

Sortie :    A = Code de la dernière touche pressée si 0F336h est différent de 0.  
              0F336h est mis à 0FFh si une touche est pressée.

Modifie : Tout.

0408Fh (Disk-ROM v1.XX)

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

Entrée :    A = Code du caractère.

              CSRX (0F3DDh) = Abscisse du curseur.

              CSRY (0F3DCh) = Ordonnée du curseur.

Sortie :    CSRX et CSRY sont actualisés.

Modifie : Rien.

Notes :    - Appel au Hook H.CHPU (0FDA4h).

              - Cette routine prend en compte les codes de contrôle et d'échappement.

              - Cette routine ne fait qu'appeler la routine CHPUT (00A2h) de la Main-ROM.

#### 0409Bh (Disk-ROM v1.XX)

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

Entrée : A = Code d'un caractère ou 1 pour annoncer un caractère graphique étendu.

Sortie : GRPHED (0FCA6h) est mis à 1 lorsque le code entrée est 1.

F = Si l'indicateur CF est à 0, c'est que le code entrée est 1.

Les indicateurs CF et ZF sont à 1 lorsque que le code est un caractère étendu.

L'indicateur CF est à 1 et ZF à 0 lorsque que le code un caractère ordinaire.

Modifie : AF.

Notes : - GRPHED (0FCA6h) est pris en compte pour indiquer si il s'agit d'un caractère graphique étendu (compris entre 65 et 95) ou pas.

- Voir les jeux de caractères MSX pour connaître les caractères graphiques correspondants.

- Cette routine ne fait qu'appeler la routine CHGET (00A5h) de la Main-ROM.

#### 040A7h (Disk-ROM v1.XX)

Rôle : Retour au mode direct du Basic. (Identique à la fonction BDOS 0)

Entrée : Rien.

Sortie : Rien.

Modifie : Tout.

Note: Cette routine ne fait qu'appeler la routine READYR (0409Bh) de la Main-ROM.

#### 040ABh (Disk-ROM v1.XX)

Rôle : Appel inter-Slot à l'adresse indiquée par le registre IX.

Entrée : Rien.

Sortie : Dépend de la fonction appelée.

Modifie : Dépend de la fonction appelée.

#### 040B8h (Disk-ROM v1.XX)

Rôle : Indiquer la présence de la RTC. Si elle est présente, le format de l'heure est réglé à 24 et l'alarme est désactivée.

Entrée : Rien.

Sortie : CLCKIC (0F338h) = 0FFh si l'horloge est présente sinon, 00h.

Modifie : AF, BC.



#### 04115h (Disk-ROM v1.XX)

Rôle : Régler la date de la RTC.

Entrée : .

Sortie : .

Modifie : AF, BC, DE.

#### 04130h (Disk-ROM v1.XX)

Rôle : Régler la l'heure de la RTC.

Entrée : Rien.

Sortie : Rien.

Modifie : .

#### 04142 (Disk-ROM v1.XX)

Rôle : Sauvegarder les registres E, D, C et B dans la RTC.

Entrée : H = Donnée.

E = Nombre de grignotage.

Sortie : .

Modifie : AF, HL.

#### 0414Eh (Disk-ROM v1.XX)

Rôle : Activer le timer et sélectionner le bloc 0 de registres de la RTC.

Entrée : Rien.

Sortie : Rien.

Modifie : Modifie AF.

#### 04159h (Disk-ROM v1.XX)

Rôle : Stopper le timer et sélectionner le bloc 0 de registres de la RTC.

Entrée : .

Sortie : .

Modifie : Modifie AF.

#### 04160h (Disk-ROM v1.XX)

Rôle : Convertir la valeur du registre H au format BCD puis l'écrire dans la RTC à partir du registre E.

Entrée : .

Sortie : .

Modifie : .

#### 04179h (Disk-ROM v1.XX)

Rôle : Lire la date et l'heure de la RTC.

Entrée : .

Sortie : .

Modifie : .

#### 041ADh (Disk-ROM v1.XX)

Rôle : Convertir les registres (E-1) et (E-2) de la RTC en nombres binaires et les mettre dans A.

Entrée : E = .

Sortie : A = D = Valeurs de .

Modifie : AF, DE.

#### 041C1h (Disk-ROM v1.XX)

Rôle : Chaîne de caractères « MSX-DOS ver. 2.2 Copyright 1984 by Microsoft ».

Note : La chaîne ne se termine pas par 00h. Il y a un espace au début et à la fin.

### **Routines des fonctions BDOS**

#### 041EFh (Disk-ROM v1.XX) **CPMVER**

Rôle : Renvoyer dans A la version CP/M compatible.

Entrée : Rien.

Sortie : A = Quatre bits de point fort indiquent la version majeurs et les quatre bits de point faible indiquent la version mineure.

B = 00h.

Modifie : A, B.

Note : Utilisée par la fonction BDOS \_CPMVER (0Ch).

#### 0436Ch (Disk-ROM v1.XX)

Rôle : Supprimer un ou plusieurs fichiers.

Entrée : .

Sortie : .

Modifie : A = 0FFh si aucun fichier n'a été effacé, 0 si au moins un fichier a été effacé.

#### 04392h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

#### 04427h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

#### 04462h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

#### 04555h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

#### 04916h (Disk-ROM v1.XX)

Rôle : Multiplier DE par BC.

Entrée : DE = Nombre entier à multiplier ;  
BC = Nombre entier multiplicateur.

Sortie : HLBC = Résultat.

Modifie : BC, HL.

#### 0576Fh (Disk-ROM v1.XX)

Rôle : Initialiser le système de disques.

Entrée : .

Sortie : .

Modifie : .

#### 05897h (Disk-ROM v1.XX)

Rôle : Sauter depuis le Hook H.RUNC.

Entrée : .

Sortie : .

Modifie : .

#### 05B1Bh (Disk-ROM v1.XX)

Rôle : Chaîne de caractères « AUTOEXECBAS ».

Note : La chaîne se termine par 00h.

#### 05B1Bh (Disk-ROM v1.XX)

Rôle : Chaîne de caractères « RUN"AUTOEXEC.BAS" ».

Note : La chaîne se termine par 00h.

#### 0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

#### 0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

#### 0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

#### 0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

#### 0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

0h (Disk-ROM v1.XX)

Rôle : .

Entrée : .

Sortie : .

Modifie : .

## 13.8 Fonctions BDOS

Les fonctions BDOS servent principalement à gérer les fichiers et les dossiers des disques dans vos programmes. Ces fonctions doivent être appelées à l'adresse 0005h sous MSX-DOS. On peut aussi les appeler à l'adresse 0F37Dh (ROMBDOS) sous Basic.

En entrée, le registre C doit contenir le numéro de la fonction à appeler, ainsi que les paramètres indiqués dans la [liste des fonctions BDOS](#) à la page 451 mais, auparavant voici quelques infos nécessaires.

### Le FCB (File Control Block)

Les fonctions BDOS du MSX-DOS1 se basent sur les données contenues dans un FCB pour manipuler les fichiers. L'utilisation du FCB diffère un peu si vous souhaitez la compatibilité avec le CP/M ou pas. Le FCB contiendra les données suivantes après l'ouverture d'un fichier.

| Adresse | Nom   | Longueur | Contenu                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------|-------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FCB     | DR    | 1        | Numéro du disque où se trouve le fichier. (0 = Disque par défaut, 1 = disque A, etc)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| FCB+1   | FNAME | 8        | Nom du fichier. Mettez des espaces (20h) pour les caractères non utilisés. Les métacaractères peuvent être présents pour certaines fonctions.                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| FCB+9   | FEXT  | 3        | Extension du nom du fichier. Mettez des espaces (20h) pour les caractères non utilisés. Les métacaractères peuvent être présents pour certaines fonctions.<br>Sous CP/M le bits 7 des octets indiquent ceci :<br>Bit 7 de FEXT+1 = 1 pour lecture seule<br>Bit 7 de FEXT+2 = 1 pour fichier du système                                                                                                                                                                                                                                                                                     |
| FCB+12  | EX    | 1        | Octet de poids faible du numéro du bloc d'enregistrements actuels pour les accès séquentiels.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| FCB+13  | S1    | 1        | Octet de poids fort du numéro du bloc d'enregistrements actuel pour les accès séquentiels.<br>Sous CP/M, seul le bit 0 est utilisé. Ce qui fait 512 blocs de 16384 octets possibles, un fichier peut donc atteindre une taille de 8Mo maximum.<br>Ou bien, attributs du fichier réservé au système.<br>Le format est le suivant :<br>Bit 0 = 1 pour fichier protégé contre l'écriture<br>Bit 1 = 1 pour fichier invisible<br>Bit 2 = 1 pour fichier du système<br>Bit 3 = 1 pour nom du volume (MSX-DOS2)<br>Bit 4 = 1 pour dossier (MSX-DOS2)<br>Bit 5 = 1 pour archive<br>Bit 6 et 7 = 0 |



|        |        |        |                                                                                                                                                                                      |
|--------|--------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FCB+14 | S2     | 1      | 8 bits de poids faible de la taille des enregistrements. Taille des enregistrements sous CP/M (128 octets par défaut pour être compatible CP/M) ou, numéro d'extension.              |
| FCB+15 | RC     | 1      | 8 bits de poids fort de la taille des enregistrements, ou numéro de l'enregistrement. Toujours 0 sous CP/M.                                                                          |
| FCB+16 | FILSIZ | 4      | Taille du fichier (en octet).                                                                                                                                                        |
| FCB+20 | DATE   | 2      | Date de la dernière modification du fichier. Le format est le suivant :<br>FCB+20 = MMMJJJJJ, FCB+21 = AAAAAAM<br>A = Année (1980~2079), M = Mois et J = Jour.                       |
| FCB+22 | TIME   | 2      | Heure de la dernière modification du fichier. Le format est le suivant :<br>FCB+22 = MMMSSSSS, FCB+23 = HHHHHMMM<br>H = Heure, M = Minute et S = Seconde.                            |
| FCB+24 | DEVID  | 1      | Numéro d'identifiant du périphérique.<br>- FBh = PRN (Imprimante)<br>- FCh = LST (Liste)<br>- FDh = NUL (Aucun)<br>- FEh = AUX (Auxiliaire)<br>- FDh = CON (Clavier)                 |
| FCB+25 | DIRLOC | 1      | Position du fichier/dossier dans la racine des dossiers.                                                                                                                             |
| FCB+26 | STRCLS | 2      | Numéro du premier Cluster du fichier. Indiqué en bits inversés.                                                                                                                      |
| FCB+28 | CLRCLS | 2      | Numéro du dernier Cluster accédé. Indiqué en bits inversés.                                                                                                                          |
| FCB+30 | CLSOFF | 2      | Nombre de Cluster entre le premier et le dernier Cluster du fichier accédé. Indiqué en bits inversés.                                                                                |
| FCB+32 | CR     | 1      | Numéro (0~127) de l'enregistrement actuel pour les accès séquentiels. (Voir les fonctions 14h et 15h.)                                                                               |
| FCB+33 | RN     | 3 ou 4 | Numéro de l'enregistrement actuel pour les accès directs. Le quatrième octet est utilisé si la taille des enregistrements est inférieure à 64. (Voir les fonctions 21h, 22h et 23h.) |

#### Notes:

- Pour ouvrir un fichier vous devez spécifier DR, FNAME, FEXT et aussi EX pour une compatibilité CP/M. EX est généralement mis à 0. Le reste doit être mis à 0 au moins jusqu'à RC.
- Le numéro du bloc d'enregistrements est généralement appelé numéro de l'étendue d'enregistrements par les utilisateurs du CP/M.
- Les données de FCB+16 jusqu'à FCB+31 ne doivent pas être modifiées. Elles sont là seulement pour être consultées.
- Seuls les 33 premiers octets sont utilisés lors d'accès séquentiels.
- L'adresse de FCB peut être connue avec l'instruction Basic VARPTR().

## Les métacaractères (Wildcard)

Les métacaractères sont des caractères spéciaux utilisés pour filtrer les noms de fichier (et de dossier sous MSX-DOS2). Ils sont utiles pour rechercher un ou des noms de fichier ou de dossier en omettant une partie. Les métacaractères utilisables sont le point d'interrogation (?) qui permet d'omettre un caractère et, l'astérisque (\*) qui permet d'omettre la fin du nom.

## Le gestionnaire de fichier (File Handle)

À partir du MSX-DOS2, l'accès à un fichier ou à certains périphériques se fait via un gestionnaire de fichier (File Handle). Certains périphériques doivent être manipulés en utilisant un des numéros suivants.

| Descripteur de fichier | Périphérique connecté          |
|------------------------|--------------------------------|
| 0                      | Entrée de la console (clavier) |
| 1                      | Sortie de la console (écran)   |
| 2                      | Entrée auxiliaire              |
| 3                      | Sortie auxiliaire              |
| 4                      | Sortie de l'imprimante         |

Ces numéros, appelés « descripteur de fichier » sont attribués à l'initialisation. Ensuite, le MSX-DOS2 alloue automatiquement un numéro supérieur aux fichiers que l'on ouvre.

## Le FIB (File Information Block)

Autre nouveauté dans le MSX-DOS 2, c'est le bloc d'informations de fichier (FIB) qui remplace le FCB. Le FIB a une taille de 64 octets. La description de ce bloc est la suivante.

| Adresse | Longueur |                                                            |
|---------|----------|------------------------------------------------------------|
| FIB+0   | 1        | Doit contenir 0FFh afin de distinguer un FIB d'un chemin   |
| FIB+1   | 13       | Nom du fichier ( <a href="#">ASCII</a> ).                  |
| FIB+14  | 1        | Attributs du fichier (Voir ci-dessous pour la description) |
| FIB+15  | 2        | Heure de la dernière mise à jour                           |
| FIB+17  | 2        | Date de dernière mise à jour                               |
| FIB+19  | 2        | Numéro du premier Cluster                                  |
| FIB+21  | 4        | Taille du fichier                                          |
| FIB+25  | 1        | Disque logique                                             |
| FIB+26  | 38       | Informations internes (ne doivent pas être modifiées)      |

## Attributs de fichier

Les attributs d'un fichier sont sur un octet dont les bits définissent le type d'élément (fichier, dossier, etc). Le format de cet octet est le suivant.

Bit 0 = 1 pour fichier protégé en écriture

Bit 1 = 1 pour fichier caché

Bit 2 = 1 pour fichier système

Bit 3 = 1 pour nom de volume

Bit 4 = 1 pour dossier

Bit 5 = 1 pour archive

Bit 6 = Réserve (mettre 0)

Bit 7 = 1 pour indiquer que le FIB fait référence à un périphérique au lieu d'un élément. Tous les autres bits d'attribut sont ignorés dans ce cas.

## Liste des fonctions BDOS du MSX-DOS 1 et 2

Voici la listes de toutes les fonctions disponibles pour MSX-DOS 1 et 2. Celles qui sont spécifiques au MSX-DOS 2 commence à partir de la [fonction 40h](#).

### Fonction 00h - Program Terminate (\_TERM0)

Rôle : Quitter le programme du fichier de commande chargé.

Sous MSX-DOS 1 ou CP/M, terminer la gestion d'une erreur avant de quitter un programme. Cette fonction à le même effet qu'un saut à l'adresse 0000h (RST 0).

Sous MSX-DOS 2, vous pouvez utiliser directement la fonction 62h (\_TERM) à la place.

Entrée : Rien

Sortie : Rien

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2.

## Fonction 01h - Console Input (\_CONIN)

Rôle : Attendre l'entrée d'un caractère au clavier puis l'afficher à l'écran.

Entrée : Rien

Sortie : A = L = Caractère lu dans le cache du clavier.

Si le cache du clavier n'est pas vide, le dernier caractère entré sera pris en compte. La fonction gère les [codes de contrôle et d'échappement](#) (page 552) en plus des trois suivants.

CTRL+C termine le programme du fichier de commande chargé.

CTRL+P active l'envoi du caractère entrée aussi vers l'imprimante (écho).

CTRL+N stoppe l'envoi du caractère vers l'imprimante.

Notes :

- Lorsque l'envoi du caractère vers l'imprimante est activée et que l'imprimante n'est pas opérationnelle, la fonction attendra sans que l'on puisse l'interrompre jusqu'à ce que l'imprimante reçoive le caractère.
- Les 32 caractères spéciaux précédés du code 01h ne sont pas utilisables.

Modifie : Tout sauf C, D et E.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

## Fonction 02h - Console Output (\_CONOUT)

Rôle : Afficher un caractère à l'écran. L'affichage se fait au curseur. Si l'écho vers l'imprimante est activé, le caractère est également imprimé.

Entrée : E = Caractère à envoyer

Cette fonction reconnaît les [codes de contrôle](#) (page 552) et aussi CTRL+C, CTRL+P, CTRL+N. Les tabulations correspondent à chaque huitième colonne.

Une vérification de l'état de l'entrée de la console est effectuée et si l'un des caractères de contrôle spéciaux décrits pour la fonction 0Bh (Console Status) est trouvée, il sera traité comme pour cette fonction. Tout autre caractère sera sauvegardé en interne pour un appel de fonction 01h (Console Input) au dessus.

Sortie : Rien

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

### **Fonction 03h - Auxiliary Input (\_AUXIN)**

Rôle : Lire le caractère envoyé par le périphérique auxiliaire et si aucun caractère n'est présent, il en attendra un.

Entrée : Rien

Sortie : A = L = Caractère lu.

Si aucun périphérique auxiliaire n'est installé, ces registres contiendront toujours 1Ah, ce qui est le code de fin de fichier (CTRL+Z).

Note : Cette fonction est généralement utilisé par l'interface RS-232-C.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

### **Fonction 04h - Auxiliary Output (\_AUXOUT)**

Rôle : Envoyer un caractère vers le périphérique auxiliaire. Le périphérique auxiliaire doit être installé avant de pouvoir utiliser cette fonction sinon, cette fonction n'aura aucun effet.

Entrée : E = Caractère à envoyer.

Sortie : Rien.

Note : Cette fonction est généralement utilisé par l'interface RS-232-C.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

### **Fonction 05h - Printer Output (\_LSTOUT)**

Rôle : Envoyer un caractère vers l'imprimante.

Entrée : E = Caractère à imprimer

Le code de tabulation n'est pas pris en compte, bien que ce soit le cas lorsque la sortie d'écran est renvoyée vers l'imprimante avec CTRL+P.

Sortie : Rien

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

## **Fonction 06h - Direct Console I/O (\_DIRIO)**

Rôle : Lire le dernier caractère entré dans le cache du clavier, ou bien envoyer un caractère à l'écran..

Entrée : E = FFh pour lire le dernier caractère entré au clavier ou caractère à afficher à l'écran.

Mettez E à FFh pour lire le clavier, autrement . Les codes de contrôle sont ignorés ainsi que CTRL+C, CTRL+P, CTRL+N et CTRL+S.

Si E contient un caractère, il sera affiché à l'écran. La tabulation et l'écho vers l'imprimante sont ignorés. Les codes de contrôle ainsi que CTRL+C, CTRL+P, CTRL+N et CTRL+S sont aussi ignorés.

Sortie : A = L = Caractère entré au clavier sinon, 00h

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

## **Fonction 07h - Direct Console Input (\_DIRIN)**

Rôle : Cette fonction a le même effet que la fonction 06h, sauf qu'elle attendra jusqu'à ce qu'un caractère soit envoyé au clavier. Cette fonction n'est pas compatible avec CP/M qui utilise ce numéro pour "obtenir l'octet d'entrée / sortie".

Entrée : Rien

Sortie : A = L = Caractère entré au clavier

Compatibilité : MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

## **Fonction 08h - Console Input Without Echo (\_INNOE)**

Rôle : Attendre l'entrée d'un caractère au clavier puis l'afficher à l'écran. Cette fonction est identique à la fonction 01h sauf que le caractère saisi ne pourra pas être envoyé à l'imprimante (sans écho possible). Cette fonction n'est pas compatible avec le CP/M qui utilise ce numéro pour "définir l'octet d'Entrée / Sortie".

Entrée : Rien

Sortie : A = L = Caractère entré

Compatibilité : MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

## Fonction 09h - String Output (\_STROUT)

Rôle : Afficher une chaîne de caractères (code ASCII) à l'écran.

Entrée : DE = Adresse du premier caractère de la chaîne.

La chaîne doit se terminer par le caractère « \$ » (24h).

Sous Nextor : Lorsque le mode rapide est activé, la longueur maximale de la chaîne à afficher est de 511 caractères. Le mode rapide est désactivé par défaut. Il doit être explicitement. Utilisez la fonction 71h (\_FOUT) pour activer le mode rapide.

Sortie : Rien.

Note : Appelle la fonction 02h (Console Output)

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

## Fonction 0Ah - Buffered Line Input (\_BUFIN)

Rôle : Entrée d'une ligne entrée au clavier en mémoire cache.

Entrée : DE = Adresse du cache.

Le premier octet de ce cache doit contenir le nombre de caractères qu'il peut contenir (0 ~ 255). Le code CR sera placé en fin de ligne. Si le nombre de caractères entrés dépasse la longueur du cache spécifiée, le code CR sera placé à l'adresse du cache + (DE) + 1.

Lors de la saisie à partir du clavier (ce qui sera normalement le cas), un éditeur de ligne simple est fourni, ainsi qu'un tampon circulaire de 256 octets des lignes précédentes pouvant être édité et ré-entrée. Les détails de ces fonctions d'édition sont décrits dans le document séparé " Command Specification ", ils ne sont donc pas inclus ici. Lorsque la mémoire tampon d'entrée est pleine, la sonnerie de la console retentit pour chaque caractère saisi qui ne peut pas être placé dans la mémoire tampon. Chaque caractère saisi sera répercuté sur la sortie standard et également sur l'imprimante si l'écho d'imprimante est activé.

Sortie : Rien

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

## Fonction 0Bh - Console Status (\_CONST)

Rôle : Vérifier si une touche du clavier est pressée.

Entrée : Rien

Sortie : A = L = 0 si aucun caractère ou code n'a été entrée, sinon FFh.

Le caractère/code entré est stocké dans le cache KEYBUF (0FBF0h) à l'adresse indiquée par le contenu de la variable système GETPNT (0F3FAh) puis la variable est incrémentée. Lorsque le cache KEYBUF est plein, le caractère/code suivant continu à être stocké au début du cache et ainsi de suite.

Si CTRL+C sont pressés, le programme se termine. Si c'est CTRL+P, l'écho vers l'imprimante est activé, Il pourra être désactivé avec CTRL+N.

Lorsque A = FFh, appelez la fonction 08h (\_STROUT) avant de rappeler cette fonction sinon A restera bloqué sur FFh et les codes de contrôle seront ignorés.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

## Fonction 0Ch - Return Version Number (\_CPMVER)

Rôle : Cette fonction renvoie le numéro de version du CP/M compatible.

Entrée : Rien

Sortie : A = 22h (Ce qui correspond à la version 2.2)

B = 00h

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

## Fonction 0Dh - Disk Reset (\_DSKRST)

Rôle : Réinitialisation des disques.

Entrée : Rien

Sortie : A = 0

DE et IX = Adresse du DPB (Disk Parameter Block) du premier disque installé.

Les données en attente dans le cache secteur sont écrites sur le disque si elles s'avèrent différentes puis, le disque « A: » est sélectionné comme disque par défaut et le DTA/DMA placé à l'adresse 0080h.

Note : Il n'est pas nécessaire d'appeler cette fonction pour permettre un changement de disque, comme c'est le cas sous CP/M.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2.



## Fonction 0Eh - Select Disk (\_SELDSK)

Rôle : Sélectionner le disque à utiliser par défaut (disque actuel).

Entrée : E = Numéro du disque logique 0=A: 1=B: etc.

Sortie : F = L'indicateur CF sera mis à 1 si le disque spécifié n'est pas un disque physique.

A = L = Nombre de disques installés (1~8) sans inclure le disque en RAM même si présent.

Le disque actuel est également stocké à l'adresse 0004h pour la compatibilité CP/M.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2.

## Fonction 0Fh - Open File (\_FOPEN)

Rôle : Ouverture d'un fichier présent sur un disque.

Entrée : DE = Pointeur du FCB du fichier à ouvrir.

Le nom (FNNAME et FEXT) peut contenir des métacaractères. Dans ce cas, le premier fichier qui correspond au filtre sera ouvert.

Le nom d'un périphérique peut être placé dans le FCB (sans le deux-points) pour permettre l'accès au périphérique correspondant comme s'il s'agissait d'un fichier.

Sous MSX-DOS2, le fichier correspondant dans le dossier actuel du disque spécifié sera ouvert s'il est présent.

Sortie : A = L = 0FFh si le fichier est introuvable, 0 si le fichier est trouvé.

L'octet de poids faible du numéro de bloc d'enregistrements n'est pas modifié par cette fonction et un fichier ne sera ouvert que s'il est suffisamment volumineux pour contenir le bloc spécifié. Normalement, le programme transitoire mettra le numéro de bloc à zéro avant d'appeler cette fonction. L'octet de poids fort du numéro de bloc sera mis à zéro pour assurer la compatibilité avec CP/M.

Le nom de fichier et l'extension entrés dans le FCB seront remplacés par le nom réel du fichier ouvert. Ce sera normalement le même que ce qui existait auparavant, mais peut être différent si un nom de fichier comportait des métacaractères ou des lettres minuscules en entrée.

Le nombre d'enregistrements sera défini sur le nombre d'enregistrements de 128 octets dans le bloc spécifiée, calculée à partir de la taille du fichier. Le champ de taille de fichier lui-même, l'identifiant de volume et les 8 octets réservés seront également configurés. Les champs d'enregistrement courant et d'enregistrement pour accès direct ne seront pas modifiés par cette fonction, il appartient au programme d'application de les initialiser avant d'utiliser les fonctions de lecture ou d'écriture.

Sous MSX-DOS2, si le fichier n'est pas trouvé, la variable d'environnement "APPEND" sera examinée. Si elle est définie, elle est interprétée comme une chaîne disque/chemin spécifiant un autre dossier dans lequel rechercher le fichier. Le fichier spécifié sera recherché pour le fichier et s'il est trouvé, il sera ouvert comme indiqué ci-dessus. Dans ce cas, l'octet de disque du FCB sera défini sur le disque sur lequel le fichier a été trouvé pour assurer un accès correct au fichier si l'octet de disque d'origine était égal à zéro (valeur par défaut).

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2.

## Fonction 10h - Close File (\_FCLOSE)

Rôle : Fermer un fichier.

Entrée : DE = Pointeur du FCB du fichier ouvert.

Le FCB doit être déjà ouvert avec la fonction 10h (Open File) ou 16h (Create File). Si le fichier a seulement été lu entre temps, cette fonction ne fait rien. Si le fichier a été modifié et que des données restent en mémoire cache, elles seront écrites dans le fichier et l'entrée du dossier sera mise à jour en conséquence.

Sortie : A = L = 0FFh Si échec, 0 si réussite

Le fichier peut toujours être lu après une fermeture, de sorte que la fonction peut être considérée comme une fonction pour assurer le bon déroulement des écritures.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2.

## Fonction 11h - Search For First Entry (\_SFIRST)

Rôle : Rechercher un fichier sur le disque spécifié (dans le dossier actuel si MSX-DOS2).

Entrée : DE = Pointeur du FCB pour effectuer la recherche (d'une longueur de 16 octets).

Le nom du fichier (FNNAME et FEXT) peut contenir des métacaractères. Dans ce cas, le premier fichier trouvé sera celui qui correspond au filtre.

Le numéro de disque (DR) et le numéro de bloc d'enregistrements ne peuvent pas être remplacés par un point d'interrogation (3Fh) comme sous CP/M.

Normalement, le numéro de bloc d'enregistrements doit être mis à zéro avant d'appeler cette fonction. Si l'octet de poids faible du champ d'extension est utilisé, un fichier ne sera trouvé que s'il est suffisamment volumineux pour contenir ce numéro de bloc. Cette fonction ne peut pas trouver les fichiers système, ni les dossiers qu'il peut y avoir sous MSX-DOS 2.

Sortie : A = L = 0FFh si aucun fichier n'est trouvé, 0 si le fichier est trouvé.

La position relative de départ se calcule pas en multipliant le registre par 32 comme sous CP/M.

F = L'indicateur CF est mis à 1 lorsqu'aucun fichier n'est trouvé.

Le FCB du fichier trouvée est placé à la DTA. Il peut être utilisé directement pour ouvrir le fichier avec la fonction 0Fh (\_FOPEN). Le numéro du bloc d'enregistrements sera défini sur l'octet de poids faible du bloc à partir du FCB de recherche et le nombre d'enregistrements sera initialisé de manière appropriée comme la fonction 0Fh (\_FOPEN). L'octet d'attribut du fichier sera stocké à l'octet FCB+13 (S1), car sa position normale est utilisée pour le numéro d'étendu.

Si aucun fichier n'est trouvée (A = 0FFh), le DTA ne sera pas modifié. Cette fonction garde en mémoire suffisamment d'informations en interne pour permettre de poursuivre la recherche avec la fonction 12h (\_SNEXT) ci-dessous. Il n'est donc pas nécessaire de conserver le FCB pour utiliser la fonction 12h (\_SNEXT).

Modifie : Tous les registres.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2.

## Fonction 12h - Search For Next Entry (\_SNEXT)

Rôle : Continuer la recherche d'un fichier à la suite.

Entrée : Rien.

Le FCB d'origine utilisé par la fonction 11h (\_SFIRST) n'est pas utilisé par cette fonction.

Sortie : A = L = 0FFh si aucun fichier n'est trouvé, 0 si le fichier trouvé.

F = L'indicateur CF est mis à 1 lorsqu'aucun fichier n'est trouvé.

On aura les mêmes résultats qu'avec la fonction 11h (\_SFIRST) ci-dessus. Les informations utilisées pour poursuivre la recherche sont conservées en interne par le MSX-DOS.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2.

## Fonction 13h - Delete File (\_FDEL)

Rôle : Supprimer un ou plusieurs fichiers.

Entrée : Rien

Le nom (FNNAME et FEXT) peut contenir des métacaractères. Dans ce cas, tous les fichiers du dossier actuel du disque spécifié dans le FCB et correspondant au filtre seront effacés. Les fichiers système, les fichiers cachés, les fichiers en lecture seule et les sous-dossiers du MSX-DOS 2 ne seront pas effacés.

Sortie : A = L = 0FFh si aucun fichier n'a été effacé, 0 si au moins un fichier a été effacé.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2.

## Fonction 14h - Sequential Read (\_RDSEQ)

Rôle : Lire séquentiellement l'enregistrement d'un fichier.

Entrée : DE = Pointeur du FCB du fichier ouvert avec la fonction 0Fh (\_FOPEN).

Le numéro de l'enregistrement et du bloc d'enregistrements actuel définissent l'endroit à lire (0 par défaut). Contrairement au CP/M v2.2, il est possible d'avoir des enregistrements partiellement remplis, car la taille du fichier n'est pas nécessairement un multiple de 128 octets. Si cela se produit, l'enregistrement partiel est complété avec des zéros lorsqu'il est copié à l'adresse de transfert (DTA).

Sortie : A = L = 01H si erreur (end of file), 0 si le fichier a été lu correctement.

L'enregistrement est chargé à l'adresse de transfert (DTA) du disque actuel. Le numéro de l'enregistrement est incrémenté après avoir été lu avec succès et, s'il atteint 080h, il est remis à zéro et le numéro du bloc est incrémenté à son tour.

Note : Si la compatibilité CP/M n'est pas nécessaire, il est préférable d'utiliser la fonction 27h (\_RDBLK) qui offre bien plus d'avantages.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2.

## Fonction 15h - Sequential Write (\_WRSEQ)

Rôle : Écrire séquentiellement un enregistrement dans un fichier.

Entrée : DE = Pointeur du FCB du fichier ouvert avec la fonction 0Fh (\_FOPEN).

L'enregistrement à écrire doit être placé à l'adresse de transfert (DTA) du disque actuelle.

Sortie : A = L = 01H si erreur (disque plein), 0 si succès.

Le numéro de l'enregistrement est incrémenté après avoir été écrit avec succès et, s'il atteint 080h, il est remis à zéro et le numéro de bloc d'enregistrements est incrémenté à son tour. Le nombre d'enregistrements est mis à jour si le fichier est étendu ou si l'écriture est déplacée dans un nouveau bloc. La taille du fichier dans le FCB est également mise à jour si le fichier est étendu, ainsi que le nombre d'enregistrements si nécessaire.

Note : Si la compatibilité CP/M n'est pas nécessaire, il est préférable d'utiliser la fonction 26h (\_WRBLK) qui offre bien plus d'avantages.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2.

## Fonction 16h - Create File (\_FMAKE)

Rôle : Créer un nouveau fichier sur le disque spécifié et l'ouvre pour pouvoir y écrire des données.

Entrée : DE = Pointeur du FCB contenant les données pour créer le fichier.

Le disque, le nom de fichier et l'octet de poids faible du numéro de bloc d'enregistrements doivent être configurés dans le FCB. Les métacaractères ne peuvent pas être utilisés. Des vérifications seront effectuées pour s'assurer que le nom de fichier est valide et pas déjà créé.

Sortie : A = L = 1 si échec (disque plein), 0 si succès.

Si un fichier au même nom existe déjà, et que le numéro de bloc est zéro, le nouveau fichier sera créé par dessus l'ancien autrement, si le numéro de bloc est différent de zéro, le fichier existant sera ouvert sans créer de nouveau fichier. Cela garantit la compatibilité avec le CP/M dont chaque extension devait être créée explicitement.

Dans tous les cas, le fichier résultant sera ouvert avec le numéro de bloc requis exactement comme si un appel de fonction 0Fh (\_FOPEN) avait été effectué.

Le fichier est créé dans le dossier actuel sous MSX-DOS 2.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2.

## Fonction 17h - Rename File (\_FREN)

Rôle : Renommer un ou plusieurs fichiers.

Entrée : DE = Pointeur du FCB contenant les données pour renommer.

Ce FCB doit contenir le numéro de disque et le nom de fichier à renommer, mais ainsi que le nom de fichier de remplacement placé à FCB+17. Les métacaractères sont utilisables afin de renommer plusieurs fichier.

Sortie : A = L = 1 si échec (disque plein), 0 si succès.

Des contrôles sont effectués pour empêcher la création de noms de fichiers en double ou illégaux. Les fichiers cachés et les fichiers système ne seront pas renommés.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2.

## Fonction 18h - Get Login Vector (\_LOGIN)

Rôle : Indiquer quels disques sont installés ou pas.

Entrée : Rien

Sortie : H = 0

L = Disques installés.

Les bits 0~7 indiquent que le disque correspondant (A ~ H) est installé lorsqu'ils sont à 1.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

## Fonction 19h - Get Current Drive (\_CURDRV)

Rôle : Indiquer le numéro du disque actuel (disque par défaut).

Entrée : Rien

Sortie : A = L = Disque actuel (0 = A:, 1= B:, etc)

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

## Fonction 1Ah - Set Disk Transfer Address (\_SETDTA)

Rôle : Définir l'adresse de transfert des disques (DTA).

Entrée : DE = Adresse de transfert des disques. (0080h par défaut)

Cette zone de transfert sera utilisée par toutes les fonctions BDOS qui utilisent un FCB pour la recherche de fichier, ouvrir un fichier, ainsi que pour la lecture et écriture de secteurs. Elle n'est pas utilisé par les nouvelles fonctions du MSX-DOS 2. L'adresse est remise à 0080h par la fonction 0Dh (\_DSKRST).

Sortie : Rien

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2.

## Fonction 1Bh - Get Allocation Information (\_ALLOC)

Rôle : Obtenir des informations sur le formatage du disque du disque spécifié.

Entrée : E = Numéro du disque (0 = Actuel, 1 = « A: », etc)

Sortie : A = Secteurs par cluster, FFh si le numéro de disque entrée est invalide.

F = L'indicateur CF est à 1 si le numéro de disque entrée est invalide.

BC = Taille du secteur (Toujours 512)

DE = Nombre total de clusters sur le disque

HL = Clusters libres sur le disque

IX = Pointeur sur DPB

IY = Pointeur sur le premier secteur FAT

Notes :

- Cette fonction n'est pas compatible avec CP/M qui utilise ce numéro de fonction pour renvoyer l'adresse d'un vecteur d'attribution.
- Contrairement au MSX-DOS 1, seul le premier secteur de la FAT est accessible à partir de l'adresse indiquée dans IY et les données qui y figurent resteront valides jusqu'au prochain appel MSX-DOS.

Compatibilité : MSX-DOS 1 et 2.

## Fonction 21h - Random Read (\_RDRND)

Rôle : Lecture directe de l'enregistrement d'un fichier de la taille de 128 octets.

Entrée : DE = Pointeur vers le FCB.

RN (FCB+33 à FCB+35) spécifie la position de l'enregistrement à lire.

Sortie : A = L = Erreur (1 si la lecture atteint la fin du fichier), 0 si la lecture est un succès.

Contrairement au CP/M v2.2, les trois octets du numéro d'enregistrement sont utilisés.

L'enregistrement lus est transférés vers la DTA. Si celui à la fin du fichier est partiel, il sera complété avec des zéros.

Le numéro d'enregistrement pour accès direct (FCB+33) n'est pas modifié par la fonction. Le numéro de l'enregistrement actuel et de l'étendue sont paramétrés pour correspondre au même enregistrement que le numéro d'enregistrement pour accès direct. Cela signifie que les lectures séquentielles (ou écritures) peuvent suivre une lecture directe et commenceront à partir du même enregistrement. L'octet du nombre d'enregistrements est également paramétré correctement pour l'étendue.

Note : Si la compatibilité avec le CP/M n'est pas nécessaire, il est préférable d'utiliser la fonction 27h (\_RDBLK) qui offre bien plus d'avantages.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2.

## Fonction 22h - Random Write (\_WRRND)

Rôle : Écriture directe d'un enregistrement de la taille de 128 octets dans un fichier.

Entrée : DE = Pointeur du FCB du fichier ouvert.

RN (FCB+33 à FCB+35) spécifie la position de l'enregistrement.

Les octets à écrire doivent être pointés par la DTA.

Sortie : A = L = 01H Si erreur (disk full), 0 Si sans erreur.

Le numéro d'enregistrement direct du FCB ne sera pas modifié, mais les champs de l'enregistrement actuel et du bloc seront paramétrés pour correspondre au même enregistrement. Le nombre d'enregistrements sera ajusté si nécessaire si le fichier est en cours d'extension ou si l'écriture passe dans une nouvelle mesure.

Si la position de l'enregistrement est au-delà de la fin du fichier, un espace disque non utilisé sera alloué pour combler le vide.

Note : Si la compatibilité avec le CP/M n'est pas nécessaire, il est préférable d'utiliser la fonction 26h (\_WRBLK) qui offre bien plus d'avantages.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2.

## Fonction 23h - Get File Size (\_FSIZE)

Rôle : Indiquer la taille d'un fichier. Cette fonction est particulièrement utile pour ajouter des enregistrements à la suite des autres d'un fichier à accès direct existant.

Entrée : DE = Pointeur sur un FCB ouvert ou fermé.

Le nom (FNNAME et FEXT) peut contenir des métacaractères. Dans ce cas, c'est la taille du premier fichier qui correspond au filtre sera indiqué.

Sortie : A = L = 0FFH si fichier non trouvé, 0 si fichier trouvé.

La taille du fichier (FILSIZ) est arrondie au nombre d'enregistrement multiplié par leur taille. RN (FCB+33 à FCB+35) donnera le numéro suivant le dernier enregistrement.

Modifie : Tous les registres.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2.

## Fonction 24h - Set Random Record (\_SETRND)

Rôle : Définir le numéro de l'enregistrement à accès direct RN (FCB+33 à FCB+35) à partir du numéro d'enregistrement séquentiel indiqué par la taille des enregistrements, numéro du bloc et le numéro de l'enregistrement du bloc actuel.

Entrée : DE = Pointeur pour ouvrir le FCB

Sortie : Rien. Cette fonction change juste le numéro d'enregistrement direct. Aucune vérification n'est faite pour savoir si l'enregistrement existe réellement dans le fichier.

Note : Cette fonction est buguée au point d'être inutilisable sur certains anciens contrôleurs de lecteur de disquette.

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2.

## Fonction 26h - Random Block Write (\_WRBLK)

Rôle : Ecriture directe d'un ou plusieurs enregistrements.

La taille des enregistrements est déterminée par les octets S2 et RC du FCB (FCB+14 et FCB+15). Celle-ci doit être définie après l'ouverture du fichier et avant l'appel de cette fonction. L'écriture de grand enregistrements sera beaucoup plus rapide qu'avec la fonction 22h compatible CP/M.

Entrée : DE = Pointeur du FCB ouvert.

HL = Nombre de bloc à écrire.

Si HL=0, aucune donnée ne sera écrite, mais la taille du fichier sera modifiée pour atteindre la valeur spécifiée par le champ d'enregistrement pour accès direct. Cela peut être plus long ou plus court que la taille actuelle du fichier et l'espace disque sera alloué ou libéré selon les besoins. L'espace disque supplémentaire alloué de cette manière ne sera pas initialisé à une valeur particulière.

Sortie : A = 00h si l'écriture s'est déroulée sans erreur, sinon 01h.

Une erreur sera renvoyée si le nombre d'enregistrements dépasse 64 Ko. Si vous le souhaitez, la taille des enregistrements peut être définie à 1 et le nombre d'enregistrement devient alors un nombre d'octets. Il est souhaitable d'écrire autant que possible avec un seul appel de fonction car un gros transfert sera plus rapide que plusieurs petits.

BC = HL = Nombre d'enregistrements qui ont été écrit. Ce nombre est ajouté à la valeur RN du FCB (FCB+33 à FCB+35) afin d'écrire à la suite la fois suivante.

Compatibilité : MSX-DOS 1 et 2.



## Fonction 27h - Random Block Read (\_RDBLK)

Rôle : Lecture directe d'un ou plusieurs enregistrements.

La taille des enregistrements est déterminée par les octets S2 et RC du FCB (FCB+14 et FCB+15). Celle-ci doit être définie après l'ouverture du fichier et avant l'appel de cette fonction. La lecture de grands enregistrements sera beaucoup plus rapide qu'avec la fonction 21h compatible CP/M.

Si vous souhaitez lire 20k d'un fichier, par exemple, il est préférable de lire le 20k avec un appel à la fonction plutôt que 20 appels pour lire 1k.

Entrée : DE = Pointeur vers le FCB ouvert.

HL = Nombre d'enregistrement à lire. Ce nombre peut être supérieur au nombre d'enregistrements restants. Si la fin du fichier est rencontrée, seuls les enregistrements restants seront lus.

Sortie : A = 00h la lecture s'est déroulée sans erreur, sinon 01h (souvent pour cause de fin de fichier).

Une erreur sera renvoyée si le nombre d'enregistrements dépasse 64 Ko. Si vous le souhaitez, la taille des enregistrements peut être définie à 1 et le nombre d'enregistrement devient alors un nombre d'octets. Il est souhaitable d'écrire autant que possible avec un seul appel de fonction car un gros transfert sera plus rapide que plusieurs petits.

BC = HL = Nombre d'enregistrement lus. Ce nombre est ajouté à la valeur RN du FCB (FCB+33 à FCB+35) afin d'écrire à la suite la fois suivante.

Compatibilité : MSX-DOS 1 et 2.

## Fonction 28h - Random Write With Zero Fill (\_WRZER)

Rôle : Écriture directe d'un enregistrement de la taille de 128 octets dans un fichier. Cette fonction est similaire à la fonction 22h (\_WRRND) à la différence que si des enregistrement précédents celui spécifié n'ont pas encore été écrits, il le seront avec des zéros.

Entrée : Les octets à écrire doivent être pointés par la DTA.

DE = Pointeur vers le FCB ouvert.

RN (FCB+33 à FCB+35) spécifie la position de l'enregistrement.

Sortie : A = L = 01H si erreur, 0 si pas d'erreur

Compatibilité : CP/M v2.2, MSX-DOS 1 et 2.

## Fonction 2Ah - Get Date (\_GDATE)

Rôle : Donner la date.

Entrée : Rien

Sortie : HL = Année (1980~2079)

D = Mois (1 pour Janvier, ..., 12 pour Décembre)

E = Date (1, ..., 31)

A = Jour de la semaine (0 pour Dimanche, 1 pour Lundi, ..., 6 pour Samedi)

Compatibilité : MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

## Fonction 2Bh - Set Date (\_SDATE)

Rôle : Régler la date. Les ordinateurs MSX2 ou supérieur utilisent une RTC qui actualise la date en temps réel même l'ordinateur éteint.

Entrée : HL = Année (1980~2079)

D = Mois (1 pour Janvier, ..., 12 pour Décembre)

E = Date (1, ..., 31)

A = Jour de la semaine (0 pour Dimanche, 1 pour Lundi, ..., 6 pour Samedi)

Sortie : A = 0 si date valide, sinon FFh.

Compatibilité : MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

Note : La validité de la date fournie est vérifiée et si elle est valide, elle est utilisée à la place de l'ancienne date. Les contrôles de validité incluent une vérification complète du nombre de jours de chaque mois et des années bissextiles. Si la date n'est pas valide, la date ne sera pas modifiée.

## Fonction 2Ch - Get Time (\_GTIME)

Rôle : Donner l'heure.

Entrée : Rien

Sortie : H = Heures (0~23)

L = Minutes (0~59)

D = Secondes (0~59)

E = Réserve pour les centièmes de secondes (toujours à zéro)

Compatibilité : MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

## Fonction 2Dh - Set Time (\_STIME)

Rôle : Régler l'heure du système. Les ordinateurs MSX2 ou supérieur utilisent une RTC qui actualise l'heure en temps réel même l'ordinateur éteint.

Entrée : H = Heures (0~23)

L = Minutes (0~59)

D = Secondes (0~59)

E = Centièmes de secondes (toujours à zéro)

Sortie : A = 0 si les paramètres sont valides, sinon FFh et l'heure actuelle restera inchangée.

Compatibilité : MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

Note : La validité de l'heure fournie est vérifiée et si elle est valide, elle est utilisée à la place de l'ancienne.

## Fonction 2Eh - Set/Reset Verify Flag (\_VERIFY)

Rôle : Activer ou désactiver la vérification automatique des écritures sur les disques. La vérification améliore la fiabilité du système mais ralentit les écritures. Cette fonction est désactivée par défaut, et n'est pas prise en charge par certains pilotes de disque.

Entrée : E = 0 pour activer la vérification, autrement elle sera désactivée.

Sortie : Rien

Le cache à la DTA contiendra ce qui a été lu.

Compatibilité : MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

## Fonction 2Fh - Absolute Sector Read (\_RDABS)

Rôle : Lire des secteurs directement sur un disque sans tenir compte des fichiers.

Entrée : DE = Numéro du premier secteur

H = Nombre de secteurs à lire

L = Numéro du disque (0=A: etc.)

Sortie : A = Code d'erreur. (0 si pas d'erreur)

Les secteurs seront lus à l'adresse de transfert des disques actuelle.

Notes :

- Le disque doit être un disque au format DOS valide pour que le numéro de secteur soit traduit dans une position physique sur le disque.
- Cette fonction est incompatible avec les disques formatés en FAT16. Utilisez la fonction 73h (\_RDDRIV) à la place sous Nextor.

Compatibilité : MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

## Fonction 30h - Absolute Sector Write (\_WRABS)

Rôle : Écrire des secteurs directement sur un disque sans tenir compte des fichiers.

Entrée : DE = Numéro du premier secteur

H = Nombre de secteurs à écrire

L = Numéro du disque (0 = A:, etc.)

Sortie : A = Code d'erreur. (0 si pas d'erreur)

Les secteurs seront écrits avec les données se trouvant à l'adresse de transfert des disques actuelle.

Notes :

- Le disque doit être un disque au format DOS valide pour que le numéro de secteur soit traduit dans une position physique sur le disque.
- Cette fonction est incompatible avec les disques formatés en FAT16. Utilisez la fonction 74h (\_WRDRV) à la place sous Nextor.

Compatibilité : MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

## Fonction 31h - Get Disk Parameters (\_DPARM)

Rôle : Créer une table de 32 octets contenant les paramètres relatifs au format du disque du disque spécifié, dans le cache à l'adresse spécifiée. Cette fonction est utile pour les programmes qui vont lire ou écriture des secteurs absolus, afin de pouvoir interpréter le nombre de secteurs absolus.

Entrée : DE = Adresse de la table à créer. La table doit se trouver dans la zone réservée à l'utilisateur.

L = Numéro du disque (0 = défaut, 1 = A:, etc.)

Sortie : A = Code d'erreur. (0 si pas d'erreur)

DE = Adresse de la table créée (ne change pas).

Sous MSX-DOS 1, les paramètres sont accessibles de la façon suivante.

DE + 0 = Numéro de disque physique (1 = A: etc)

DE + 1 = Taille des secteurs (toujours 512 jusqu'à maintenant)

DE + 3 = Nombre de secteurs par Cluster (Valeur réelle =  $n^2$ , avec  $n > 0$ )

DE + 4 = Nombre de secteurs réservés (généralement 1)

DE + 6 = Nombre de copies de la FAT (généralement 2)

DE + 7 = Nombre d'entrées du dossier racine (sur 2 octets)

DE + 9 = Nombre total de secteur logique (sur 2 octets). 0 sous Nextor lorsque le nombre de secteur est supérieur à 65535. (Voir DE + 24)

DE + 11 = Type de média.

F8h pour une disquette 360K, 3,5", simple-face, 9 secteur

F9h pour une disquette 720K, 3,5", double-face, 9 secteur

FAh pour une disquette 320K, 3,5", simple-face, 8 secteur

FBh pour une disquette 640K, 3,5", double-face, 8 secteur

FCh pour une disquette 180K, 5,25", simple-face, 9 secteur

FDh pour une disquette 360K, 5,25", double-face, 9 secteur

FEh pour une disquette 160K, 5,25", simple-face, 8 secteur

FFh pour une disquette 320K, 5,25", double-face, 8 secteur

DE + 12 = Nombre de secteurs par FAT

DE + 13 = Numéro du premier secteur de la racine des dossiers (sur 2 octets)

DE + 15 = Numéro du premier secteur des données (sur 2 octets)

DE + 17 = Nombre maximal de Cluster (sur 2 octets)

DE + 19 = Indicateur de disque corrompu

DE + 20 = Identificateur de volume. (-1 pour aucun identifiant) (sur 4 octets)

DE + 24 = Réservés (sur 8 octets, tous à zéro).

Nombre total de secteur logique (sur 4 octets) sous Nextor.

DE + 28 = 0 pour FAT12, 1 pour FAT16. (Nextor seulement)

L'indicateur de disque corrompu permet de savoir s'il y a sur le disque un fichier pouvant être récupéré par la commande UNDEL. L'indicateur est réinitialisé lorsque l'allocation du fichier est terminée.

Compatibilité : MSX-DOS 1 et 2 (même pendant le traitement d'une erreur).

## **Fonction 40h - Find First Entry (\_FFIRST)**

Rôle : Trouver le premier fichier ou dossier correspondant au nom de la chaîne de caractères [ASCIIZ](#) ou au FIB spécifié.

Entrée : B = Attributs du fichier ou dossier à trouver. (Voir [la description sur les attributs](#) page 451)

Les attributs spécifient le type d'élément à trouver. Si tous les bits sont à zéro, seul un fichier qui n'est pas du système et qui n'est pas caché pourra être trouvé. Pour chercher un dossier, mettez le bit 4 à 1. Les bits 0 (lecture seule) et 5 (archive) sont ignorés.

Si le bit 3 (nom de volume) est mis à 1, la recherche est exclusive, seul le nom de volume sera recherché. Dans ce cas également, le FIB et le nom du fichier ou, la chaîne « Disque\Chemin\Fichier » sont ignorés sauf pour spécifier le disque car le nom du volume sera toujours recherché dans la racine du disque.

DE = Pointeur vers la chaîne de caractères ASCIIZ "Disque/Chemin/Fichier" ou, le FIB du fichier.

La partie "Disque/Chemin" de la chaîne, ou le FIB, spécifie le dossier dans lequel effectuer la recherche. Si DE pointe sur un FIB, IX peut, si vous le souhaitez, pointer sur le même FIB. Dans ce cas, lorsqu'une correspondance est trouvée, le nouveau bloc info fichier remplace l'ancien.

Le nom du fichier peut contenir des métacaractères ("?" ou "\*"), auquel cas le premier élément correspondant sera trouvée. Si le nom de fichier est vide (la chaîne de caractères ASCIIZ pointée par DE est nulle ou se termine par un "\" ou la chaîne pointée par HL est vide), cette fonction se comportera exactement comme si le nom de fichier était "\*. \*" donc n'importe quel nom correspondra.

HL = Adresse d'une chaîne de caractères ASCIIZ contenant le nom du fichier. (Ignorée si DE pointe vers une chaîne de caractères ASCIIZ "Disque/Chemin/Fichier").

Sortie : A = Code d'erreur. (0 si pas d'erreur)

Le code d'erreur renvoyée est 0CFh si un FIB spécifiant un fichier est transmis. La partie "Fichier" de la chaîne, ou le nom de fichier de la chaîne de caractères ASCIIZ pointé par HL, détermine le nom de fichier correspondants. Si aucun élément n'est trouvé, le code d'erreur est 0D7h.

IX = Adresse du nouveau FIB.

Compatibilité : MSX-DOS 2.

## **Fonction 41h - Find Next Entry (\_FNEXT)**

Rôle : Poursuivre la recherche d'un fichier ou dossier trouvée avec la fonction 40h (\_FFIRST).

Cette fonction peut être utilisée plusieurs fois après un appel de fonction 40h en utilisant les métacaractères.

Entrée : IX = Pointeur vers le FIB de la recherche précédente.

La recherche se fait, dans le dossier du disque spécifié.

Sortie : A = Code d'erreur. (0 si pas d'erreur)

S'il n'y a plus d'élément qui correspond, le code erreur renvoyée est 0D7h (.NOFIL). Le FIB est actualisé avec les informations relatives lorsqu'un élément a été trouvé.

IX = Adresse du nouveau FIB.

Compatibilité : MSX-DOS 2.

## **Fonction 42h - Find New Entry (\_FNEW)**

Rôle : Trouver un nouveau fichier ou dossier.

Cette fonction est similaire à la fonction 40h (\_FFIRST) décrite ci-dessus. Cependant, au lieu de faire seulement une recherche correspondante au nom spécifié dans le dossier indiqué, elle permet de créer un nouvel élément vide par dessus.

Entrée : B = Attributs de fichier.

Bits 0~6 = Attributs (Voir [la description sur les attributs](#) page 451)

Bit 7 = 1 pour ne pas créer de nouvel élément.

Les attributs spécifient le type de fichier ou dossier à créer. Si le bit 4 est à 1, il s'agit d'un dossier, sinon un fichier. Les bits 2 (fichier système) et 0 (fichier en lecture seule) peuvent être mis à 1 pour un fichier, et le bit 1 (fichier caché) peut être mis à 1 pour un fichier un dossier. Si le bit 7 est à 1, l'élément ne sera pas créé sur le disque.

Un seul cluster sera alloué pour créer un élément. Celui-ci sera vide avec la date et l'heure actuelles. Un fichier aura une longueur nulle et un dossier aura les éléments "." et ".." initialisées de manière appropriée.

DE = Pointeur vers la chaîne de caractères [ASCIIZ](#) "Disque/Chemin/Fichier" ou, le FIB du fichier.

HL = Adresse d'une chaîne de caractères ASCIIZ contenant le nom du fichier. (Ignorée si DE pointe vers une chaîne de caractères ASCIIZ "Disque/Chemin/Fichier").

IX = Adresse du FIB modèle.

Les paramètres dans HL et DE sont utilisés exactement de la même manière pour simuler spécifier une entrée de dossier. Toutefois, au lieu de rechercher dans le dossier sélectionné une entrée correspondant au nom spécifié, une nouvelle entrée sera créée avec ce nom. Le FIB pointé par IX sera rempli d'informations sur la nouvelle entrée comme si elle avait été trouvée avec la fonction 40h (\_FFIRST).

Sortie : A = Code d'erreur. (0 si pas d'erreur)

S'il n'y a plus d'élément qui correspond, le code erreur renvoyée est 0D7h (.NOFIL). Le FIB est actualisé avec les informations relatives lorsqu'un élément a été trouvé.

Comme avec la fonction 40h (\_FFIRST), si le nom du fichier est null, il sera traité exactement comme s'il s'y avait les métacaractères "\*" "\*" mais pour cette fonction, cela signifie que le nom de « fichier modèle » sera utilisé comme nouveau nom de fichier à créer.

Le code d'erreur 0D5h (.DRFUL) sera renvoyée s'il n'y a pas de place dans la racine des dossiers, si un dossier doit être étendu ou si le disque est déjà plein.

S'il existe déjà une entrée avec le nom spécifié dans le dossier, l'action dépend de l'indicateur "créer un nouveau" (bit 7 du registre B) et du type d'élément. Si l'indicateur "créer un nouveau" est défini, le code d'erreur 0CBh (.FILEX) sera toujours renvoyée. La définition de cet indicateur garantit donc qu'un fichier existant ne sera pas supprimé. Si une entrée existe déjà et que l'indicateur "créer une nouvelle" n'est pas défini, le type de l'entrée existante est examiné pour voir s'il peut être supprimé pour laisser de la place au nouveau fichier. L'élément ne sera pas créé et un code erreur sera renvoyée si l'élément est un fichier en lecture seule (0D1h), un fichier système (0CDh), un dossier (0CCh), s'il y a un descripteur de fichier déjà ouvert pour ce fichier (0CAh) ou si vous essayez de créer un dossier avec le même nom qu'un élément déjà existant (0CBh).

Pour tous ces codes d'erreur, 0CBh (.FILEX), 0D1h (.FILRO), 0CDh (.SYSX), 0CCh (.DIRX) et 0CAh (.FOPEN), le FIB sera archivé avec les détails du fichier existant entry et ce FIB peut être utilisé exactement comme s'il avait été renvoyé par une fonction 40h (\_FFIRST).

IX = Adresse du nouveau FIB.

Un fichier sera toujours créé avec le bit 5 (archive) des attributs mis à 1 dans le FIB. Si des métacaractères ("?" Ou "\*"") étaient dans le nom du fichier, ils sont remplacés par les caractères du nom de fichier du « FIB modèle ». Si aucun fichier n'est trouvé, le code d'erreur 0DAh (.IFNM) est renvoyée. Ceci est utile pour les opérations de copie faisant un changement de nom automatique.

Compatibilité : MSX-DOS 2.

## **Fonction 43h - Open File Handle (\_OPEN)**

Rôle : Ouvrir un descripteur de fichier.

Le fichier spécifié est automatiquement ouvert et le descripteur de fichier correspondant est renvoyé dans le registre B.

Entrée : DE = Pointeur vers une chaîne de caractères [ASCIIZ](#) "Disque/Chemin/Fichier" ou un FIB.

La chaîne "Disque\Chemin\Fichier" ou le FIB doit normalement faire référence uniquement à un fichier. Celui-ci sera ouvert et prêt à être lu et/ou écrit (en fonction du mode d'ouverture) et un nouveau descripteur de fichier sera retourné dans le registre B. Le plus petit numéro de descripteur de fichier disponible sera utilisé.

A = Mode d'ouverture.

Le bit 0 sert à interdire l'écriture.

Le bit 1 sert à interdire la lecture.

Le bit 2 définit l'héritabilité.

Les bits 3~7 doivent être mis à 0.

Sortie : A = Code d'erreur. (0 si pas d'erreur)

Si la chaîne pointée par DE en entrée contient un nom de volume au lieu d'un nom de fichier, le



code d'erreur renvoyé est 0CFh (.IATTR). S'il s'agit d'un dossier, le code d'erreur renvoyé est 0CCh (.DIRX).

Si il n'y a pas de descripteurs de fichiers de disponible, le code renvoyé est 0C4h (.NHAND). Si la mémoire est insuffisante, le code est 0DEh (.NORAM).

Si le bit 1 (lecture interdite) est à 1 en entrée, alors les lectures du descripteur de fichier seront rejetées avec le code d'erreur 0C6h (.ACCV).

Si le bit 0 (écriture interdite) est à 1, les écritures seront rejetées avec 0D1h (.FILRO)).

Si le bit 2 (héritabilité) est à 1, le descripteur de fichier sera hérité par un nouveau processus créé par l'appel de la fonction 60h (\_FORK).

Si un descripteur de fichier de périphérique est ouvert en donnant un nom de fichier correspondant à l'un des périphériques intégrés (par exemple, "CON" ou "NUL"), il sera toujours initialement ouvert en mode ASCIIZ. La fonction 4Bh (\_IOCTL) peut être utilisée pour passer en mode binaire, mais vous devez faire très attention lors de la lecture d'appareils en mode binaire car il n'y a pas de condition de fin de fichier.

B = Descripteur du fichier ouvert.

Compatibilité : MSX-DOS 2.

## **Fonction 44h - Create File Handle (\_CREATE)**

Rôle : Créer un descripteur de fichier.

Le fichier spécifié est automatiquement créé et ouvert et le descripteur de fichier correspondant est renvoyé dans le registre B.

Entrée : DE = Pointeur vers une chaîne de caractères [ASCIIZ](#) "Disque/Chemin/Fichier" ou un FIB. Le nom et le dossier dans lequel il sera créé est défini par cette chaîne.

A = Mode d'ouverture.

Le bit 0 sert à interdire l'écriture

Le bit 1 sert à interdire la lecture

Le bit 2 définit l'héritabilité

Les bits 3~7 doivent être mis à 0

Ce paramètre est interprété de la même manière que pour la fonction 43h (\_OPEN).

Un dossier ne sera pas ouvert (car cela n'a pas de sens), donc le registre B sera retourné sous la forme 0FFh, qui ne peut jamais être un descripteur de fichier valide.

B = Attributs du fichier.

Bits 0~6 = Attributs requis. (Voir [la description sur les attributs](#) page 451)

Bit 7 = Mettre à 1 pour qu'un fichier déjà existant ne soit pas écrasé.

Un fichier de type tel que spécifié par les attributs sera créé. Le code d'erreur 0CFh (.IATTR) est renvoyé si le registre B spécifie un nom de volume.

Les bits 0 (lecture seule), 1 (fichier caché) et 2 (fichier système) peuvent être à 1 ou 0 pour créer un fichier aux propriétés correspondantes. Les attributs invalides seront simplement ignorés. Un fichier sera toujours créé avec le bit 5 (archive) des attributs à 1.

Sortie : A = Code d'erreur. (0 si pas d'erreur)

Un code d'erreur est renvoyé lorsque le fichier ne peut pas être créé. Les conditions

d'erreur dans ce cas sont les mêmes que pour la fonction 42h (\_FNEW), les principaux codes d'erreur étant 0CBh (.FILEX), 0CCh (.DIRX), 0CDh (.SYSX), 0D1h (.FILRO), 0CAh (.FOPEN), 0D5h (.DRFUL) et 0D4h (.DKFUL).

B = Nouveau descripteur de fichier

Compatibilité : MSX-DOS 2.

### **Fonction 45h** - Close File Handle (\_CLOSE)

Rôle : Fermer un descripteur de fichier.

Entrée : B = Descripteur de fichier.

Sortie : A = Code d'erreur. (0 si pas d'erreur)

Cette fonction libère le descripteur du fichier spécifié. Le fichier associé sera mis à jour avec la date et l'heure actuelle, le bit 5 (archive) des attributs sera mis à 1 et toutes les données mises en mémoire cache y seront écrites si nécessaire. S'il existe d'autres copies de ce descripteur de fichier, créées avec la fonction 47h (\_DUP) ou 60h (\_FORK), ceux-là pourront toujours être utilisées.

Compatibilité : MSX-DOS 2.

### **Fonction 46h** - Ensure File Handle (\_ENSURE)

Rôle : S'assurer que le fichier associé au descripteur de fichier spécifié soit actualisé sur le disque.

Entrée : B = Descripteur de fichier.

Sortie : A = Code d'erreur. (0 si pas d'erreur)

Le fichier associé au descripteur sera mis à jour avec la nouvelle date et la nouvelle heure, le bit 5 (archive) des attributs sera mis à 1 et toutes les données mises en mémoire cache seront écrites sur le disque. Le descripteur de fichier reste toujours disponible et peut donc toujours être utilisé pour accéder au fichier. Le réglage actuel du pointeur de fichier ne sera pas modifié.

Compatibilité : MSX-DOS 2.

### **Fonction 47h** - Duplicate File Handle (\_DUP)

Rôle : Dupliquer un descripteur de fichier.

Le plus petit numéro de descripteur de fichier disponible sera toujours utilisé et une erreur ".NHAND" renvoyée s'il n'y en a pas. Le nouveau descripteur de fichier fera référence au même fichier que l'original et l'un ou l'autre peut être utilisé. Si le pointeur de fichier d'une des poignées est déplacé, l'autre le sera également. Si l'une des poignées est fermée, l'autre peut toujours être utilisée.

Notez que, puisque les descripteurs de fichiers en double créés par cette fonction ne sont pas "ouverts séparément", ils ne sont pas considérés comme des descripteurs de fichiers distincts pour générer des erreurs ".FOPEN". Ainsi, par exemple, un descripteur de fichier "DUP" peut être renommé (fonction 53h) ou avoir ses attributs modifiés (fonction

55h) et l'effet s'appliquera aux deux descripteurs de fichier. Notez en particulier que si une copie d'un descripteur de fichier "DUP" est supprimée (fonction 54h), le fichier sera réellement supprimé et l'autre, bien qu'il soit encore ouvert, ne peut plus être utilisé en toute sécurité. S'il est utilisé (autre que d'être fermé, assuré ou supprimé), une erreur ".FDEL" sera renvoyée.

Entrée : B = Descripteur de fichier

Sortie : A = Code d'erreur. (0 si pas d'erreur)

B = Nouveau descripteur de fichier

Compatibilité : MSX-DOS 2.

## **Fonction 48h** - Read From File Handle (\_READ)

Rôle : Lire depuis un descripteur de fichier.

Lire dans un fichier le nombre d'octets spécifiés par HL à partir de la position actuelle du pointeur du gestionnaire de fichier. Les octets lus sont chargés à l'adresse spécifiée par le registre DE. Le pointeur est ensuite mis à jour avec l'adresse de l'octet suivant le dernier octet lu. Une erreur ".ACCV" sera renvoyée si le descripteur de fichier a été ouvert avec le bit d'accès "lecture interdite".

Le nombre d'octets lus peut être inférieur au nombre demandé pour différentes raisons, et le nombre lu sera retourné dans le registre HL s'il n'y a pas d'erreur. En général, si moins que ce qui est demandé est lu, cela ne devrait pas être traité comme une condition d'erreur, mais une autre lecture devrait être effectuée pour lire la partie suivante, jusqu'à ce qu'une erreur ".EOF" soit renvoyée. Une erreur ".EOF" ne sera jamais renvoyée pour une lecture partielle, mais uniquement pour une lecture qui lit zéro octet. La lecture des fichiers de cette manière garantit le bon fonctionnement des descripteurs de fichier de périphérique (voir ci-dessous).

Pour les fichiers sur disque, le nombre d'octets lus ne sera inférieur au nombre demandé que si la fin du fichier est atteinte. Dans ce cas, la prochaine opération de lecture lira zéro octet et renverra une erreur ".EOF". Lors de la lecture d'un descripteur de fichier de périphérique (par exemple, les descripteurs de fichier standard 0 à 4), le comportement dépend du périphérique et de la lecture en mode [ASCIIZ](#) ou binaire (voir la fonction 4Bh ci-dessous). Le périphérique "CON" sera décrit à titre d'exemple car il s'agit du périphérique le plus couramment utilisé, mais les autres périphériques se comportent de la même manière.

Lors de la lecture à partir du périphérique "CON" en mode binaire, les caractères sont lus sur le clavier, sans interprétation et sans écho sur l'écran ou l'imprimante. Le nombre exact de caractères demandés sera toujours lu et il n'y a pas de condition de fin de fichier. En raison de l'absence d'indication de fin de fichier, un soin particulier doit être pris lors de la lecture de périphériques en mode binaire.

Un appel de fonction de lecture à l'appareil "CON" en mode ASCIIZ (le mode par défaut et celui qui s'applique normalement au canal d'entrée standard), ne lit qu'une ligne d'entrée. La ligne d'entrée sera lue à partir du clavier avec les fonctions d'édition de ligne normales disponibles pour l'utilisateur, et le caractère saisi sera renvoyé à l'écran et à l'imprimante si CTRL-P est activé. Les caractères de contrôle spéciaux "CTRL-P", "CTRL-N", "CTRL-S" et "CTRL-C" seront testés et traités exactement comme pour la fonction d'état de la console 0Bh.

Lorsque l'utilisateur tape un retour chariot, la ligne sera copiée dans le tampon de lecture,

terminée par une séquence CR-LF et la fonction de lecture retournera avec un nombre d'octets approprié. La prochaine lecture démarrera une autre opération d'entrée de ligne mise en mémoire tampon. Si le nombre d'octets demandés lors de la lecture était inférieur à la longueur de l'entrée de ligne, le nombre de caractères demandé sera renvoyé et le prochain appel de fonction de lecture sera immédiatement renvoyé avec la partie suivante de la ligne jusqu'à la lecture complète. .

Si l'utilisateur tape une ligne qui commence par un caractère "CTRL-Z", cela sera interprété comme indiquant la fin du fichier. La ligne sera supprimée et l'appel de fonction lu lira zéro octet et retournera une erreur ".EOF". Une lecture ultérieure après cela reviendra à la normale et lancera une autre entrée de ligne. La condition de fin de fichier n'est donc pas permanente.

Entrée : B = Descripteur de fichier  
DE = Adresse de destination  
HL = Nombre d'octets à lire

Sortie : A = Code d'erreur. (0 si pas d'erreur)  
HL = Nombre d'octets lu

Compatibilité : MSX-DOS 2.

## **Fonction 49h - Write To File Handle (\_WRITE)**

Rôle : Écrire vers un descripteur de fichier.

Cette fonction est très similaire à la fonction "lire" ci-dessus (fonction 48h). Le nombre d'octets spécifié sera écrit à la position actuelle du pointeur de fichier dans le fichier, et le pointeur de fichier sera ajusté pour pointer juste après le dernier octet écrit. Si le fichier a été ouvert avec le bit d'accès "pas d'écriture", une erreur ".ACCV" sera renvoyée et si le fichier est en lecture seule, une erreur ".FILRO" est renvoyée.

Si l'écriture dépasse la fin du fichier en cours, le fichier sera étendu si nécessaire. Si le pointeur de fichier se trouve déjà au-delà de la fin du fichier, de l'espace disque sera alloué pour combler le vide et ne sera pas initialisé. Si l'espace disque est insuffisant, une erreur ".DKFUL" sera renvoyée et aucune donnée ne sera écrite, même s'il y avait de la place pour certaines de ces données.

Le nombre d'octets écrits peut généralement être ignoré puisqu'il sera égal à zéro si une erreur est renvoyée ou égal au nombre demandé si l'écriture a réussi. Il est beaucoup plus efficace d'écrire des fichiers dans quelques gros blocs que dans plusieurs petits. Par conséquent, les programmes doivent toujours essayer d'écrire dans des blocs aussi grands que possible.

Cette fonction définit un bit "modifié" pour le descripteur de fichier, ce qui garantit que, lorsque le descripteur de fichier est fermé ou garanti, explicitement ou implicitement, l'entrée du dossier est mise à jour avec les nouvelles informations de date, d'heure et d'attribution. De plus, le bit 5 (archive) des attributs sera mis à 1 pour indiquer que ce fichier a été modifié lors du dernier archivage.

L'écriture dans les descripteurs de fichiers de périphériques n'est pas une tâche compliquée, car il n'y a aucune condition de fin de fichier ni d'entrée de ligne à s'inquiéter. Il existe certaines différences entre le mode [ASCIIZ](#) et le mode binaire lors de l'écriture sur le périphérique "CON", en ce sens qu'une vérification de l'état de la console est effectuée en mode ASCIIZ uniquement. De plus, l'écho de l'imprimante, s'il est activé, ne sera effectué qu'en mode ASCIIZ.

Entrée : B = Descripteur de fichier  
DE = Adresse du tampon  
HL = Nombre d'octets à écrire  
Sortie : A = Code d'erreur. (0 si pas d'erreur)  
HL = Nombre d'octets déjà écrits  
Compatibilité : MSX-DOS 2.

### **Fonction 4Ah** - Move File Handle Pointer (\_SEEK)

Rôle : Déplacer le pointeur d'un descripteur de fichier.  
Le pointeur de fichier associé au descripteur de fichier spécifié sera modifié en fonction du code de la méthode et de l'offset, et la nouvelle valeur du pointeur renvoyée dans DE: HL. Le code de méthode spécifie où le décalage signé est relatif comme suit:

A = 0 par rapport au début du fichier  
A = 1 par rapport à la position actuelle  
A = 2 par rapport à la fin du fichier.

Notez qu'un décalage de zéro avec un code de méthode de 1 retournera simplement la valeur actuelle du pointeur, et avec un code de méthode de 2 retournera la taille du fichier. Aucune vérification de fin de fichier n'est effectuée, il est donc tout à fait possible (et parfois utile) de définir le pointeur de fichier au-delà de la fin du fichier. S'il existe des copies de ce descripteur de fichier créées par la fonction "descripteur de fichier dupliqué" (fonction 47h) ou la fonction "fourchette" (fonction 60h), leur pointeur de fichier sera également modifié.

Le pointeur de fichier n'a de signification réelle que sur les fichiers du disque, car l'accès direct est possible. Sur les fichiers de périphérique, le pointeur de fichier est mis à jour de manière appropriée lorsqu'une lecture ou une écriture est effectuée, et peut être examiné ou modifié par cette fonction. Cependant, le changement n'aura aucun effet et son examen ne sera probablement pas utile.

Entrée : B = Descripteur de fichier  
A = Méthode de codage  
DE & HL = Offset signé  
Sortie : A = Code d'erreur. (0 si pas d'erreur)  
Compatibilité : MSX-DOS 2.

### **Fonction 4Bh** - I/O Control For Devices (\_IOCTL)

Rôle : Contrôle des E/S pour les périphériques.  
Le pointeur de fichier associé au descripteur de fichier spécifié sera modifié en fonction

du code de la méthode et de l'offset, et la nouvelle valeur du pointeur renvoyée dans DE: HL. Le code de méthode spécifie où le décalage signé est relatif comme suit:

A = 0 par rapport au début du fichier

A = 1 par rapport à la position actuelle

A = 2 par rapport à la fin du fichier.

Notez qu'un décalage de zéro avec un code de méthode de 1 retournera simplement la valeur actuelle du pointeur, et avec un code de méthode de 2 retournera la taille du fichier. Aucune vérification de fin de fichier n'est effectuée, il est donc tout à fait possible (et parfois utile) de définir le pointeur de fichier au-delà de la fin du fichier. S'il existe des copies de ce descripteur de fichier créées par la fonction "descripteur de fichier dupliqué" (fonction 47h) ou la fonction "fourchette" (fonction 60h), leur pointeur de fichier sera également modifié.

Le pointeur de fichier n'a de signification réelle que sur les fichiers du disque, car l'accès direct est possible. Sur les fichiers de périphérique, le pointeur de fichier est mis à jour de manière appropriée lorsqu'une lecture ou une écriture est effectuée, et peut être examiné ou modifié par cette fonction. Cependant, le changement n'aura aucun effet et son examen ne sera probablement pas utile. Cette fonction permet d'examiner et de modifier divers aspects des descripteurs de fichiers. En particulier, il peut être utilisé pour déterminer si un descripteur de fichier fait référence à un fichier de disque ou à un périphérique. Ceci est utile pour les programmes qui souhaitent se comporter différemment pour les fichiers de disque et les E / S de périphérique.

Cette fonction reçoit le descripteur de fichier dans le registre B et un code de sous-fonction dans le registre A qui spécifie l'une des différentes opérations. Tous les autres paramètres requis par la sous-fonction particulière sont passés dans le registre DE et les résultats sont renvoyés dans le registre DE. Si le code de sous-fonction n'est pas valide, une erreur ".ISBFN" sera renvoyée.

Si A = 0, l'opération est "obtenir le statut du descripteur de fichier". Ceci retourne un mot de drapeaux donnant diverses informations sur le descripteur de fichier. Le format de ce mot est différent pour les descripteurs de fichier de périphérique et de disque, et le bit 7 spécifie le nom. Le format du mot est le suivant :

Pour les appareils :

DE = b0 set => périphérique d'entrée console

b1 set => périphérique de sortie de la console

b2~b4 réservé

b5 set => mode ASCII, clear => mode binaire

b6 set => fin de fichier

b7 toujours défini (=> périphérique)

b8..b15 réservé

Pour les fichiers sur disque :

DE = b0..b5 numéro de disque (0 = A: etc)

b6 set => fin de fichier

b7 toujours 0 (=> fichier disque)

b8..b15 réservé

Notez que l'indicateur de fin de fichier est le même pour les périphériques que pour les fichiers sur disque. Pour les périphériques, il sera défini si la tentative précédente de lecture à partir du périphérique a généré une erreur ".EOF" et sera effacé lors de la prochaine lecture. Pour les fichiers sur disque, il est élaboré en comparant le pointeur de fichier avec la taille du fichier.

Si A = 1, l'opération est un "mode ASCII / binaire défini". Cette opération n'est autorisée que pour les descripteurs de fichier de périphérique. Un drapeau ASCII / binaire doit être passé dans le bit 5 du registre E (exactement là où il est retourné par "get descripteur de fichier status"). Ceci est défini pour le mode ASCII et vide pour le mode binaire. Tous les autres bits du registre DE sont ignorés.

Si A = 2 ou 3, l'opération est respectivement "entrée de test prête" ou "sortie de test prête". Dans les deux cas, un indicateur est renvoyé dans le registre E, qui est FFh si le descripteur de fichier est prêt pour un caractère et 00h sinon. La signification exacte de "prêt pour un personnage" dépend de l'appareil. Les descripteurs de fichier de disque sont toujours prêts pour la sortie et pour l'entrée sauf si le pointeur de fichier se trouve à la fin du fichier. Le dispositif "CON" vérifie l'état du clavier pour déterminer s'il est prêt à être entré.

Si A = 4, l'opération est "obtenir la taille de l'écran". Cela retourne la taille d'écran logique du descripteur de fichier avec le nombre de lignes dans le registre D et le nombre de colonnes dans le registre E. Pour les périphériques sans taille d'écran (tels que les fichiers de disque), les valeurs D et E seront nulles. Zéro pour l'un ou l'autre résultat doit donc être interprété comme "illimité". Par exemple, cette fonction est utilisée par la commande "DIR / W" pour décider du nombre de fichiers à imprimer par ligne. La valeur zéro pour le registre E est définie par défaut sur 80.

Entrée : B = Descripteur de fichier

A = Code de sous fonction

0 => récupère le statu du descripteur de fichier

1 => Défini le mode ASCII/binaire

2 => teste si l'entrée est prête

3 => teste si la sortie est prête

4 => Trouve la taille de l'écran

DE = Autre paramètre

Sortie : A = Code d'erreur. (0 si pas d'erreur)

DE = Autre résultat

Compatibilité : MSX-DOS 2.

## **Fonction 4Ch** - Test File Handle (\_HTEST)

Rôle : Tester un descripteur de fichier.

Cette fonction plutôt spécialisée reçoit un descripteur de fichier et une chaîne disque\chemin\fichier ou un bloc info fichier qui identifie un fichier. Il détermine si les

deux fichiers sont réellement le même fichier et renvoie un indicateur indiquant le résultat. Notez que si le descripteur de fichier concerne un périphérique plutôt qu'un fichier sur disque, il renvoie toujours "B = 00h" pour indiquer "pas le même fichier".

Cette fonction permet à la commande "COPY" de détecter certaines conditions d'erreur, telles que la copie d'un fichier sur eux-mêmes, et de donner à l'utilisateur des messages d'erreur informatifs. Cela peut également être utile pour d'autres programmes qui doivent faire des tests similaires.

Entrée : B = Descripteur de fichier

DE = Chaîne [ASCIIZ](#) "Disque/Chemin/Fichier" , ou pointeur vers un FIB.

Sortie : A = Code d'erreur. (0 si pas d'erreur)

B = 00H => Pas le même fichier, FFH => Même fichier

Compatibilité : MSX-DOS 2.

### **Fonction 4Dh** - Delete File Or Subdirectory (\_DELETE)

Rôle : Supprimer un fichier ou un dossier.

Cette fonction supprime l'objet (fichier ou sous-dossier) spécifié par la chaîne Disque\Chemin\Fichier ou le bloc d'info fichier. Les caractères de nom de fichier globaux ne sont pas autorisés, aussi un seul fichier ou sous-dossier peut être supprimé avec cette fonction. Un sous-dossier ne peut être supprimé que s'il est vide ou si une erreur (".DIRNE") survient sinon. Le "." et les entrées ".." d'un sous-dossier ne peuvent pas être supprimées (erreur ".DOT"), pas plus que la racine des dossiers. Un fichier ne peut pas être supprimé si un descripteur de fichier lui est ouvert (erreur .FOPEN) ou s'il est en lecture seule (erreur .FILRO).

S'il s'agit d'un fichier, tout espace disque qui lui a été attribué sera libéré. Si le disque est un disque MSX-DOS 2, suffisamment d'informations sont conservées sur le disque pour permettre au programme utilitaire "UNDEL" de restaurer le fichier. Ces informations ne sont conservées que jusqu'à la prochaine allocation d'espace disque (généralement une écriture dans un fichier) sur ce disque. Après avoir passé cet appel de fonction, si un bloc info fichier a été passé, il ne doit plus être utilisé (à part le passer à une fonction "trouver la prochaine entrée") car le fichier auquel il fait référence n'existe plus.

Si un nom de périphérique tel que "CON" est spécifié, aucune erreur ne sera renvoyée, mais le périphérique ne sera pas supprimé.

Entrée : B = Descripteur de fichier

DE = Chaîne [ASCIIZ](#) "Disque/Chemin/Fichier", ou pointeur vers un FIB.

Sortie : A = Code d'erreur. (0 si pas d'erreur)

Compatibilité : MSX-DOS 2.

### **Fonction 4Eh** - Rename File Or Subdirectory (\_RENAME)

Rôle : Renommer un fichier ou un dossier.

Cette fonction renomme l'objet (fichier ou sous-dossier) spécifié par la chaîne Disque/Chemin/Fichier ou le bloc d'info fichier, avec le nouveau nom de la chaîne indiquée par HL. La nouvelle chaîne de nom de fichier ne doit pas contenir de lettre de



disque ni de chemin de dossier (erreur ".IFNM" si tel est le cas). Si un nom de périphérique tel que "CON" est spécifié, aucune erreur ne sera renvoyée, mais le périphérique ne sera pas renommé.

Les caractères de nom de fichier globaux ne sont pas autorisés dans la chaîne Disque/Chemin/Fichier, vous ne pouvez donc renommer qu'un seul objet avec cette fonction. Toutefois, les caractères de nom de fichier globaux sont autorisés dans le nouveau nom de fichier transmis dans HL et, là où ils se produisent, le caractère de nom de fichier existant reste inchangé. Des vérifications sont effectuées pour éviter de créer un nom de fichier illégal. Par exemple, un fichier appelé "XYZ" ne peut pas être renommé avec une nouvelle chaîne de nom de fichier de "???? A" car le nouveau nom de fichier serait "XYZ A", ce qui est illégal. Dans ce cas, une erreur ".IFNM" sera renvoyée.

S'il existe déjà une entrée avec le nouveau nom de fichier, une erreur (".DUPF") est renvoyée pour éviter de créer des noms de fichiers en double. Le "." et les entrées ".." d'un sous-dossier ne peuvent pas être renommés (erreur ".IDOT"), pas plus que la racine des dossiers (il n'a pas de nom). Un fichier ne peut pas être renommé si un descripteur de fichier y est ouvert (erreur ".FOPEN"), bien qu'un fichier en lecture seule puisse être renommé.

Notez que si DE pointe sur un FIB, il n'est pas mis à jour avec le nouveau nom du fichier. Par conséquent, le bloc info fichier doit être utilisé avec soin après cet appel de fonction.

Entrée : B = Descripteur de fichier

DE = Chaîne [ASCIIZ](#) "Disque/Chemin/Fichier", ou pointeur vers un FIB.

HL = Nouveau fichier chaîne ASCIIZ.

Sortie : A = Code d'erreur. (0 si pas d'erreur)

Compatibilité : MSX-DOS 2.

## **Fonction 4Fh** - Move File Or Subdirectory (\_MOVE)

Rôle : Déplacer un fichier ou un dossier.

Cette fonction déplace l'objet (fichier ou sous-dossier) spécifié par la chaîne Disque/Chemin/Fichier ou le bloc info fichier vers le dossier spécifié par la nouvelle chaîne de chemin indiquée par HL. Il ne doit pas y avoir de nom de disque dans la nouvelle chaîne de chemin. Si un nom de périphérique tel que "CON" est spécifié, aucune erreur ne sera renvoyée, mais le périphérique ne sera pas réellement déplacé.

Les caractères de nom de fichier globaux ne sont autorisés dans aucune des chaînes. Un seul objet (fichier ou sous-dossier) peut donc être déplacé par cette fonction. Toutefois, si un sous-dossier est déplacé, tous ses descendants le seront également. S'il existe déjà une entrée du nom requis dans le dossier cible, une erreur ".DUPF" est renvoyée pour empêcher la création de noms de fichiers en double. Le "." et ".." entrées dans un sous-dossier ne peuvent pas être déplacées (erreur ".DOT") et un dossier ne peut pas être déplacé dans l'un de ses propres descendants (erreur ".DIRE") car cela créerait une boucle isolée dans la Système de dépôt. Un fichier ne peut pas être déplacé si un descripteur de fichier lui est ouvert (erreur ".FOPEN").

Notez que si un bloc info fichier est transmis à cette fonction, les informations internes du bloc info fichier ne sont pas mises à jour pour refléter le nouvel emplacement du fichier. Cela est nécessaire car sinon le bloc info fichier ne pourrait pas être utilisé pour un appel de fonction "trouver le suivant" ultérieur. Cependant, cela signifie que le bloc

info fichier ne fait plus référence au fichier déplacé et ne doit donc être utilisé pour aucune opération dessus, telle que "renommer" ou "ouvrir".

Entrée : B = Descripteur de fichier

DE = Chaîne [ASCIIIZ](#) "Disque/Chemin/Fichier", ou pointeur vers un FIB.

HL = Nouveau fichier chaîne ASCII.

Sortie : A = Code d'erreur. (0 si pas d'erreur)

Compatibilité : MSX-DOS 2.

### **Fonction 50h** - Get/Set File Attributes (\_ATTR)

Rôle : Lire/Régler les attributs de fichier.

Cette fonction est normalement utilisée pour modifier les attributs d'un fichier ou d'un sous-dossier. Il peut également être utilisé pour rechercher les attributs actuels, mais le plus souvent avec la fonction "rechercher la première entrée" (fonction 40h). Si A = 0, l'octet d'attribut actuel du fichier ou du sous-dossier sera simplement renvoyé dans le registre L.

Si A = 1, l'octet d'attribut sera défini sur la nouvelle valeur spécifiée dans le registre L et cette nouvelle valeur sera également renvoyée dans le registre L. Seuls les bits système, caché, en lecture seule et d'archivage peuvent être modifiés pour un fichier, et seulement le bit caché pour un sous-dossier. Le code d'erreur 0CFh (.IATTR) sera renvoyée si l'on tente de modifier d'autres bits d'attribut. Si un bloc info fichier est passé, l'octet d'attribut qu'il contient ne sera pas mis à jour avec le nouveau paramètre.

Les caractères de nom de fichier globaux ne sont pas autorisés, ainsi un seul objet (fichier ou sous-dossier) peut avoir ses attributs définis par cette fonction. Les attributs du dossier racine ne peuvent pas être modifiés car il n'en a pas. Les attributs d'un fichier ne peuvent pas être modifiés si un descripteur de fichier lui est ouvert (erreur ".FOPEN"). Les attributs du "." et ".." les entrées du dossier peuvent cependant être changées. Si un nom de périphérique tel que "CON" est spécifié, aucune erreur ne sera renvoyée, mais les attributs du périphérique ne seront pas réellement modifiés (car il n'en a pas).

Entrée : DE = Chaîne [ASCIIIZ](#) "Disque/Chemin/Fichier", ou pointeur vers un FIB.

A = 0 => Attributs, 1 => défini les attributs

L = Nouvel Octet d'attribut (seulement si A=1)

Sortie : A = Code d'erreur. (0 si pas d'erreur)

L = Octet de l'attribut courant

Compatibilité : MSX-DOS 2.

### **Fonction 51h** - Get/Set File Date And Time (\_FTIME)

Rôle : Lire/Régler la date et l'heure du fichier.

Si A = 1, cette fonction définit la date et l'heure de la dernière modification du fichier ou du sous-dossier spécifié par le Disque/Chemin/Fichier chaîne ou le FIB. Les caractères de nom de fichier globaux ne sont autorisés dans aucune partie de la chaîne, de sorte que la date et l'heure d'un seul fichier peuvent être modifiées par cette fonction. Si un nom de

périphérique tel que "CON" est spécifié, aucune erreur ne sera renvoyée, mais la date et l'heure du périphérique ne seront pas modifiées.

Le format de date et d'heure est exactement tel qu'il est contenu dans l'entrée du dossier et les blocs info fichier (voir "Spécification de l'interface du programme"). Aucune vérification n'est effectuée pour les dates ou les heures sensibles, les valeurs sont simplement stockées. Notez que si un bloc info fichier est passé, la date et l'heure qui y sont stockées ne seront pas mises à jour par cette fonction.

Si A = 0, les valeurs actuelles sont simplement renvoyées. Notez que bien que la valeur time soit passée dans IX, elle est renvoyée dans DE. La date et l'heure d'un fichier ne peuvent pas être modifiées (bien qu'il puisse être lu) si un descripteur de fichier est ouvert sur le fichier (erreur ".FOPEN").

Entrée : DE = Chaîne [ASCIIZ](#) "Disque/Chemin/Fichier", ou pointeur vers un FIB.

A = 0 pour récupérer la date et l'heure, 1 pour définir la date et l'heure

IX = Nouvelle valeur d'heure (seulement si A=1)

HL = Nouvelle valeur de date (Seulement si A=1)

Sortie : A = Code d'erreur. (0 si pas d'erreur)

DE = Valeur de d'heure courante

HL = Valeur de date courante

Compatibilité : MSX-DOS 2.

## **Fonction 52h** - Delete File Handle (\_HDELETE)

Rôle : Supprimer un descripteur de fichier.

Cette fonction supprime le descripteur de fichier associé au fichier spécifié et ferme le descripteur de fichier. Un descripteur de fichier ne peut pas être supprimé s'il existe d'autres descripteurs de fichier ouverts séparément ouverts sur le même fichier (erreur ".FOPEN"). S'il existe des doublons du descripteur de fichier (créés par une fonction "descripteur de fichier en double" ou "fourchette"), ces doublons seront marqués comme non valides et toute tentative d'utilisation entraînera une erreur ".HDEAD".

Les conditions d'erreur pour cette fonction sont les mêmes que pour la fonction "supprimer un fichier ou un sous-dossier" (fonction 4Dh). Le descripteur de fichier sera toujours fermé, même s'il existe une condition d'erreur telle que ".FILRO" ou ".FOPEN".

Entrée : B = Descripteur de fichier

Sortie : A = Code d'erreur. (0 si pas d'erreur)

Compatibilité : MSX-DOS 2.

## **Fonction 53h** - Rename File Handle (\_HRENAME)

Rôle : Renommer un descripteur de fichier.

Cette fonction renomme le fichier associé au descripteur de fichier spécifié avec le nouveau nom de la chaîne indiquée par HL. Outre le fait que le fichier est spécifié par un descripteur de fichier plutôt que par une chaîne de caractères [ASCIIZ](#) ou un FIB, cette fonction est identique à la fonction "renommer un fichier ou un sous-dossier" (fonction

4Eh) et présente les mêmes conditions d'erreur.

Un descripteur de fichier ne peut pas être renommé s'il existe d'autres descripteurs de fichier ouverts séparément pour ce fichier (erreur ".FOPEN"), bien qu'il puisse être renommé s'il existe des copies de ce descripteur de fichier. Dans ce cas, les copies seront renommées. Renommer un descripteur de fichier ne modifiera pas le pointeur de fichier, mais effectuera une opération "assurer" implicite.

Entrée : B = Descripteur de fichier

HL = Nouvelle chaîne ASCIIZ contenant le nom du fichier.

Sortie : A = Code d'erreur. (0 si pas d'erreur)

Compatibilité : MSX-DOS 2.

### **Fonction 54h - Move File Handle (\_HMOVE)**

Rôle : Déplacer un descripteur de fichier.

Cette fonction déplace le fichier associé au descripteur de fichier spécifié dans le dossier spécifié par la nouvelle chaîne de chemin indiquée par HL. Outre le fait que le fichier est spécifié par un descripteur de fichier plutôt que par une chaîne de caractères [ASCIIZ](#) ou un FIB, cette fonction est identique à la fonction "déplacer un fichier ou un sous-dossier" (fonction 4Fh) et présente les mêmes conditions d'erreur.

Un descripteur de fichier ne peut pas être déplacé s'il existe d'autres descripteurs de fichier ouverts séparément pour ce fichier (erreur ".FOPEN"), bien qu'il puisse être déplacé s'il existe des copies de ce descripteur de fichier; dans ce cas, les copies seront également déplacées. . Déplacer un descripteur de fichier ne modifiera pas le pointeur de fichier, mais effectuera une opération "assurer" implicite.

Entrée : B = Descripteur de fichier

HL = Nouvelle chaîne ASCIIZ contenant le nom de fichier.

Sortie : A = Code d'erreur. (0 si pas d'erreur)

Compatibilité : MSX-DOS 2.

### **Fonction 55h - Get/Set File Handle Attributes (\_HATTR)**

Rôle : Lire/Régler les attributs d'un descripteur de fichier. Cette fonction obtient ou définit l'octet d'attributs du fichier associé au descripteur de fichier spécifié. Outre le fait que le fichier est spécifié par un descripteur de fichier plutôt que par une chaîne de caractères [ASCIIZ](#) ou un FIB, cette fonction est identique à la fonction "obtenir / définir les attributs de fichier" (fonction 50h) et présente les mêmes conditions d'erreur.

Un descripteur de fichier ne peut pas avoir ses attributs modifiés (bien qu'ils puissent être lus) s'il existe d'autres descripteurs de fichier ouverts séparément pour ce fichier (erreur ".FOPEN"). Le pointeur de fichier ne sera pas modifié, mais une opération "s'assurer" implicite sera effectuée.

Entrée : B = Descripteur de fichier

A = 0 => Récupère attributs, 1 => Définit attributs

L = Nouvel octet d'attributs byte (Seulement si A=1)

Sortie : A = Code d'erreur. (0 si pas d'erreur)  
L = Octet d'attribut courant. (Voir [la description sur les attributs](#) page 451)  
Compatibilité : MSX-DOS 2.

### **Fonction 56h** - Get/Set File Handle Date And Time (\_HFTIME)

Rôle : Lire/Régler la date et l'heure d'un descripteur de fichier.  
Cette fonction obtient ou définit la date et l'heure du fichier associé au descripteur de fichier spécifié. Outre le fait que le fichier est spécifié par un descripteur de fichier plutôt que par une chaîne de caractères ASCII ou un bloc d'info fichier, cette fonction est identique à la fonction "get / set file date and time" (fonction 51h) et présente les mêmes conditions d'erreur. .  
La date et l'heure d'un descripteur de fichier ne peuvent pas être modifiées (bien qu'elles puissent être lues) s'il existe d'autres descripteurs de fichier ouverts séparément pour ce fichier (erreur ".FOPEN"). Le pointeur de fichier ne sera pas modifié, mais une opération "s'assurer" implicite sera effectuée.

Entrée : B = Descripteur de fichier  
A = 0 => Récupère date et heure, 1 => Définit date et heure  
L = Nouveaux attributs (Seulement si A=1) (Voir [la description sur les attributs](#) page 451)  
IX = Nouvelle valeur d'heure (seulement si A=1)

Sortie : A = Code d'erreur. (0 si pas d'erreur)  
DE = Valeur d'heure courante  
HL = Valeur de date courante

Compatibilité : MSX-DOS 2.

### **Fonction 57h** - Get Disk Transfer Address (\_GETDTA)

Rôle : Indiquer l'adresse de transfert des disques (DTA) actuelle.  
Cette adresse n'est utilisée que pour les fonctions FCB de type CP/M "traditionnel" et pour les fonctions de lecture et d'écriture de secteur absolu.

Entrée : Rien

Sortie : DE = Adresse de transfert des disques actuelle

Compatibilité : MSX-DOS 2 (même pendant le traitement d'une erreur).

### **Fonction 58h** - Get Verify Flag Setting (\_GETVFY)

Rôle : Lire le paramètre de l'indicateur de vérification.  
Cette fonction renvoie simplement l'état actuel de l'indicateur de vérification, qui peut être défini avec la fonction MSX-DOS 2Eh.

Entrée : Rien

Sortie : B = 0 pour désactiver la vérification, FFh pour activer la vérification

Compatibilité : MSX-DOS 2 (même pendant le traitement d'une erreur).

### **Fonction 59h** - Get Current Directory (\_GETCD)

Rôle : Obtenir le dossier actuel.

Cette fonction renvoie une chaîne de caractères contenant le chemin du dossier ouvert du disque spécifié dans B à l'adresse spécifiée par DE. La chaîne n'inclura pas de nom de disque ni de caractère "\" en fin de chaîne. On accédera au disque pour s'assurer que le dossier en cours existe réellement sur le disque en cours. Sinon, le dossier en cours sera redéfini sur la racine et une chaîne vide sera renvoyée.

Entrée : B = Numéro du disque (0 = Disque actuel, 1 = A, 2 = B, etc)

DE = Pointeur vers tampon de 64 octets

Sortie : A = Code d'erreur. (0 si pas d'erreur)

DE = Rempli avec le chemin actuel

Compatibilité : MSX-DOS 2.

### **Fonction 5Ah** - Change Current Directory (\_CHDIR)

Rôle : Changer de dossier actuel.

La chaîne Disque/Chemin/Fichier doit spécifier un dossier plutôt qu'un fichier. Le dossier actuel du disque sera changé pour être ce dossier. Si le dossier spécifié n'existe pas, le paramètre actuel reste inchangé et une erreur ".NODIR" est renvoyée.

Entrée : DE = Pointeur vers une chaîne de caractères contenant le chemin

Sortie : A = Code d'erreur. (0 si pas d'erreur)

Compatibilité : MSX-DOS 2.

### **Fonction 5Bh** - Parse Pathname (\_PARSE)

Rôle : Analyser le chemin d'un dossier.

Cette fonction est purement une fonction de manipulation de chaîne. Elle n'effectuera aucun accès aux disques et ne modifiera pas la chaîne de l'utilisateur. Elle est destinée à aider les programmes transitoires à analyser les lignes de commande.

Entrée : B = Indicateur de nom de Volume (bit 4).

Ce bit se trouvera dans le même état que celui du nom de volume dans un octet d'attribut) détermine si la chaîne sera analysée comme une chaîne "Disque/Chemin/Fichier" (si le bit est à 0) ou une chaîne "disque/volume" (si le bit est à 1).

DE = chaîne ASCII pour l'analyse

Sortie : A = Code d'erreur. (0 si pas d'erreur)

DE = Pointeur sur le caractère de fin ou le premier non valide dans la chaîne.

HL = Pointeur du le premier caractère dernier élément (partie du nom de fichier).

Par exemple, lorsqu'une chaîne "A:\XYZ\P.Q /F" est passée, DE pointe sur le caractère d'espace blanc avant "/" F" et HL pointe sur "P". Si la chaîne analysée se termine par le caractère "\" ou est null (à l'exception du nom du disque), il n'y aura pas de "dernier élément". HL et DE désigneront donc le même caractère. Dans ce cas, des procédures spéciales seront nécessaires pour tous les programmes utilisant cette fonction.

B = Indicateurs des résultats de l'analyse.

Pour un nom de volume, les bits 1, 4, 5, 6 et 7 seront toujours à 0. Pour un nom de fichier, les bits 3 à 7 se rapportent au dernier élément de la chaîne (le nom de fichier). Les affectations de bits sont les suivantes :

bit 0 = 1 si des caractères autres que le nom du disque sont analysés

bit 1 = 1 si un chemin dans les dossiers est spécifié

bit 2 = 1 si le nom du disque est spécifié

bit 3 = 1 si le nom de fichier principal spécifié dans la dernière partie

bit 4 = 1 si l'extension du nom de fichier spécifiée dans la dernière partie

bit 5 = 1 si la dernière partie est ambigu

bit 6 = 1 si la dernière partie est "." ou ".."

bit 7 = 1 si la dernière partie est ".."

C = Numéro du disque logique spécifié en début de chaîne (1=A:, etc).

Si la chaîne ne commence pas par indiquer le disque, le registre C contiendra le numéro du disque par défaut.

Compatibilité : MSX-DOS 2.

## **Fonction 5Ch** - Parse Filename (\_PFILE)

Rôle : Analyser le nom de fichier.

Cette fonction est purement une fonction de manipulation de chaîne, elle n'accédera pas du tout aux disques et ne modifiera pas la chaîne du tout. Il est principalement destiné à aider les programmes transitoires à imprimer les noms de fichiers de manière formatée. La chaîne de caractères ASCII sera analysée comme un élément de nom de fichier unique et le nom de fichier sera stocké dans la mémoire tampon de 11 octets de l'utilisateur sous forme développée, avec le nom de fichier et l'extension complétés avec des espaces.

Les drapeaux d'analyse retournés dans le registre B sont identiques à ceux de la fonction "chemin d'accès" ci-dessus (fonction 5Bh), sauf que les bits 0, 1 et 2 seront toujours vides. Le tampon de l'utilisateur sera toujours rempli, même s'il n'y a pas de nom de fichier valide dans la chaîne, auquel cas le tampon sera rempli d'espaces. "\*" caractères seront étendus au nombre approprié de "?" s. Si le nom du fichier ou son extension est trop long, les caractères en excès seront ignorés.

Le pointeur renvoyé dans le registre DE pointe vers le premier caractère de la chaîne qui ne faisait pas partie du nom de fichier, ce qui peut être le caractère nul situé à la fin de la

chaîne. Ce caractère ne sera jamais un caractère de nom de fichier valide (voir "Spécification de commande" pour plus de détails sur les caractères de nom de fichier valides).

Entrée : DE = chaîne ASCII pour l'analyse  
HL = Pointeur vers tampon de 11 byte octets

Sortie : A = Code d'erreur. (0 si pas d'erreur)  
DE = Pointeur Caractère de fin  
HL = Préserve  
B = Flag d'analyse

Compatibilité : MSX-DOS 2.

### **Fonction 5Dh** - Check Character (\_CHKCHR)

Rôle : Vérification de caractère.  
Cette fonction autorise un langage indépendamment des majuscules, et facilite également la gestion des caractères sur 16 bits et des noms de fichiers.

Entrée : D = Paramètres du caractère  
bit 0 = Mettre à 1 pour forcer la mise en minuscule, 0 pour mise en majuscule en fonction du réglage de la langue de la machine.  
bit 1 = Mettre à 1 si premier octet d'un caractère sur 16 bits  
bit 2 = Mettre à 1 si deuxième octet d'un caractère sur 16 bits  
bit 3 = Mettre à 1 si nom de volume, 0 pour nom de fichier  
bit 5~7 = Réserve (toujours à 0)  
Note : Faites attention avec les indicateurs des bits 1 et 2 lorsque vous revenez en arrière car des chaînes peuvent contenir des caractères 16 bits.  
E = Caractère à vérifier

Sortie : A = Code d'erreur. (0 si pas d'erreur)  
D = Indicateurs de caractère  
bit 4 = 1 si le caractère du nom de fichier / volume est valide  
bit 5~7 = Réserve (toujours à 0)  
E = Caractère vérifié (majuscule)

Compatibilité : MSX-DOS 2 (même pendant le traitement d'une erreur).

### **Fonction 5Eh** - Get Whole Path String (\_WPATH)

Rôle : Obtenir le chemin complet.  
Cette fonction copie simplement une chaîne de chemin [ASCIIZ](#) d'un tampon interne dans le tampon de l'utilisateur. La chaîne représente le chemin d'accès complet et le nom du fichier, à partir du dossier racine, d'un fichier ou d'un sous-dossier situé par une fonction précédente "rechercher la première entrée" ou "rechercher une nouvelle entrée". La



chaîne renvoyée n'inclura ni disque, ni caractère "\" initial. Register HL pointe sur le premier caractère du dernier élément de la chaîne, exactement comme pour la fonction "chemin d'analyse" (fonction 5Bh).

Si un appel de fonction "trouver la première entrée" ou "trouver une nouvelle entrée" est effectué avec DE pointant sur une chaîne de caractères ASCII, un appel de fonction "obtenir le chemin complet" retournera une chaîne représentant le sous-dossier ou le fichier correspondant au bloc info fichier retourné par la fonction "find". S'il s'agit d'un sous-dossier, le bloc info fichier peut être renvoyé dans le registre DE à un autre appel de fonction "trouver la première entrée", qui localisera un fichier dans ce sous-dossier. Dans ce cas, le fichier nouvellement localisé sera ajouté en interne à la chaîne de chemin d'accès existante déjà existante. Un appel de fonction "obtenir la chaîne de chemin d'accès entier" retournera par conséquent une chaîne de chemin d'accès correcte pour le fichier localisé.

Vous devez faire très attention lorsque vous utilisez cette fonction car la chaîne du chemin complet interne est modifiée par de nombreux appels de fonction et, dans de nombreux cas, peut être invalide. L'appel de la fonction "obtenir le chemin complet" doit être effectué immédiatement après la fonction "trouver la première entrée" ou "trouver une nouvelle entrée" à laquelle elle se rapporte.

Entrée : DE = Pointeur vers tampon de 64 octets

Sortie : A = Code d'erreur. (0 si pas d'erreur)

DE = Rempli avec le chemin entier

HL = Pointeur vers le début du dernier objet

Compatibilité : MSX-DOS 2.

## **Fonction 5Fh** - Flush Disk Buffers (\_FLUSH)

Rôle : Débusquer les caches des disques.

Cette fonction informe le système qu'un processus enfant est sur le point d'être démarré. Il s'agit généralement d'un nouveau programme ou sous-commande en cours d'exécution. Par exemple, COMMAND2.COM effectue un appel de fonction 60h (\_FORK) avant d'exécuter une commande ou un programme transitoire.

Un nouveau jeu de descripteurs de fichier est créé et tous les descripteurs de fichier actuels ouverts avec le bit de mode d'accès "pouvant être hérité" (voir la fonction "ouvrir le descripteur de fichier" = fonction 43h) sont copiés dans le nouveau jeu de descripteurs de fichier. Tous les descripteurs de fichier ouverts avec le bit "pouvant être hérité" bien clair ne seront pas copiés et ne seront donc pas disponibles pour le processus enfant. Les descripteurs de fichier standard (00h ... 05h) sont héritables et seront donc copiés.

Un nouvel identifiant de processus est alloué pour le processus enfant et l'identifiant de processus du processus parent est renvoyé afin qu'un appel de fonction 61h (\_JOIN) plus tard puisse revenir au processus parent. Une erreur ".NORAM" peut être générée par cette fonction si la mémoire est insuffisante pour dupliquer les descripteurs de fichier.

Étant donné que le processus enfant dispose désormais d'une copie des descripteurs de fichier précédents plutôt que des originaux, si l'un d'entre eux est fermé, l'original reste ouvert. Ainsi, par exemple, si le processus enfant ferme le descripteur de fichier de sortie standard (descripteur de fichier numéro 1), il le rouvre dans un nouveau fichier, puis, lorsqu'une fonction 61h (\_JOIN) est créée pour renvoyer au processus parent le canal de sortie standard d'origine toujours être là.

Entrée : B = Numéro du disque (0=actuel, FFH=tout)  
D = 00H => Flush only, FFH => Flush and invalidate

Sortie : A = Code d'erreur. (0 si pas d'erreur)

Cette fonction vide tous les tampons de disque endommagés du disque spécifié ou de tous les disques si B = FFh. Si le registre D est FFh, tous les tampons de ce disque seront également invalidés.

Compatibilité : MSX-DOS 2.

### **Fonction 60h** - Fork A Child Process (\_FORK)

Rôle : Faire bifurquer un processus enfant.

La chaîne Disque/Chemin/Fichier doit spécifier un dossier plutôt qu'un fichier. Le dossier actuel du disque sera changé pour être ce dossier. Si le dossier spécifié n'existe pas, le paramètre actuel reste inchangé et une erreur ".NODIR" est renvoyée.

Entrée : Rien

Sortie : A = Code d'erreur. (0 si pas d'erreur)  
B = ID du processus parent

Compatibilité : MSX-DOS 2.

### **Fonction 61h** - Rejoin Parent Process (\_JOIN)

Rôle : Rejoindre le processus parent.

Cette fonction retourne au processus parent spécifié et renvoie le code d'erreur auquel s'est terminé le processus enfant dans le registre B, ainsi qu'un code d'erreur secondaire de l'enfant dans le registre C. Bien que la relation entre les processus parent et enfant soit strictement un à un. Premièrement, cette fonction peut remonter de plusieurs niveaux en lui attribuant un identifiant de processus approprié. Une erreur ".IPROC" sera renvoyée si l'ID de processus n'est pas valide.

L'ensemble de descripteurs de fichiers du processus enfant est automatiquement fermé et l'ensemble de descripteurs de processus du processus parent redevient actif. De même, tous les segments de RAM utilisateur alloués par le processus enfant seront libérés.

Si l'identifiant de processus transmis à cette fonction est zéro, une réinitialisation partielle du système est effectuée. Tous les descripteurs de fichier sont fermés et les descripteurs standard d'entrée et de sortie sont rouverts et tous les segments d'utilisateur sont libérés. Cela ne devrait normalement pas être effectué par un programme utilisateur s'il a l'intention de retourner à l'interpréteur de commandes, car celui-ci ne sera plus dans un état cohérent par la suite.

Cette fonction veille à ce que la libération de mémoire et le réglage de l'identifiant du processus soient effectués avant la fermeture des descripteurs de fichiers, et donc avant l'accès au disque. Cela garantit que si une erreur de disque se produit et est annulée, l'opération de jointure aura été effectuée avec succès. Toutefois, si une "jointure 0" produit une erreur de disque qui est annulée, la réinitialisation des descripteurs de fichier par défaut n'aura pas été effectuée. Dans ce cas, un autre appel de fonction "join 0" doit être effectué. Il ne tentera pas d'accéder au disque (car tous les fichiers ont été fermés) et

aboutira.

Notez que si cet appel de fonction est effectué via ROMBDOS (0F37Dh), les registres B et C ne renverront pas les codes d'erreur. Cela est dû au fait que la gestion du programme et de l'abandon du programme doit être effectuée par le programme d'application. Le code d'erreur aura été transmis au vecteur d'abandon et le code doit retenir le code d'erreur s'il en a besoin. Voir la fonction 62h (\_TERM) pour connaître la signification des codes d'erreur primaire et secondaire.

Entrée : B = ID du processus parent, ou zéro  
Sortie : A = Code d'erreur. (0 si pas d'erreur)  
B = Code primaire d'erreur de l'enfant  
C = Code secondaire d'erreur de l'enfant  
Compatibilité : MSX-DOS 2.

### **Fonction 62h** - Terminate With Error Code (\_TERM)

Rôle : Terminer avec le code d'erreur.

Cette fonction termine le programme avec le code d'erreur spécifié, qui peut être zéro et ne pas indiquer d'erreur. Cet appel de fonction ne retournera jamais à l'appelant (sauf si une routine d'abandon d'utilisateur exécute l'obligation de le voir = voir fonction 63h). Le fonctionnement de cette fonction est différent selon qu'elle a été appelée depuis l'environnement MSX-DOS via BDOS (0005h) ou depuis l'environnement BASIC du disque via ROMBDOS (0F37Dh).

Si elle est appelée via BDOS (0005h), si une routine d'abandon utilisateur a été définie par la fonction 63h, elle sera appelée avec le code d'erreur spécifié (et un code d'erreur secondaire nul). En supposant que cette routine soit renvoyée, ou si aucune routine d'abandon d'utilisateur n'a été définie, le contrôle sera renvoyé à celui qui a chargé le programme transitoire via un saut à l'adresse 00000h. Ce sera presque toujours l'interpréteur de commande, mais dans certains cas, il peut s'agir d'un autre programme transitoire. Le système mémorisera le code d'erreur et la prochaine fonction "rejoindre" (fonction 61h) qui sera effectuée renverra ce code d'erreur. L'interpréteur de commande imprimera un message d'erreur pour tout code de la plage 20h ... FFh, mais n'imprimera pas de message d'erreur plus bas.

Si cette fonction est appelée depuis l'environnement du Disk-BASIC à ROMBDOS (0F37Dh), le contrôle sera alors transmis au vecteur d'abandon à l'adresse BREAKVECT. Dans cet environnement, il n'y a pas de routine d'abandon définie séparément par l'utilisateur et le code d'erreur doit être mémorisé par le code situé dans BREAKVECT, car "rejoindre" ne renvoie pas le code d'erreur.

Entrée : B = Code d'erreur  
Sortie : Rien  
Compatibilité : MSX-DOS 2.

### **Fonction 63h** - Define Abort/Exit Routine (\_DEFAB)

Rôle : Définir la routine d'abandon/sortie.

Cette fonction n'est pas disponible en l'appelant à ROMBDOS (0F37Dh) depuis l'environnement Disk-BASIC. Elle ne peut être appelée que sous MSX-DOS2.

Si le registre DE contient zéro, alors la routine d'abandon définie précédemment restera indéfinie, sinon une nouvelle sera définie. La routine d'abandon sera appelée par le système lorsque le programme transitoire est sur le point de se terminer pour une raison autre qu'un saut direct à l'adresse 0000h. Les programmes écrits pour MSX-DOS 2 doivent quitter avec un appel de la fonction 62h (\_TERM) plutôt que par un saut vers l'adresse 0000h.

La routine d'abandon utilisateur sera entrée avec la pile d'utilisateurs active, avec IX, IY et le registre alternatif défini tel qu'il était au moment de l'appel de la fonction et avec l'ensemble du TPA paginé. Le code d'erreur de terminaison sera transmis à la routine dans le registre A avec un code d'erreur secondaire dans le registre B et si la routine exécute un "RET", les valeurs renvoyées dans les registres A et B seront stockées en tant que codes d'erreur à renvoyer par la fonction "join" et normalement imprimées par l'interprète de commande. Sinon, la routine peut sauter à un code de démarrage à chaud dans le programme transitoire plutôt que de revenir. Le système sera dans un état parfaitement stable capable d'accepter tous les appels de fonction.

Le code d'erreur principal transmis à la routine dans le registre A sera le code que le programme lui-même a transmis à la fonction 62h (\_TERM), qui peut être zéro, si tel est le motif de la terminaison. La routine sera également appelée si un CTRL-C ou CTRL-STOP est détecté (erreur ".CTRLC" ou ".STOP"), si une erreur de disque est abandonnée (erreur ".ABORT"), ou si une erreur est survenue le l'un des canaux d'entrée ou de sortie standard accessibles via les appels de fonctions MSX-DOS 01h ... 0Bh (".INERR" ou ".OUTERR").

Les erreurs ".ABORT", ".INERR" et ".OUTERR" sont générées par le système à la suite d'une autre erreur. Par exemple, un ".ABORT" peut résulter d'une erreur ".NRDY" ou un ".INERR" peut résulter d'une erreur ".EOF". Dans ces cas, le code d'erreur d'origine (".NRDY" ou ".EOF") est transmis à la routine d'abandon dans le registre B en tant que code d'erreur secondaire. Pour toutes les autres erreurs, il n'y a pas de code d'erreur secondaire et le registre B sera égal à zéro.

Si la routine d'abandon exécute "POP HL: RET" (ou l'équivalent) plutôt qu'un simple retour, le contrôle passera à l'instruction immédiatement après l'appel MSX-DOS ou l'appel du BIOS dans lequel l'erreur s'est produite. Cela peut être utile en conjonction avec une routine de descripteur d'erreur de disque (voir fonction 64h ci-dessous) pour permettre à une option d'abandonner l'appel MSX-DOS en cours lorsqu'une erreur de disque se produit.

Entrée : DE = Adresse de la routine d'abandon (0000H vers une routine indéfinie)

Sortie : A = 0 (Ne génère jamais d'erreur)

Compatibilité : MSX-DOS 2.

## **Fonction 64h** - Define Disk Error Handler Routine (\_DEFER)

Rôle : Définir la routine du descripteur d'erreurs de disque.

Cette fonction spécifie l'adresse d'une routine utilisateur qui sera appelée si une erreur de disque se produit. La routine sera entrée avec le TPA complet, mais avec la pile système de la page 3 active et aucun des registres ne sera conservé lors de l'appel de la fonction MSX-DOS.

La routine d'erreur peut effectuer des appels MSX-DOS, mais vous devez être très prudent pour éviter la récursivité. La liste des appels de fonction dans la section 2 de ce document indique quels appels de fonction peuvent être effectués en toute sécurité à partir d'une routine d'erreur utilisateur. Cette routine est appelée avec l'état de redirection invalidé temporairement si les canaux d'E / S standard ont été redirigés. Voir la fonction 70h (\_REDIR) pour plus de détails.

La spécification des paramètres et des résultats pour la routine elle-même est la suivante. Tous les registres, y compris IX, IY et le jeu de registres alternatif, peuvent être détruits mais la pagination et la pile doivent être préservées. La routine doit revenir dans le système, elle ne doit pas sauter pour continuer le programme transitoire. S'il veut faire cela, alors il devrait renvoyer A = 1 (« abort ») et une routine d'abandon d'utilisateur obtiendra alors le contrôle et cela pourra faire tout ce qu'il voudra.

Entrée :

A = code d'erreur qui a causé l'erreur

B = Disque physique

C = bit 0 = défini si écriture

bit 1 = défini si ignorer non recommandé

bit 2 = défini si l'abandon automatique est suggéré

bit 3 = défini si le numéro de secteur est valide

DE = numéro de secteur (si b3 de C est défini)

Sorties :

A = 0 => routine d'erreur du système d'appel

1 => Abandonner 2 => Réessayer 3 => Ignorer

Entrée : DE = Adresse de la routine d'erreur du disque (0000H pour annuler la définition de la routine)

Sortie : A = 0 (ne génère jamais d'erreurs)

Compatibilité : MSX-DOS 2.

## **Fonction 65h - Get Previous Error Code (\_ERROR)**

Rôle : Obtenir le code d'erreur précédent.

Cette fonction permet à un programme utilisateur de connaître le code d'erreur à l'origine de l'échec du précédent appel de fonction MSX-DOS. Il est conçu pour être utilisé avec les anciennes fonctions compatibles CP/M qui ne renvoient pas de code d'erreur. Par exemple, si une fonction qui crée un FCB renvoie A = 0FFh, il peut y avoir plusieurs raisons à cet échec et cet appel de fonction renverra l'actuel, par exemple ".DRFUL" ou ".SYSX".

Entrée : Rien

Sortie : A = 0

B = Code d'erreur de la fonction précédente

Compatibilité : MSX-DOS 2 (même pendant le traitement d'une erreur).

## **Fonction 66h** - Explain Error Code (\_EXPLAIN)

Rôle : Expliquer le code d'erreur.

Cette fonction permet à un programme utilisateur d'obtenir une chaîne d'explication ASCII pour un code d'erreur donné renvoyé par l'une des fonctions MSX-DOS. Si une erreur provient d'une des anciennes fonctions, il faut d'abord appeler "obtenir le code d'erreur précédent" pour obtenir le code d'erreur réel, puis cette fonction peut être appelée pour obtenir une chaîne d'explication.

La "Spécification d'interface de programme" contient une liste de tous les codes d'erreur actuellement définis et de leurs messages. Les versions en langue étrangère du système auront bien sûr des messages différents. Si le code d'erreur comporte une chaîne d'explication intégrée, cette chaîne sera renvoyée et le registre B sera mis à zéro. S'il n'y a pas de chaîne d'explication, une chaîne de la forme: "Erreur système 194" ou "Erreur utilisateur 45" sera renvoyée et le registre B ne sera pas modifié. (Les erreurs système sont celles situées dans la plage 40h ... FFh et les erreurs utilisateur sont 00h ... 3Fh.)

Entrée : B = Code d'erreur à expliciter

DE = Pointeur vers tampon de 64 octets

Sortie : A = 0

B = 0 (ou bien identique)

DE = Rempli avec le message d'erreur

Compatibilité : MSX-DOS 2 (même pendant le traitement d'une erreur).

## **Fonction 67h** - Format A Disk (\_FORMAT)

Rôle : Formater un disque. Cette fonction n'est présente que pour la commande "FORMAT" mais, il est possible de l'utiliser dans vos programmes (avec précaution).

Entrée : A = 0 pour renvoyer l'adresse du texte qui propose le choix de formatage dans HL.

1~9 pour formater comme indiqué par la ligne de texte correspondante.

0AH...FDH => illegal

FEH, FFH pour seulement re-écrire le secteur de boot sans formater.

HL = Adresse du cache (seulement si A était 1~9)

DE = Taille du cache (seulement si A était 1~9)

Sortie : A = Code d'erreur. (0 si pas d'erreur)

B = Slot où se trouve le texte (seulement si A était à 0 en entrée)

HL = Adresse du texte qui propose le choix de formatage (seulement si A était à 0 en entrée)

Si A = 0, les registres B et HL renvoient respectivement le numéro d'emplacement et l'adresse d'une chaîne de caractères ASCII spécifiant le choix des formats disponibles. Une erreur ".IFORM" sera renvoyée si ce disque ne peut pas être formaté (par exemple, le disque RAM). Normalement, la chaîne sera lue à l'aide de la routine "RDSLT" et affichée à l'écran suivie du message "?". L'utilisateur spécifie ensuite un choix "1" ... "9"

et ce choix est renvoyé à la fonction "formater", après un message d'avertissement approprié, pour formater le disque. Si A = 0, dans certains cas, zéro est renvoyé dans HL. Cela signifie qu'il n'y a qu'un seul type de format et qu'aucune invite n'est requise. Il n'y a aucun moyen de savoir à quel format de disque un choix particulier fait référence, car cela dépend du pilote de disque en question.

Si A = 01h ... 09h, ceci est interprété comme un choix de format et un disque sera formaté dans le disque spécifié sans autre invite. Register HL et DE doivent spécifier une zone tampon à utiliser par le pilote de disque. Il n'existe aucun moyen de savoir quelle doit être la taille de ce tampon, il est donc préférable de le rendre aussi grand que possible. Si la mémoire tampon dépasse les limites de la page, cette fonction sélectionne la plus grande partie de celle-ci, qui se trouve sur une page, à transmettre au pilote de disque. De nombreux pilotes de disque n'utilisent pas du tout ce tampon.

Si A = FFh, le disque ne sera pas réellement formaté, mais un nouveau secteur de démarrage lui sera attribué pour en faire un véritable disque MSX-DOS 2. Ceci est conçu pour mettre à jour les anciens disques MSX-DOS 1.0 pour qu'ils aient un identifiant de volume et permettre ainsi la vérification et l'annulation complètes du disque autorisé par MSX-DOS 2. Le cas A = FEh est identique à A = FFh, à la différence que seuls les paramètres du disque sont mis à jour correctement et que l'ID de volume n'écrase pas le programme de démarrage. De plus, certaines implémentations de MSX-DOS 1.0 placent un secteur de démarrage incorrect sur le disque et ces disques ne peuvent pas être utilisés par MSX-DOS 2 tant qu'ils n'ont pas été corrigés par cette fonction.

La fonction "nouveau secteur de démarrage" est principalement destinée au programme utilitaire "FIXDISK", mais peut être utilisée par d'autres programmes s'ils le jugent utile. S'il est utilisé, un appel de fonction "get format choice" (A = 0) doit être effectué en premier. S'il renvoie une erreur (généralement ".IFORM"), l'opération doit être abandonnée car il s'agit d'un disque qui n'aime pas. être formaté et le disque pourrait être endommagé par cette fonction.

Compatibilité : MSX-DOS 2.

### **Fonction 68h** - Create Or Destroy Ram Disk (\_RAMD)

Rôle : Créer/Supprimer le disque en RAM.

Entrée : B = 0 pour supprimer le disque en RAM.

1~254 pour crée un nouveau disque en RAM. La valeur de B indique le nombre de segments à allouer pour créer le disque.

255 pour renvoyer la taille du disque en RAM.

Sortie : A = Code d'erreur. (0 si pas d'erreur).

Le code d'erreur 0BCh (.RAMDX) sera renvoyée s'il on tente de créer un disque lorsqu'il y en a un déjà présent, mais pas si l'on tente de supprimer un disque non présent RAM.

Si les segments de RAM libres sont insuffisants pour créer un disque de la taille demandée, le disque sera créé avec autant de segment que possible mais aucune erreur ne sera renvoyée.

B = Nombre de segments du Memory Mapper (16k de RAM) alloués au disque en RAM.  
Si B = 0, c'est qu'il n'y a pas de disque en RAM.

Notez qu'une partie de la RAM est utilisée pour les tables d'allocation de fichiers et la racine des dossiers. La taille du disque RAM indiquée par la commande "DIR" ou



"CHKDSK" sera donc légèrement inférieure à la quantité totale de RAM utilisée. La lettre de disque "H:" est toujours affectée à la RAM, quel que soit le nombre de disques présents dans le système.

Compatibilité : MSX-DOS 2.

### **Fonction 69h** - Allocate Sector Buffers (\_BUFFER)

Rôle : Allocation des caches des secteurs.

Si B = 0, cette fonction renvoie simplement le nombre de mémoires tampons de secteurs actuellement allouées. Si B <> 0, cette fonction tentera d'utiliser ce nombre de tampons de secteur (doit toujours être au moins 2). S'il ne peut pas en affecter autant qu'il est demandé, il en affectera autant que possible et renverra le numéro dans le registre B mais ne renverra pas d'erreur. Le nombre de zones tampons de secteur peut être réduit ou augmenté.

Les tampons de secteur sont alloués dans un segment de 16 ko de RAM en dehors des 64 ko normaux, de sorte que le nombre de tampons ne diminue pas la taille du TPA. Cependant, le nombre de tampons affecte l'efficacité, car avec plus de tampons, plus de FAT et de secteurs d'annuaire peuvent rester résidents. Le nombre maximum de tampons sera d'environ 20.

Entrée : B = 0 => retourne le nombre de tampons, sinon nombre de tampons requis

Sortie : A = Code d'erreur. (0 si pas d'erreur)

B = Nombre actuel de tampons

Compatibilité : MSX-DOS 2.

### **Fonction 6Ah** - Logical Drive Assignment (\_ASSIGN)

Rôle : Affectation d'un lecteur logique.

Cette fonction contrôle l'affectation des unités logiques à physiques. Il est principalement destiné à la commande "ASSIGN", bien que les programmes utilisateur veuillent l'utiliser pour traduire les numéros de disque logique en numéros de disque physique.

Si B et D sont tous deux nuls, une nouvelle affectation sera créée. Si le registre B est différent de zéro et que le registre D est zéro, toute affectation de l'unité logique spécifiée par B sera annulée. Si les registres B et D sont tous deux nuls, toutes les assignations seront annulées. Si le registre B est différent de zéro et que le registre D est FFh, l'affectation actuelle de l'unité logique spécifiée par le registre B sera simplement renvoyée dans le registre D.

Tous les disques utilisés dans les différents appels de fonction, y compris les noms de disque dans les chaînes et les numéros de disque en tant que paramètres d'appels de fonction, sont des disques logiques. Cependant, le numéro de disque transmis aux routines d'erreur de disque est un disque physique. Par conséquent, si "ASSIGN" a été utilisé, il peut être différent du disque logique correspondant.

Entrée : B = Numéro du disque logique (1=A: etc)

D = Numéro du disque physique (1=A: etc)

Sortie : A = Code d'erreur. (0 si pas d'erreur)



D = Numéro du disque physique (1=A: etc)

Compatibilité : MSX-DOS 2 (même pendant le traitement d'une erreur).

### **Fonction 6Bh** - Get Environment Item (\_GENV)

Rôle : Obtenir une variable d'environnement.

Cette fonction donne le contenu de la variable d'environnement dont le nom est spécifié à l'adresse pointé par le registre HL. Une erreur ".IENV" est renvoyée si la chaîne de nom n'est pas valide. Si la variable d'environnement n'a pas été créée, une chaîne vide sera renvoyée dans le cache. S'il elle présente, son contenu sera copiée dans le cache. Si le cache est trop petit, le contiendra seulement une partie de la variable sans code de fin (0) et une erreur ".ELONG" sera renvoyée. Un cache de 255 octets sera toujours suffisamment grand car les variable ne peuvent pas être plus longues (code de fin inclut).

Entrée : B = Taille du cache

DE = Pointeur vers le cache

HL = Nom de la chaîne ASCII

Sortie : A = Code d'erreur. (0 si pas d'erreur)

DE ne change pas. Le premier octet du cache sera 0 si la variable est vide ou n'existe pas.

Compatibilité : MSX-DOS 2.

### **Fonction 6Ch** - Set Environment Item (\_SENV)

Rôle : Définir une variable d'environnement.

Cette fonction définit un nouvel élément d'environnement. Si la chaîne de nom n'est pas valide, une erreur ".IENV" est renvoyée. Sinon, la chaîne de valeur est vérifiée et une erreur ".ELONG" renvoyée si sa longueur dépasse 255 caractères ou une erreur ".NORAM" si la mémoire est insuffisante. pour stocker le nouvel article. Si tout va bien, tout élément ancien de ce nom est supprimé et le nouvel élément est ajouté au début de la liste des environnements. Si la chaîne de valeur est null, l'élément d'environnement sera supprimé.

Entrée : DE = Chaîne de valeur ASCII

HL = Chaîne ASCII

Sortie : A = Code d'erreur. (0 si pas d'erreur)

Compatibilité : MSX-DOS 2 (même pendant le traitement d'une erreur).

### **Fonction 6Dh** - Find Environment Item (\_FENV)

Rôle : Trouver une variable d'environnement.

Cette fonction permet de connaître les éléments d'environnement actuellement définis. Le numéro d'article dans le registre DE identifie quel article de la liste doit être trouvé (le premier article correspond à DE = 1). S'il existe un numéro d'élément <DE>, la chaîne de nom de cet élément sera copiée dans le tampon pointé par HL. Si la mémoire tampon est

trop petite, le nom sera tronqué sans fin, et une erreur ".ELONG" sera renvoyée. Un tampon de 255 octets ne sera jamais trop petit. S'il n'y a pas de numéro d'élément <DE>, une chaîne nulle sera renvoyée, puisqu'un élément ne peut jamais avoir une chaîne de nom nulle.

Entrée : DE = Numéro d'article d'environnement  
HL = Pointeur sur le tampon pour la chaîne de nom

Sortie : A = Code d'erreur. (0 si pas d'erreur)  
HL = Conservé, cache rempli

Compatibilité : MSX-DOS 2 (même pendant le traitement d'une erreur).

### **Fonction 6Eh** - Get/Set Disk Check Status (\_DSKCHK)

Rôle : Obtenir/Définir le statut de vérification du disque.

La variable de vérification du disque contrôle si le système va vérifier à nouveau le secteur de démarrage d'un disque pour voir s'il a été modifié à chaque fois qu'un accès à un descripteur de fichier, un bloc info fichier ou un FCB est effectué. S'il est activé, il sera impossible d'accéder accidentellement au mauvais disque en changeant un disque au milieu d'une opération. Dans le cas contraire, cela sera possible et risque de corrompre le disque. Selon le type d'interface de disque, l'activation de cette fonctionnalité peut entraîner des coûts supplémentaires, bien qu'avec de nombreux types de disques (ceux dotés d'un matériel de détection de changement de disque explicite), cela ne fera aucune différence et la sécurité supplémentaire en vaut la peine.

Entrée : A = 00H => obtenir l'état de vérification du disque, 01H => définir le statut de vérification du disque  
B = 00H => activé (uniquement si A = 01H), FFH => désactiver (uniquement si A = 01H)

Sortie : A = Code d'erreur. Si A = 0, la valeur actuelle de la variable de vérification du disque est renvoyée dans le registre B. Si A = 01h, la variable est définie sur la valeur du registre B. Une valeur de 00h signifie que la vérification du disque est activée et une valeur non nulle signifie qu'il est désactivé. L'état par défaut est activé.  
B = Paramètre actuel de vérification du disque

Compatibilité : MSX-DOS 2 (même pendant le traitement d'une erreur).

### **Fonction 6Fh** - Get MSX-DOS Version Number (\_DOSVER)

Rôle : Obtenir le numéro de version du MSX-DOS.

Cette fonction permet à un programme de déterminer quelle version de MSX-DOS est installée. Deux numéros de version sont renvoyés, l'un dans BC pour le noyau MSX-DOS en ROM et l'autre dans DE pour le fichier système MSXDOS2.SYS. Ces deux numéros de version sont des valeurs au format BCD avec le numéro de version principale dans l'octet de poids fort et le numéro de version mineure à deux chiffres dans l'octet de poids faible. Par exemple, la version 2.31 du système sera notée 0234h dans les registres.

Pour assurer la compatibilité avec MSX-DOS 1.0, la procédure suivante doit toujours être suivie. Premièrement, s'il y a une erreur (A <> 0), c'est que le MSX-DOS n'est pas installé. Ensuite, examinez le registre B. S'il est inférieur à 2, le MSX-DOS est antérieur

à la v2.00 et dans ce cas, les registres C et DE doivent être ignorés. Si le registre B est supérieur ou égal à 2, les registres BC et DE sont définis comme suit.

Entrée : A = Rien.

Sortie : A = Code d'erreur. (Noujours 0)

BC = Version du kernel. (B = version principale, C = version mineure)

DE = Version du MSXDOS2.SYS. (D = version principale, E = version mineure)

Compatibilité : MSX-DOS 2 (même pendant le traitement d'une erreur).

## **Fonction 70h - Get/Set Redirection Status (\_REDIR)**

Rôle : Obtenir/Définir le statut de redirection.

Cette fonction est principalement fournie pour les routines d'erreur de disque et les autres entrées / sorties de caractères qui doivent toujours aller à la console, quelle que soit la redirection. Lorsque les fonctions de caractère CP/M (fonctions 01h à 0Bh) sont utilisées, elles font normalement référence à la console. Toutefois, si les descripteurs de fichier d'entrée ou de sortie standard (descripteurs de fichier 0 et 1) ont été fermés et rouverts dans un fichier de disque, les fonctions de caractères CP/M seront également transférées dans le fichier de disque. Cependant, certaines sorties, telles que les erreurs de disque, doivent toujours apparaître à l'écran.

Cette fonction permet d'annuler temporairement une telle redirection en appelant cette fonction avec A = 1 et B = 0. Cela garantira que toutes les E/S ultérieures de la console CP/M iront à la console et renverra également le paramètre précédent afin que celui-ci puisse être restauré par la suite. Le système est dans un état quelque peu instable lorsque l'état de redirection a été modifié de la sorte et que de nombreux appels de fonction réinitialisent la redirection sur son état réel remplaçant cette fonction. En général, tout appel de fonction manipulant des descripteurs de fichier, tels que "ouvrir", "fermer", "dupliquer", etc., réinitialisera l'état de redirection. L'effet de cette fonction est donc purement temporaire.

Entrée : Rien

Sortie : A = Code d'erreur. (0 si pas d'erreur)

B = Etat de redirection avant commande

b0 = 1 lorsque l'entrée est redirigée

b1 = 1 lorsque la sortie est redirigée

Compatibilité : MSX-DOS 2 (même pendant le traitement d'une erreur).

Il ne reste que la description des codes d'erreur que MSX-DOS 1 transmet à la routine de traitement des erreurs. Tout d'abord, le registre A contient le numéro de disque. C'est le disque qui a généré l'erreur. Le registre C contiendra une indication de l'erreur, voir le tableau suivant:

b7 = 1: mauvaise grasse

b0 = 0: action de lecture

b0 = 1: action d'écriture

b3 b2 b1

0 0 0: protégé en écriture

0 0 1: Non lu (disque hors ligne)

0 1 0: erreur CRC

0 1 1: erreur de recherche

1 0 0: enregistrement non trouvé (secteur non trouvé)

1 0 1: erreur d'écriture

1 1 0: Autre

Lors du retour du gestionnaire d'erreurs, le registre C devrait spécifier l'action à prendre:

0 = ignorer

1 = réessayer

2 = Abandonner

Lorsque l'option 2 (abandon) est choisie, il passe directement à la routine d'abandon que nous avons précédemment utilisée.

C'est tout ce qu'on peut en dire. Encore une chose: faites attention à l'endroit où vous mettez la routine de traitement des erreurs. Cette routine est appelée par le DiskROM et il est possible qu'elle ne puisse pas atteindre une routine de la page 1 (4000..7FFF). Toutefois, cela n'est pas un problème pour la routine d'abandon, car à ce stade, la mémoire TPA complète a été réactivée.

Pour la gestion des erreurs MSX-DOS 2, veuillez vous reporter au manuel de l'application MSX-DOS.

### **Fonction 71h** - Get/Set fast STROUT mode (\_FOUT)

Rôle : Activer ou désactiver l'affichage rapide pour la [fonction 08h](#) et la 72h.

Entrée : A = 00h pour indiquer si l'affichage rapide est activé ou pas, autrement 01h pour régler.  
B = FFh pour activer l'affichage rapide, 0 pour le désactiver. (sans effet si A = 00h)

Sortie : A = Code d'erreur. (0 si pas d'erreur).  
B = FFh si l'affichage rapide est activé.

Compatibilité : Nextor

### **Fonction 72h** - Print a zero-terminated string (\_ZSTROUT)

Rôle : Afficher une chaîne de caractères à l'écran.

Entrée : DE = Adresse du premier caractère de la chaîne qui doit se terminer par le code 00h et ne peut pas faire plus de 511 caractères lorsque l'affichage rapide est activé par la [fonction 71h](#).

Sortie : A = 0

Compatibilité : Nextor

### **Fonction 73h** - Read absolute sectors from drive (\_RDDRIV)

Rôle : Lire un ou des secteurs d'un disque. Contrairement à la [fonction 2Fh](#), cette fonction est capable de lire les secteurs sur un disque formaté FAT16, même si il n'y a pas de fichiers système.

Entrée : A = Numéro du disque (0 = A:, etc)  
B = Nombre de secteurs à lire  
HLDE = Numéro du premier secteur

Sortie : A = Code d'erreur. (0 si pas d'erreur)

Compatibilité : Nextor

### **Fonction 74h** - Write absolute sectors to drive (\_WRDRV)

Rôle : Écrire un ou des secteurs sur un disque. Contrairement à la [fonction 30h](#), cette fonction est capable d'écrire des secteurs sur un disque formaté FAT16, même si il n'y a pas de fichiers système.

Entrée : A = Numéro du disque (0 = A:, etc)  
B = Nombre de secteurs à écrire  
HLDE = Numéro du premier secteur

Sortie : A = Code d'erreur. (0 si pas d'erreur)

Compatibilité : Nextor

### 13.9 Traiter les erreurs du MSX-DOS 1 dans vos programmes

Il est 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. La variable ERRADR (0F323h) contient une adresse qui pointe vers une autre adresse qui à son tour contient l'adresse de la routine le traitement d'erreur des disques du MSX-DOS 1. Modifiez cette adresse afin de détourner la routine du MSX-DOS 1 vers votre propre routine.
2. Votre routine pourra envoyer les messages adéquats sachant que l'accumulateur contient le numéro du disque, et le registre C contient les informations suivantes.

| Bit 3 | Bit 2 | Bit 1 | Bit 0 | Type d'erreur                                    |
|-------|-------|-------|-------|--------------------------------------------------|
| 0     | 0     | 0     | 1     | Protégée contre l'écriture (« Write protected ») |
| 0     | 0     | 1     | 0/1   | Disque pas prêt (« Disk offline »)               |
| 0     | 1     | 0     | 0/1   | Erreur de CRC (« CRC error »)                    |
| 0     | 1     | 1     | 0/1   | Erreur de recherche (« Seek error »)             |
| 1     | 0     | 0     | 0/1   | Enregistrement non trouvé (« Record not found ») |
| 1     | 0     | 1     | 0     | Média non compatible (« Unsupported media »)     |
| 1     | 0     | 1     | 1     | Erreur d'écriture (« Write error »)              |
| 1     | 1     | 0     | 0/1   | Autre erreur                                     |

| Bit 7 | Type d'erreur                       |
|-------|-------------------------------------|
| 0     | Pas de problème détecté dans la FAT |
| 1     | FAT endommagée (« Bad FAT »)        |

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).

Lorsque c'est l'option « Annuler » qui est spécifiée, le MSX-DOS saute à l'adresse 0F1E6h qui contient une instruction de saut (JP) pour revenir à l'invite. Vous pouvez changer cela en remplaçant l'adresse du saut à 0F1E7h par l'adresse de votre routine.

0F1E6h -> Gestionnaire d'abandon

Le gestionnaire d'abandon existe pour intercepter les erreurs de pile. Si Ignorer ou Réessayer est spécifié, une tentative réussie renvoie au site d'appel de la fonction BDOS et, lorsqu'une autre erreur se produit, la routine de traitement des erreurs est appelée à nouveau.

Ensuite, la pile doit être préservée. Cela peut être fait comme suit, en introduisant une routine de

wrapper universelle pour invoquer le BDOS au programme:

```
bdos:    ld      (errStack),sp
        call    5
        ret
```

À ce stade, les routines de traitement des erreurs doivent déjà avoir été configurées et, en cas d'abandon, elles doivent passer à la routine suivante:

```
bdosAbort: ld  sp, (errStack)
          ret
```

Cela retournera à la routine qui a initialement appelé la fonction BDOS. Tout ce qui reste est un mécanisme qui vous permet de vérifier les erreurs. Cela peut être fait en définissant d'abord une variable avant d'utiliser la méthode ci-dessus pour retourner.

Lorsque l'application transfère ces deux adresses, il est important que les valeurs originales soient préservées et restaurées à la fin du programme.

Une dernière chose, soyez prudent lorsque vous détournez la routine de traitement des erreurs. Cette routine est appelée par le Disk-ROM, donc votre routine ne peut se trouver sur la plage mémoire de 4000h à 7FFFh. Toutefois, la routine d'abandon n'a pas cette limite, car à ce stade, la Disk-ROM n'est plus sélectionnée.

Voici un petit programme de principe en assembleur :

```
; Disk error trap

BDOS      equ      0005h
ERRADR    equ      0F323h

EBDOS:
    push    hl

    ld      hl, (ERRADR)
    ld      (VESAV),hl      ; Stocke ERRADR
    ld      hl,ERRV
    ld      (ERRADR),hl     ; Met l'adresse de votre routine

    ld      hl,BDSTAT
    ld      (hl),0

    pop     hl

    ld      (SPSAV),sp      ; Stocke the register SP

    call    BDOS
```

```

        JR      BACK

ERROR:
        ld      sp, (SPSAV)      ; Restore le register SP
        ld      a, c
        ld      (BDSTAT), a
BACK:
        push    hl
        ld      hl, (VESAV)
        ld      (ERRADR), hl    ; Restore ERRADR
        pop     hl
        ret

ERRV:   dw      ERROR

SPSAV:  ds      2
VESAV:  ds      2
BDSTAT: ds      2

```



### 13.10 Codes d'erreur des fonctions du MSX-DOS 2

La liste suivante contient toutes les erreurs qui peuvent se produire lors d'un appel à une fonction du MSX-DOS. Pour les détails, veuillez consulter la description de la fonction avec laquelle l'erreur ce produit.

#### **0B8h - Invalid sub-function number (.ISBFN)**

Le numéro de sous-fonction transmis à la fonction 4Bh (IOCTL) n'est pas valide.

#### **0B9h - (.EOL)**

Erreur interne qui n'aurait jamais dû se produire.

#### **0BAh - File handle has been deleted (.HDEAD)**

Le fichier associé au gestionnaire de fichier a été supprimé. Par conséquent, il ne peut plus être utilisé.

#### **0BBh - RAM disk does not exist (.NRAMD)**

Il y a eu demande de suppression du disque en RAM alors qu'il n'a pas été créé. Le code d'erreur 0DBh (.IDRV) sera renvoyé si une fonction tente d'accéder au disque en RAM en alors qu'il est inexistant.

#### **0BCh - RAM disk (drive H:) already exists (.RAMDX)**

Se produit si la fonction 68h (\_RAMD) essaie de créer un disque en RAM alors qu'il a déjà été créé.

#### **0BDh - Invalid time (.ITIME)**

Les paramètres pour régler l'heure ne sont pas valides.

#### **0BEh - Invalid date (.IDATE)**

Les paramètres pour régler la date ne sont pas valides.

#### **0BFh - Environment string too long (.ELONG)**

Le nom ou le contenu de la variable d'environnement est trop long. La longueur maximale autorisée est de 255 octets.

#### **0C0h - Invalid environment string (.IENV)**

Le nom la variable d'environnement contient un caractère non valide.

#### **0C1h - Invalid device operation (.IDEV)**

L'opération demandée (une recherche ou un déplacement par exemple) n'est pas faisable avec le gestionnaire de fichier de périphérique ou le bloc d'informations de fichier (FIB) spécifié.

#### **0C2h - File handle not open (.NOPEN)**

Le gestionnaire de fichier spécifié n'est pas ouvert.

#### **0C3h - Invalid file handle (.IHAND)**

Le numéro du gestionnaire de fichier spécifié est supérieur au nombre maximal autorisé.

#### **0C4h - No spare file handles (.NHAND)**

Attribution d'un gestionnaire de fichier alors qu'ils sont déjà tous utilisés. 64 gestionnaires de fichiers sont disponibles dans la version actuelle.

#### **0C5h - Invalid process id (.IPROC)**

Le numéro d'identification du processus transmis à la fonction 61h (\_JOIN) n'est pas valide.

### **0C6h - File access violation (.ACCV)**

Tentative de lecture ou d'écriture dans un gestionnaire de fichier qui a été déjà ouvert avec le bit d'accès approprié à 1. Certains gestionnaires de fichiers standard sont ouverts en mode lecture seule ou écriture seule.

### **0C7h - End of file (.EOF)**

Tentative de lecture à partir d'un fichier lorsque le pointeur de fichier se trouve déjà à la fin du fichier ou au-delà.

### **0C8h - File allocation error (.FILE)**

La chaîne de cluster d'un fichier a été corrompue. Utilisez CHKDSK pour récupérer autant de fichiers que possible.

### **0C9h - Cannot transfer above 64K (.OV64K)**

La zone de transfert du disque s'étend au-delà de 0FFFFh.

### **0CAh - File already in use (.FOPEN)**

Tentative de supprimer, renommer, déplacer ou modifier les attributs ou la date et l'heure d'un fichier pour lequel un gestionnaire de fichier est déjà ouvert, autrement qu'en utilisant son propre gestionnaire de fichier.

### **0CBh - File exists (.FILEX)**

Attempt to create a sub-directory of the same name as an existing file. Files are not automatically deleted when creating sub-directories.

### **0CCh - Directory exists (.DIRX)**

Attempt to create a file or sub-directory of the same name as an existing sub-directory. Sub-directories are not automatically deleted.

### **0CDh - System file exists (.SYSX)**

Attempt to create a file or sub-directory of the same name as an existing system file. System files are not automatically deleted.

### **0CEh - Invalid . or .. operation (.DOT)**

Attempt to do an illegal operation on the "." or ".." entries in a sub-directory, such as rename or move them.

### **0CFh - Invalid attributes (.IATTR)**

Can result from an attempt to change a file's attributes in an illegal way, or trying to do an operation on a file which is only possible on a sub-directory. Also results from illegal use of volume name fileinfo blocks.

### **0D0h - Directory not empty (.DIRNE)**

Attempt to delete a sub-directory which is not empty.

### **0D1h - Read only file (.FILRO)**

Attempt to write to or delete a file which has the "read only" attribute bit set.

### **0D2h - Invalid directory move (.DIRE)**

Results from an attempt to move a sub-directory into one of its own descendants. This is not allowed as it would create an isolated loop in the directory structure.

### **0D3h - Duplicate filename (.DUPF)**

Results from "rename" or "move" if the destination filename already exists in the destination

directory.

#### **0D4h - Disk full (.DKFUL)**

Usually results from a write operation if there was insufficient room on the disk for the amount of data being written. May also result from trying to create or extend a sub-directory if the disk is completely full.

#### **0D5h - Root directory full (.DRFUL)**

Returned by "create" or "move" if a new entry is required in the root directory and it is already full. The root directory cannot be extended.

#### **0D6h - Directory not found (.NODIR)**

Returned if a directory item in a drive/path/file string could not be found.

#### **0D7h - File not found (.NOFIL)**

Can be returned by any function which looks for files on a disk if it does not find one. This error is also returned if a directory was specified but not found. In other cases, .NODIR error (see below) will be returned.

#### **0D8h - Pathname too long (.PLONG)**

Can be returned by any function call which is given an [ASCIIZ](#) drive/path/file string. Indicates that the complete path being specified (including current directory if used) is longer than 63 characters.

#### **0D9h - Invalid pathname (.IPATH)**

Can be returned by any function call which is given an [ASCIIZ](#) drive/path/file string. Indicates that the syntax of the string is incorrect in some way.

#### **0DAh - Invalid filename (.IFNM)**

A filename string is illegal. This is only generated for pure filename strings, not drive/path/file strings.

#### **0DBh - Invalid drive (.IDRV)**

A drive number parameter, or a drive letter in a drive/path/file string is one which does not exist in the current system.

#### **0DCh - Invalid MSX-DOS call (.IBDOS)**

An MSX-DOS call was made with an illegal function number. Most illegal function calls return no error, but this error may be returned if a "get previous error code" function call is made.

#### **0DEh - Not enough memory (.NORAM)**

MSX-DOS has run out of memory in its 16k kernel data segment. Try reducing the number of sector buffers or removing some environment strings. Also occurs if there are no free segments for creating the RAMdisk.

#### **0DFh - Internal error (.INTER)**

Should never occur.

### 13.11 Zone fixe de travail, Hooks et des variables du MSX-DOS1 et du Disk-BASIC

Cette zone contient aussi des Hooks spécifiques au MSX-DOS.

| Adresse | Nom    | Long. | Fonction                                                                                                                                                                                                                                                                              |
|---------|--------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0F1C9h  | STROUT | 14    | Routine pour afficher une chaine de caractères se terminant par le caractère « \$ » (024h).<br>Entrée : DE = Adresse du début de la chaine.                                                                                                                                           |
| 0F1D9h  | XFER   | 10    | Routine de transfert de données vers la Main-RAM                                                                                                                                                                                                                                      |
| 0F1E2h  | WBOOT  | 6     | Routine pour interrompre un programme en cas d'erreur.                                                                                                                                                                                                                                |
| 0F1E8h  |        | 12    | Routine d'appel à une routine en Main-RAM sur la plage 1 lorsque la Disk-ROM est sélectionnée.<br>Entrée : Adresse contenue dans HL = Adresse de la routine à appeler.                                                                                                                |
| 0F1F4h  |        | 3     | Routine de validation du nom de fichier du FCB.<br>Entrée : HL = Adresse du premier caractère du nom de fichier.<br>Output : F = CF est mis à 1 si le nom n'est pas valide.                                                                                                           |
| 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 du fichier trouvé au périphérique.                                                                                                                                                                                                                                                |
| 0F213h  |        | 3     | Extension du nom du fichier trouvé au périphérique.                                                                                                                                                                                                                                   |
| 0F216h  |        | 1     | Périphérique actuelle. PRN = -5, LST = -4, ..., CON = -1.                                                                                                                                                                                                                             |
| 0F221h  |        | 2     | Date du fichier trouvé au périphérique.                                                                                                                                                                                                                                               |
| 0F223h  |        | 2     | Heure du fichier trouvé au périphérique.                                                                                                                                                                                                                                              |
| 0F22Bh  |        | 12    | Tableau contenant le nombre de jours de chaque mois de l'année sans tenir compte des années bissextiles.<br>Note : Une année est bissextile lorsqu'elle est divisible par 4 mais pas par 100, à moins qu'elle soit divisible par 400.                                                 |
| 0F237h  | BUFINP | 4     | Zone de travail de l'entrée d'une ligne de commande.<br>BUFINP = Colonne de la fin de la ligne. (Position du curseur)<br>BUFINP+1 = Colonne du début de la ligne.<br>BUFINP+2 = Indicateur d'insertion. (1 = Mode insertion)<br>BUFINP+3 = Indicateur de message secret. (0 = Normal) |

|        |               |   |                                                                                                                                                                                                                                                      |
|--------|---------------|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0F23Bh |               | 1 | Indicateur pour envoyer les caractères vers l'imprimante.<br>0 = Ne pas envoyer ; Autre valeur = Caractères vers l'imprimante.                                                                                                                       |
| 0F23Dh |               | 2 | Adresse actuelle du DTA (0080h par défaut sous DOS).                                                                                                                                                                                                 |
| 0F23Fh |               | 4 | Numéro du premier secteur du fichier dans lequel il y a eu le dernier accès.                                                                                                                                                                         |
| 0F243h |               | 2 | Pointeur à l'adresse du DPB du disque actuel.                                                                                                                                                                                                        |
| 0F245h |               | 1 | Numéro relatif du secteur du fichier suivant sur le disque actuel. Utilisé par les routines SEARCH FIRST et NEXT. (Ajouter le numéro du premier secteur du fichier dans lequel il y a eu le dernier accès pour connaître le numéro du secteur réel.) |
| 0F246h |               | 1 | Numéro du disque sur lequel il y a eu le dernier accès à un fichier. ( 0 = « A: », 1 = « B: », etc, 0FFh = Aucun accès)                                                                                                                              |
| 0F247h |               | 1 | Numéro du disque par défaut. ( 0 = « A: », 1 = « B: », etc, 0FFh = Pas de disque)                                                                                                                                                                    |
| 0F248h |               | 1 | Jour. Valeur stockée lorsque la commande DATE ou TIME est utilisée. (MSX-DOS)                                                                                                                                                                        |
| 0F249h |               | 1 | Mois. Valeur stockée lorsque la commande DATE ou TIME est utilisée. (MSX-DOS)                                                                                                                                                                        |
| 0F24Ah |               | 1 | Année. 0 = 1980, 1 = 1981, 2 = 1982, etc. Valeur stockée lorsque la commande DATE ou TIME est utilisée. (MSX-DOS)                                                                                                                                    |
| 0F24Ch |               | 1 | Heure. Valeur stockée lorsque la commande DATE ou TIME est utilisée. (MSX-DOS)                                                                                                                                                                       |
| 0F24Dh |               | 1 | Minute. Valeur stockée lorsque la commande DATE ou TIME est utilisée. (MSX-DOS)                                                                                                                                                                      |
| 0F24Eh |               | 1 | Jour de la semaine. 0 = Dimanche, 1 = Lundi, 2 = Mardi, etc. Valeur stockée lorsque la commande DATE ou TIME est utilisée. (MSX-DOS)                                                                                                                 |
| 0F24Fh | H.PROMPT      | 3 | Hook appelé avant l'affichage du message « Insert diskette for drive » de l'émulation d'un deuxième disque. Lors de l'appel à ce Hook, le registre A contient la lettre du nom de disque 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 qui vérifie le nom de périphérique dans le nom de fichier.                                                                                                                                                        |
| 0F258h | H.SEARCHDIR   | 3 | Hook appelé au début de la routine de recherche du fichier suivant. (Utile pour changer 0F2DCh afin de trouver aussi les fichiers cachés.)                                                                                                           |
| 0F25Bh |               | 3 | Hook appelé au début de la routine d'obtention du fichier suivant.                                                                                                                                                                                   |
| 0F25Eh | H.NEXTDIRSECT | 3 | Hook appelé au début de la routine de passage au fichier suivant.                                                                                                                                                                                    |

|        |          |   |                                                                                                                       |
|--------|----------|---|-----------------------------------------------------------------------------------------------------------------------|
| 0F261h |          | 3 | Hook appelé au début de la routine de vérification du FCB du disque et du nom de fichier.                             |
| 0F264h | H.OPEN   | 3 | Hook appelé avant de vérifier la taille du fichier que l'on ouvre.                                                    |
| 0F267h |          | 3 | Hook appelé au début de la routine d'obtention de la dernière FAT pour vérifier si le disque a changé ou pas.         |
| 0F26Ah | H.GETDPB | 3 | Hook appelé au début de la routine d'obtention du DPB du disque actuel.                                               |
| 0F26Dh | H.CLOSE  | 3 | Hook appelé avant la fermeture un fichier.                                                                            |
| 0F270h | H.RDABS  | 3 | Hook appelé au début de la routine de lecture de secteur(s). Utilisé au début de la routine _RDABS du BDOS (2Fh).     |
| 0F273h | H.DSKERR | 3 | Hook appelé avant la routine traitement d'erreur lors d'accès au disque. (Lors d'un changement de disque par exemple) |
| 0F276h |          | 3 | Hook appelé au début de la routine d'écriture d'un fichier.                                                           |
| 0F279h | H.WRABS  | 3 | Hook appelé au début de la routine d'écriture de secteur(s). Utilisé au début de la routine _WRABS du BDOS (30h).     |
| 0F27Ch |          | 3 | Hook appelé au début de la routine interne de multiplication. (HL et BC = DE*BC)                                      |
| 0F27Fh |          | 3 | Hook appelé au début de la routine interne de division. (BC = BC/DE, HL = Reste)                                      |
| 0F282h |          | 3 | Hook appelé au début de la routine d'obtention du cluster absolu.                                                     |
| 0F285h |          | 3 | Hook appelé au début de la routine d'obtention du cluster absolu suivant.                                             |
| 0F288h |          | 3 | Hook appelé au début de la routine de lecture partielle de secteur.                                                   |
| 0F28Bh |          | 3 | Hook appelé au début de la routine d'écriture partielle de secteur.                                                   |
| 0F28Eh |          | 3 | Hook appelé au début de la routine de lecture d'un enregistrement sur un disque.                                      |
| 0F291h |          | 3 | Hook appelé en fin de la routine de lecture d'un enregistrement sur un disque.                                        |
| 0F294h |          | 3 | Hook appelé après la lecture d'un enregistrement sur un disque.                                                       |
| 0F297h |          | 3 | Hook appelé au début de la routine d'un enregistrement                                                                |
| 0F29Ah |          | 3 | Hook appelé au début de la routine de l'écriture d'un enregistrement sur un disque.                                   |
| 0F29Dh |          | 3 | Hook appelé en fin de la routine de l'écriture d'un enregistrement sur un disque.                                     |
| 0F2A0h |          | 3 | Hook appelé au début de la routine de calcul séquentiels de secteurs.                                                 |
| 0F2A3h |          | 3 | Hook appelé au début de la routine pour obtenir le numéro de secteur du cluster.                                      |
| 0F2A6h |          | 3 | Hook appelé au début de la routine d'allocation de chaîne dans la FAT.                                                |

|        |          |    |                                                                                                                                                                                                                                                                                                              |
|--------|----------|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0F2A9h |          | 3  | Hook appelé au début de la routine de réactualisation de chaîne dans la FAT.                                                                                                                                                                                                                                 |
| 0F2ACh | H.BUFINP | 3  | Hook appelé au début de la routine BUFINP qui ajoute une donnée au cache du clavier.                                                                                                                                                                                                                         |
| 0F2AFh | H.CONOUT | 3  | Hook appelé au début de la routine CONOUT qui vérifie les codes CTRL.                                                                                                                                                                                                                                        |
| 0F2B2h |          | 3  | Hook appelé au début de la routine de lecture de la date et de l'heure d'un fichier.                                                                                                                                                                                                                         |
| 0F2B5h |          | 3  | Hook appelé au début de la routine de calcul des années bissextiles.                                                                                                                                                                                                                                         |
| 0F2B8h | DIRENT   | 1  | Position du fichier ouvert sur le disque. (0FFh = Aucune)                                                                                                                                                                                                                                                    |
| 0F2B9h |          | 11 | Nom du fichier ouvert avec son extension (à 0F2C1h).                                                                                                                                                                                                                                                         |
| 0F2C4h |          | 1  | Attributs du fichier ouvert. Le format est le suivant :<br>Bit 0 = 1 pour fichier protégé contre l'écriture<br>Bit 1 = 1 pour fichier invisible<br>Bit 2 = 1 pour fichier du système<br>Bit 3~4 = 0<br>Bit 5 = 1 pour archive<br>Bit 7 = Si 1 alors les éléments avec un attribut à 1 pourront être ouverts. |
| 0F2C4h |          | 11 | Nom du second fichier avec son extension (rename)                                                                                                                                                                                                                                                            |
| 0F2CFh |          | 2  | Heure du fichier ouvert.                                                                                                                                                                                                                                                                                     |
| 0F2D1h |          | 2  | Date du fichier ouvert.                                                                                                                                                                                                                                                                                      |
| 0F2D3h |          | 2  | Premier cluster du fichier ouvert.                                                                                                                                                                                                                                                                           |
| 0F2D5h |          | 2  | Taille du fichier ouvert.                                                                                                                                                                                                                                                                                    |
| 0F2D0h |          | 12 | Sauvegarde temporaire pour 0F2B9h et 0F2C4h. (rename)                                                                                                                                                                                                                                                        |
| 0F2DCh |          | 1  | Si cet octet est à zéro, les attributs à 1 du fichier seront ignorés. (le bit 7 à 0F2C4h annule ceci lorsqu'il est à 1!)                                                                                                                                                                                     |
| 0F2DDh |          | 1  | Secteur relatif actuel dans le cluster                                                                                                                                                                                                                                                                       |
| 0F2DEh |          | 1  | Résultat de l'enregistrement.                                                                                                                                                                                                                                                                                |
| 0F2DFh |          | 1  | Indicateur du secteur relatif actuel dans le cluster (0 = Aucun)                                                                                                                                                                                                                                             |
| 0F2E0h |          | 1  | Indicateur de lecture. (0 = lu)                                                                                                                                                                                                                                                                              |
| 0F2E1h | DRIVE    | 1  | Numéro du disque en cours de lecture / écriture absolue (0 ~ 7).<br>0 = Disque « A », 1 = Disque « B », etc.                                                                                                                                                                                                 |
| 0F2E2h |          | 2  | Adresse de transfert pour effectuer un enregistrement.                                                                                                                                                                                                                                                       |
| 0F2E4h |          | 4  | Emplacement du premier enregistrement.                                                                                                                                                                                                                                                                       |
| 0F2E8h |          | 2  | Nombre d'enregistrement.                                                                                                                                                                                                                                                                                     |
| 0F2EAh |          | 2  | Cluster relatif du fichier actuel.                                                                                                                                                                                                                                                                           |
| 0F2ECh |          | 2  | Cluster du fichier actuel.                                                                                                                                                                                                                                                                                   |
| 0F2EEh |          | 2  | Secteur relatif de l'enregistrement.                                                                                                                                                                                                                                                                         |
| 0F2F0h |          | 2  | Cluster relatif en fin de fichier après un enregistrement.                                                                                                                                                                                                                                                   |

|        |           |   |                                                                                                                                                                                                                                                              |
|--------|-----------|---|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0F2F2h |           | 2 | Décalage du début dans le secteur pour effectuer un enregistrement.                                                                                                                                                                                          |
| 0F2F4h |           | 4 | Position du premier fichier pour effectuer un enregistrement.                                                                                                                                                                                                |
| 0F2F8h |           | 2 | Premier secteur à transmettre.                                                                                                                                                                                                                               |
| 0F2FAh |           | 2 | Dernier secteur à transmettre.                                                                                                                                                                                                                               |
| 0F2FCh |           | 2 | Nombre de secteur à transmettre.                                                                                                                                                                                                                             |
| 0F2FEh |           | 1 | Première entrée libre trouvée. (0FFh = Aucune)                                                                                                                                                                                                               |
| 0F2FFh |           | 1 | Indicateur de lecture (0) / écriture (1).                                                                                                                                                                                                                    |
| 0F300h |           | 2 | Adresse de la routine de gestion des erreurs de disque.                                                                                                                                                                                                      |
| 0F302h |           | 2 | Adresse de la routine gestionnaire d'annulation.                                                                                                                                                                                                             |
| 0F304h |           | 2 | Stockage temporaire du registre de la pile (SP).                                                                                                                                                                                                             |
| 0F306h |           | 1 | Numéro du disque sur lequel le DOS a démarré (1~8). 1 = « A: », 2 = « B: », etc.                                                                                                                                                                             |
| 0F307h |           | 2 | Sauvegarde du pointeur du premier FCB trouvé (utilisé pour la recherche suivante)                                                                                                                                                                            |
| 0F309h |           | 2 | Sauvegarde du pointeur du premier DPB trouvé (utilisé pour la recherche suivante)                                                                                                                                                                            |
| 0F30Bh |           | 2 | Sauvegarde du numéro de fichier trouvé dans le dossier actuel (FFh si non trouvé)                                                                                                                                                                            |
| 0F30Dh | RAWFLG    | 1 | Indicateur de vérification. 0 = Vérification terminée ; Autre valeur = Vérification en cours.                                                                                                                                                                |
| 0F30Eh |           | 1 | Indicateur du format de la date.<br>0 = AAMMJJ, 1= MMJJAA, 2= JJMMAA.                                                                                                                                                                                        |
| 0F30Fh | KANJTABLE | 4 | Deux paires de limites pour les premiers octets des caractères Shift-JIS.                                                                                                                                                                                    |
| 0F313h | DISKVE    | 1 | Version de la Disk-ROM. 0 = Disk-ROM v.1.x, 2xH = Disk-ROM v.2.x.                                                                                                                                                                                            |
| 0F323h | ERRADR    | 2 | Contient l'adresse de la routine de gestion des erreurs de disque. Si vous avez besoin de faire une routine pour gérer les erreurs, veuillez vous référer au paragraphe : « <a href="#">Traiter les erreurs du MSX-DOS 1 dans vos programmes</a> » page 502. |
| 0F325h | BREAKV    | 2 | Adresse de la routine de gestion de CTRL+C.                                                                                                                                                                                                                  |
| 0F327h |           | 5 | Hook appelé par la routine AUXINP.                                                                                                                                                                                                                           |
| 0F32Ch |           | 5 | Hook appelé par la routine AUXOUT.                                                                                                                                                                                                                           |
| 0F331h |           | 5 | Hook appelé par les fonctions BDOS.                                                                                                                                                                                                                          |
| 0F336h |           | 1 | Indicateur de touche pressée. Contient 0FFh lorsqu'une touche a été pressée ou, 03h si CTRL+STOP ont été pressées.                                                                                                                                           |
| 0F337h |           | 1 | Indicateur de touche pressée. Contient le code de la touche qui a été pressée ou, contient 03h si CTRL+STOP sont pressées.                                                                                                                                   |
| 0F338h | CLCKIC    | 1 | Drapeau pour indiquer la présence de l'horloge interne.<br>00h = Pas d'horloge, 0FFh = Horloge présente .                                                                                                                                                    |



|        |         |   |                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------|---------|---|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0F339h |         | 6 | Zone de travail de la puce de l'horloge interne.                                                                                                                                                                                                                                                                                                                                                      |
| 0F33Fh |         | 1 | Nom du second disque virtuel (0 ~ 7). 0 = « A: », 1 = « B: », etc.                                                                                                                                                                                                                                                                                                                                    |
| 0F340h | REBOOT  | 1 | Mettre à 0 pour exécuter l'AUTOEXEC.BAT au démarrage du MSX-DOS. (Voir aussi la routine STBAS (04022h) du <a href="#">BIOS de la Disk-ROM</a> à la page 430 pour plus de détails.)                                                                                                                                                                                                                    |
| 0F341h | RAMAD0  | 1 | Numéro du Slot qui contient la RAM principale du DOS de la plage 0 (00000h~03FFFh).                                                                                                                                                                                                                                                                                                                   |
| 0F342h | RAMAD1  | 1 | Numéro du Slot qui contient la RAM principale du DOS de la plage 1 (04000h~05FFFh).                                                                                                                                                                                                                                                                                                                   |
| 0F343h | RAMAD2  | 1 | Numéro du Slot qui contient la RAM principale du DOS/Disk-Basic de la plage 2 (08000h~0BFFFh).                                                                                                                                                                                                                                                                                                        |
| 0F344h | RAMAD3  | 1 | Numéro du Slot qui contient la RAM principale du DOS/Disk-Basic de la plage 3 (0C000h~0FFFFh).                                                                                                                                                                                                                                                                                                        |
| 0F345h |         | 1 | Nombre de FCB disponibles sous Basic.                                                                                                                                                                                                                                                                                                                                                                 |
| 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 des disques.                                                                                                                                                                                                                                                                                                                                                   |
| 0F34Bh | DOSHIM  | 2 | Adresse de fin du noyau du DOS. (0000h par défaut sous Basic.)<br>Note : L'adresse de début du noyau est stockée à l'adresse 0006h. Le fichier MSX-DOS.SYS est chargé dans cette zone.                                                                                                                                                                                                                |
| 0F34Dh | SECBUF  | 2 | Adresse du cache temporaire utilisé lors de lecture / écriture dans la FAT.                                                                                                                                                                                                                                                                                                                           |
| 0F34Fh | BUFFER  | 2 | Adresse du cache temporaire utilisé lors de lecture / écriture dans les secteurs de données et nom de fichiers.                                                                                                                                                                                                                                                                                       |
| 0F351h | DIRBUF  | 2 | Adresse du cache temporaire utilisé lors de lecture / écriture d'un secteur ou de la FAT. Utilisé par les instructions DSKI\$ & DSKO\$ du Disk-Basic.                                                                                                                                                                                                                                                 |
| 0F353h | FCBBASE | 2 | Adresse du FCB du fichier actuel.                                                                                                                                                                                                                                                                                                                                                                     |
| 0F355h | DPBLIST | 2 | Adresses du DPB de chaque disque.<br>DPBLST=Adresse du DPB des disques « A: »<br>DPBLST+2=Adresse du DPB du disque « B: »<br>DPBLST+4=Adresse du DPB du disque « C: »<br>DPBLST+6=Adresse du DPB du disque « D: »<br>DPBLST+8=Adresse du DPB du disque « E: »<br>DPBLST+10=Adresse du DPB du disque « F: »<br>DPBLST+12=Adresse du DPB du disque « G: »<br>DPBLST+14=Adresse du DPB du disque « H: ». |
| 0F365h |         | 3 | Routine de lecture de l'état des Slot primaires.<br>Sortie : A = État des Slot 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.)<br>Entrée : HL = Source ; DE = Destination ; BC = Longueur. |
| 0F371h | AUXINP  | 3   | Routine de lecture du périphérique auxiliaire.<br>Sortie : A = Valeur lue. 01Ah si CTRL+Z.                                                    |
| 0F374h | AUXOUT  | 3   | Routine d'écriture au périphérique auxiliaire.<br>Entrée : A = Octet à envoyer.                                                               |
| 0F377h | BLDCHK  | 3   | Routine BLOAD du Disk-Basic. (Contient JP 0000h sous MSX-DOS)                                                                                 |
| 0F37Ah | BSVCHK  | 3   | Routine BSAVE du Disk-Basic. (Contient JP 0000h sous MSX-DOS)                                                                                 |
| 0F37Dh | ROMBDOS | 3   | Routine BDOS de la Disk-ROM. (Peut-être appelée sous environnement Basic.)<br>Entrée : C = Numéro de la Routine à appeler.                    |
| 0F459h |         | 100 | Contient la commande entrée sous MSX-DOS.                                                                                                     |
| 0F85Fh | MAXFIL  | 1   | Nombre maximum de fichiers autorisés. Modifié par l'instruction MAXFILES du Disk-Basic.                                                       |
| 0F860h | FILTAB  | 2   | Pointeur sur l'adresse des données du fichier.                                                                                                |
| 0F862h | NULBUF  | 2   | Pointeur du cache des instructions SAVE et LOAD du Disk-Basic.                                                                                |
| 0F864h | PTRFIL  | 2   | Pointeur sur les données du fichier sélectionné.                                                                                              |
| 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 | 8  | <p>DRVINF+0 = Nombre de disque physique contrôlé par la première Disk-ROM. (8 maximum)</p> <p>DRVINF+1 = Numéro de Slot de la Disk-ROM des premiers disques installés.</p> <p>DRVINF+2 = Nombre de disque physique contrôlé par la seconde Disk-ROM. (8 disques maximum au total)</p> <p>DRVINF+3 = Numéro de Slot de la Disk-ROM des disques installés en second.</p> <p>DRVINF+4 = Nombre de disque physique contrôlé par la troisième Disk-ROM. (8 disques maximum au total)</p> <p>DRVINF+5 = Numéro de Slot de la Disk-ROM des disques installés en troisième.</p> <p>DRVINF+6 = Nombre de disque physique contrôlé par la quatrième Disk-ROM. (8 disques maximum au total)</p> <p>DRVINF+7 = Numéro de Slot de la Disk-ROM des disques installés en quatrième.</p> |
| 0FB29h | DRVINT | 12 | Numéro de Slot et adresse de chaque gestionnaire d'interruption des interfaces de disque.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 0FCBFh | SAVENT | 2  | Adresse de début spécifiée par l'instruction BSAVE du Basic.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 0FD99h | DEVICE | 1  | Cet octet passe à 255 si la touche SHIFT a été pressée au démarrage pour empêcher l'installation des disques. Sinon, cet octet reste à 0.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

### 13.12 Zone fixe de travail, Hooks et des variables du MSX-DOS2 et du Disk-BASIC 2

Cette zone est assez différente de celle du MSX-DOS1, c'est pour cela que je la met à part ici.

| Adresse | Nom   | Long. | Fonction                                                                                           |
|---------|-------|-------|----------------------------------------------------------------------------------------------------|
| 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 ?                                                                                |
| 0F1E6h  | ABORD | 3     | Saut vers le gestionnaire d'abandon.                                                               |
| 0F1E8h  |       | 3     | Saut à la routine RDSLT du Bios (000Ch). (Appelé seulement lors du traitement des fonctions BDOS). |
| 0F1EBh  |       | 3     | Saut à la routine WRS�T 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.<br>Entrée : HL = Adresse, A = Numéro de page.                                                                                                            |
| 0F209h | 3 | Saut à la routine d'écriture d'un octet sur une page de Memory Mapper.<br>Entrée : HL = Adresse, A = Numéro de page.<br>Sortie : E = Octet lu.                                                                                  |
| 0F20Ch | 3 | Saut à la routine d'appel de routine se trouvant sur une page de Memory Mapper.<br>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.<br>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.<br>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.<br>Entrée : HL = adresse quelconque dans la plage mémoire sur laquelle on cherche à connaître la page actuellement sélectionnée.<br>Sortie : A = numéro de page. |
| 0F218h | 3 | Routine de sélection d'une page de Memory Mapper sur la plage mémoire 0000h~03FFFh.<br>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.<br>Entrée : A = numéro de page.                                                       |

|        |        |   |                                                                                                                                                                                                                                                        |
|--------|--------|---|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0F21Eh |        | 3 | Routine de sélection d'une page de Memory Mapper sur la plage mémoire 04000h~05FFFh.<br>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~05FFFh.<br>Entrée : A = numéro de page.                                                                              |
| 0F224h |        | 3 | Routine de sélection d'une page de Memory Mapper sur la plage mémoire 8000h~0BFFFh.<br>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.<br>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.<br>Entrée : A = numéro de page.                                                                                                                              |
| 0F23Ch |        | 1 | Numéro du disque logique actuelle.                                                                                                                                                                                                                     |
| 0F23Dh |        | 2 | Adresse actuelle de la DTA (0080h par défaut sous DOS).                                                                                                                                                                                                |
| 0F23Fh |        | 4 | Numéro du premier secteur du fichier dans lequel il y a eu le dernier accès.                                                                                                                                                                           |
| 0F243h |        | 2 | Pointeur à l'adresse du DPB du disque actuel.                                                                                                                                                                                                          |
| 0F245h |        | 1 | Numéro relatif du secteur du fichier suivant dans le dossier actuel. Utilisé par les routines SEARCH FIRST et NEXT. (Ajouter le numéro du premier secteur du fichier dans lequel il y a eu le dernier accès pour connaître le numéro du secteur réel.) |
| 0F246h |        | 1 | Numéro du disque sur lequel il y a eu le dernier accès à un dossier ou fichier. ( 0 = « A: », 1 = « B: », etc, 0FFh = Aucun accès)                                                                                                                     |
| 0F247h |        | 1 | Numéro du disque par défaut. ( 0 = « A: », 1 = « B: », etc, 0FFh = Pas de disque)                                                                                                                                                                      |
| 0F24Fh |        | 3 | Hook appelé avant ?                                                                                                                                                                                                                                    |
| 0F252h | H.BDOS | 3 | Hook appelé avant l'exécution d'une routine BDOS.<br>Page 0 = map bloc (F2D0h) ; Page 2 = map block (F2CFh)                                                                                                                                            |
| 0F255h |        | 3 | Hook appelé avant ?                                                                                                                                                                                                                                    |
| 0F258h |        | 3 | Hook appelé avant ?                                                                                                                                                                                                                                    |
| 0F261h | H.CON  | 3 | Hook appelé au début de la fonction 02h et 0Bh du BDOS.                                                                                                                                                                                                |
| 0F283h |        | 2 | Adresse de la TPA. (MSX-DOS 2.33)                                                                                                                                                                                                                      |

|        |   |                                                                                                                                                                                                                        |
|--------|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0F2B6h | 1 | Indicateur disponibles sous MSX-DOS 2.33<br>bit 0~2 = Réservés,<br>bit 3 = VDP rapide OFF,<br>bit 4 = TPA propre OFF (0F2B3h = adresse de la TPA),<br>bit 5 = RESET ON,<br>bit 6 = BUSRESET OFF,<br>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. Cache 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. Cache 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 du Bios étendu (FFCAh).                                                                                                                                                              |
| 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 Slot primaires sélectionnés sur chaque plage.                                                                                                                                                               |

|        |           |    |                                                                                        |
|--------|-----------|----|----------------------------------------------------------------------------------------|
| 0F2E6h |           | 2  | Cache utilisé pour registre de stockage temporaire IX                                  |
| 0F2E8h |           | 2  | Cache utilisé pour le stockage temporaire du registre SP.                              |
| 0F2EAh |           | 1  | Statut des Slot primaires après l'exécution d'une fonction de BDOS.                    |
| 0F2EBh |           | 1  | Statut des Slot 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 cache temporaire utilisé pendant l'interprétation d'un code d'erreur. |
| 0F2FDh |           | 1  | Lecteur à partir duquel MSXDOS.SYS doit être chargé.                                   |
| 0F2FEh |           | 2  | Pointeur du haut la pile du DOS2.                                                      |
| 0F300h |           | 13 | ?                                                                                      |
| 0F30Dh |           | 1  | Vérification du disque (00h = OFF, 01h = ON.)                                          |
| 0F30Fh | KANJTABLE | 4  | Deux paires de limites pour les premiers octets des caractères Shift-JIS.              |
| 0F313h |           | 1  | Version de la Disk-ROM. 0 = Disk-ROM v.1.x, 2xh = Disk-ROM v.2.x.                      |
| 0F314h | MAPSEG0   | 1  | Page du Memory Mapper principal sur la plage 0000h~3FFFh.                              |
| 0F315h | MAPSEG1   | 1  | Page du Memory Mapper principal sur la plage 4000h~7FFFh.                              |
| 0F316h | MAPSEG2   | 1  | Page du Memory Mapper principal sur la plage 8000h~BFFFh.                              |
| 0F317h | MAPSEG3   | 1  | Page du Memory Mapper principal sur la plage C000h~FFFFh.                              |

|        |         |       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |       |       |       |                  |       |       |       |        |   |   |   |   |   |   |   |         |   |   |   |   |   |   |   |               |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |            |   |   |   |   |   |   |   |                  |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |             |
|--------|---------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|-------|-------|------------------|-------|-------|-------|--------|---|---|---|---|---|---|---|---------|---|---|---|---|---|---|---|---------------|---|---|---|---|---|---|---|-----------|---|---|---|---|---|---|---|-----------|---|---|---|---|---|---|---|------------|---|---|---|---|---|---|---|------------------|---|---|---|---|---|---|---|-------------|---|---|---|---|---|---|---|-------------|
| 0F323h | DISKVE  | 2     | Adresse du pointeur de la routine de gestion des erreurs de disque. Si vous avez besoin de faire une routine pour gérer les erreurs, il suffit de remplacer cette adresse par celle de votre routine. Lorsqu'une erreur se produit le registre A indique le numéro du disque sur lequel l'erreur s'est produite. Le registre C indique le type d'erreur de la façon suivante.<br><br>Le bit 0 indique si l'erreur s'est produit lors d'une écriture (1) ou une lecture (0). Les autres bits indiquent le type d'erreur comme suit.                                                                                                                                                                                                                                                                                                                                                                                                                              |       |       |       |                  |       |       |       |        |   |   |   |   |   |   |   |         |   |   |   |   |   |   |   |               |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |            |   |   |   |   |   |   |   |                  |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |             |
|        |         |       | <table><tr><td>bit 7</td><td>bit 6</td><td>bit 5</td><td>bit 4</td><td>bit 3</td><td>bit 2</td><td>bit 1</td><td>Erreur</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>Bad FAT</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>Write protect</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>Not ready</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>CRC error</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>Seek error</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>Record not found</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>Write fault</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>Other error</td></tr></table> | bit 7 | bit 6 | bit 5 | bit 4            | bit 3 | bit 2 | bit 1 | Erreur | 1 | 0 | 0 | 0 | 0 | 0 | 0 | Bad FAT | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Write protect | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Not ready | 0 | 0 | 0 | 0 | 0 | 1 | 0 | CRC error | 0 | 0 | 0 | 0 | 0 | 1 | 1 | Seek error | 0 | 0 | 0 | 0 | 1 | 0 | 0 | Record not found | 0 | 0 | 0 | 0 | 1 | 0 | 1 | Write fault | 0 | 0 | 0 | 0 | 1 | 1 | 0 | Other error |
| bit 7  | bit 6   | bit 5 | bit 4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | bit 3 | bit 2 | bit 1 | Erreur           |       |       |       |        |   |   |   |   |   |   |   |         |   |   |   |   |   |   |   |               |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |            |   |   |   |   |   |   |   |                  |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |             |
| 1      | 0       | 0     | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 0     | 0     | 0     | Bad FAT          |       |       |       |        |   |   |   |   |   |   |   |         |   |   |   |   |   |   |   |               |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |            |   |   |   |   |   |   |   |                  |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |             |
| 0      | 0       | 0     | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 0     | 0     | 0     | Write protect    |       |       |       |        |   |   |   |   |   |   |   |         |   |   |   |   |   |   |   |               |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |            |   |   |   |   |   |   |   |                  |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |             |
| 0      | 0       | 0     | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 0     | 0     | 1     | Not ready        |       |       |       |        |   |   |   |   |   |   |   |         |   |   |   |   |   |   |   |               |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |            |   |   |   |   |   |   |   |                  |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |             |
| 0      | 0       | 0     | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 0     | 1     | 0     | CRC error        |       |       |       |        |   |   |   |   |   |   |   |         |   |   |   |   |   |   |   |               |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |            |   |   |   |   |   |   |   |                  |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |             |
| 0      | 0       | 0     | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 0     | 1     | 1     | Seek error       |       |       |       |        |   |   |   |   |   |   |   |         |   |   |   |   |   |   |   |               |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |            |   |   |   |   |   |   |   |                  |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |             |
| 0      | 0       | 0     | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 1     | 0     | 0     | Record not found |       |       |       |        |   |   |   |   |   |   |   |         |   |   |   |   |   |   |   |               |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |            |   |   |   |   |   |   |   |                  |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |             |
| 0      | 0       | 0     | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 1     | 0     | 1     | Write fault      |       |       |       |        |   |   |   |   |   |   |   |         |   |   |   |   |   |   |   |               |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |            |   |   |   |   |   |   |   |                  |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |             |
| 0      | 0       | 0     | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 1     | 1     | 0     | Other error      |       |       |       |        |   |   |   |   |   |   |   |         |   |   |   |   |   |   |   |               |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |            |   |   |   |   |   |   |   |                  |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |             |
| 0F325h | BREAKV  | 2     | Adresse du pointeur de la routine de gestion de CTRL+C                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |       |       |                  |       |       |       |        |   |   |   |   |   |   |   |         |   |   |   |   |   |   |   |               |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |            |   |   |   |   |   |   |   |                  |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |             |
| 0F339h |         | 6     | Zone de travail de la puce de l'horloge interne.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |       |       |       |                  |       |       |       |        |   |   |   |   |   |   |   |         |   |   |   |   |   |   |   |               |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |            |   |   |   |   |   |   |   |                  |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |             |
| 0F33Fh |         | 1     | Nom du second disque virtuel (0 ~ 7). 0 = « A: », 1 = « B: », etc.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |       |       |                  |       |       |       |        |   |   |   |   |   |   |   |         |   |   |   |   |   |   |   |               |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |            |   |   |   |   |   |   |   |                  |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |             |
| 0F340h | REBOOT  | 1     | Mettre à 0 pour exécuter l'AUTOEXEC.BAT au démarrage du MSX-DOS. (Voir aussi la routine STBAS (04022h) du <a href="#">BIOS de la Disk-ROM</a> à la page 430 pour plus de détails.)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |       |       |                  |       |       |       |        |   |   |   |   |   |   |   |         |   |   |   |   |   |   |   |               |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |           |   |   |   |   |   |   |   |            |   |   |   |   |   |   |   |                  |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |             |
| 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 caches 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 | DOSHIM  | 2   | Adresse de fin du noyau du DOS. (0000h par défaut sous Basic.)<br>Note : L'adresse de début du noyau est stockée à l'adresse 0006h lorsqu'une commande est chargée. Le fichier MSX-DOS.SYS est chargé dans cette zone.                                                                                                                                                                                |
| 0F34Dh | SECBUF  | 2   | Adresse du cache temporaire utilisé lors de lecture / écriture dans la FAT.                                                                                                                                                                                                                                                                                                                           |
| 0F34Fh | BUFFER  | 2   | Adresse du cache temporaire utilisé lors de lecture / écriture dans les secteurs de données.                                                                                                                                                                                                                                                                                                          |
| 0F351h | DIRBUF  | 2   | Adresse du cache temporaire utilisé lors de lecture / écriture d'un secteur. Utilisé par les instructions DSKI\$ & DSKO\$ du Disk-Basic.                                                                                                                                                                                                                                                              |
| 0F353h | FCBBASE | 2   | Adresse du FCB du fichier actuel.                                                                                                                                                                                                                                                                                                                                                                     |
| 0F355h | DPBLST  | 2   | Adresses du DPB de chaque disque.<br>DPBLST=Adresse du DPB des disques « A: »<br>DPBLST+2=Adresse du DPB du disque « B: »<br>DPBLST+4=Adresse du DPB du disque « C: »<br>DPBLST+6=Adresse du DPB du disque « D: »<br>DPBLST+8=Adresse du DPB du disque « E: »<br>DPBLST+10=Adresse du DPB du disque « F: »<br>DPBLST+12=Adresse du DPB du disque « G: »<br>DPBLST+14=Adresse du DPB du disque « H: ». |
| 0F365h |         | 3   | Routine de lecture de l'état des Slot primaires.<br>Sortie : A = État des Slot primaires.                                                                                                                                                                                                                                                                                                             |
| 0F377h |         | 3   | Routine d'appel des fonctions BDOS en page 0. (Peut-être appelée sous environnement Basic.)                                                                                                                                                                                                                                                                                                           |
| 0F37Ah |         | 3   | Deuxième routine d'appel des fonctions BDOS en page 0.                                                                                                                                                                                                                                                                                                                                                |
| 0F37Dh | ROMBDOS | 3   | Routine BDOS de la Disk-ROM (du MSX-DOS2).<br>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 cache des instructions SAVE et LOAD du Disk-Basic.                                                                                                                                                                                                                                                                                                                                        |
| 0F864h | PTRFIL  | 2   | Pointeur sur les données du fichier sélectionné.                                                                                                                                                                                                                                                                                                                                                      |
| 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 | 8  | <p>DRVINF+0 = Nombre de disque physique contrôlé par la première Disk-ROM. (8 maximum)</p> <p>DRVINF+1 = Numéro de Slot de la Disk-ROM des premiers disques installés.</p> <p>DRVINF+2 = Nombre de disque physique contrôlé par la seconde Disk-ROM. (8 disques maximum au total)</p> <p>DRVINF+3 = Numéro de Slot de la Disk-ROM des disques installés en second.</p> <p>DRVINF+4 = Nombre de disque physique contrôlé par la troisième Disk-ROM. (8 disques maximum au total)</p> <p>DRVINF+5 = Numéro de Slot de la Disk-ROM des disques installés en troisième.</p> <p>DRVINF+6 = Nombre de disque physique contrôlé par la quatrième Disk-ROM. (8 disques maximum au total)</p> <p>DRVINF+7 = Numéro de Slot de la Disk-ROM des disques installés en quatrième.</p> |
| 0FB29h | DRVINT | 12 | Numéro de Slot et adresse de chaque gestionnaire d'interruption des interfaces de disque.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 0FCBFh | SAVENT | 2  | Adresse de début spécifiée par l'instruction BSAVE du Disk-Basic.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 0FD99h | DEVICE | 1  | Cet octet passe à 255 si la touche SHIFT a été pressée au démarrage pour empêcher l'installation des disques. Sinon, cet octet reste à 0.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 0FFCAh | EXTBIO | 5  | <p>Appel à une routine d'un Bios du étendu.</p> <p>Entrée : D = Identifiant de l'extension.</p> <p>0 = Tous,</p> <p>4 = Memory Mapper,</p> <p>8 = MSX-Modem ou RS-232C,</p> <p>10 = MSX-Audio,</p> <p>16 = MSX-JE,</p> <p>17 = Kanji driver,</p> <p>34 = UNAPI (Konamiman),</p> <p>77 = Memman,</p> <p>132 = µ-driver (Yoshikazu Yamamoto),</p> <p>241 = MultiMente (Mogu),</p> <p>255 = Système.</p> <p>E = Numéro de la routine à appeler.</p> <p>Sortie : Dépend de la routine appelée.</p> <p>Note : Cette routine est présente si le bit 0 de HOKVLD (0FB20h) est à 1. (Voir le paragraphe 13.13 sur <a href="#">le Bios étendu</a> page 524 pour plus de précision)</p>                                                                                            |

|        |        |    |                                                                                                                                                  |
|--------|--------|----|--------------------------------------------------------------------------------------------------------------------------------------------------|
| 0FFCFh | DISINT | 5  | Appel à une routine qui permet de désactiver les interruptions pour un long moment. Il s'agit en fait de la routine suivante.                    |
|        |        |    | <pre> disint:     ld    d,0                ;broad-cast command     ld    e,2                ;'disable interrupt'     call  extbio     ret </pre> |
| 0FFD4h | ENAIN  | 5  | Appel à une routine qui permet de ré-activer les interruptions coupées avec DISINT. Il s'agit en fait de la routine suivante.                    |
|        |        |    | <pre> enaint:     ld    d,0                ;broad-cast command     ld    e,3                ;'enable interrupt'     call  extbio     ret </pre>  |
| 0FFD9h |        | 14 | Zone de travail pour DISINT / ENAIN.                                                                                                             |

---

### 13.13 Le Bios étendu

Le Bios étendu est une routine appelée EXTBIO qui a été ajouté au standard MSX avec le MSX-DOS 2 afin de connaître combien et quelles extensions sont connectées au système. Cette routine permet de les gérer plus aisément les extensions en langage machine ou en C.

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 ou créer 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 et passez à l'étape 3.
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 étendu à 0FFCAh. Votre routine devra se terminer par un saut vers la zone copiée auparavant (à l'étape 2) ou bien un RET si l'étape 2 a été sautée.
4. Créez les routines DISINT et ENAINT si elles ne sont pas présentes.

#### 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 plage 0~3FFFh
EXTBIO      equ      0ffc0h      ; Adresse d'appel à un Bios étendu

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 = CF à 1 si aucune page libre, Si 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 = CF 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   |
| Adresse indiquée par HL en sortie : | Octet réservé (00h par défaut)               |
|                                     |                                              |

**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 de la première table          |
|                                     | Octet de poids faible de l'adresse des sauts |
|                                     | Octet de poids fort de l'adresse des sauts   |
| Adresse indiquée par HL en sortie : | Octet réservé (Normalement 00h)              |
|                                     | Numéro du Slot de la seconde table           |
|                                     | Octet de poids faible de l'adresse des sauts |
|                                     | Octet de poids fort de l'adresse des sauts   |
|                                     | Octet réservé (Normalement 00h)              |

Format d'une 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 routines 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 chaîne 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 chaine MML. (0~8)  
 BUF+5 = Nombre de voix FM de la troisième chaine MML. (0~7)  
 BUF+6 = Nombre de voix FM de la quatrième chaine MML. (0~6)  
 BUF+7 = Nombre de voix FM de la cinquième chaine MML. (0~5)  
 BUF+8 = Nombre de voix FM de la sixième chaine MML. (0~4)  
 BUF+9 = Nombre de voix FM de la septième chaine MML. (0~3)  
 BUF+10 = Nombre de voix FM de la huitième chaine MML. (0~2)  
 BUF+11 = Nombre de voix FM de la neuvième chaine MML. (0~1)

Sortie : F = CF 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 **SYNTHE**

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 chaine 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 = CF passe à 1 en cas d'erreur de paramètre.

#### HL+15h **PLAYMK**

Rôle : Jour un air enregistré au clavier musical.

Entrée : BC = Adresse de début des données enregistrées au clavier musical.

DE = Adresse de fin des données enregistrées 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 chaîne 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 = CF passe à 1 en cas d'erreur de paramètre.

#### HL+24h **STPPLY**

Rôle : Stopper la lecture d'une chaîne 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 = CF 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 poids faible 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 = CF 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 = CF 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 = CF 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 = CF 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 = CF 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 = CF 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                                      | 005Fh 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 = CF 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 = CF 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 = CF 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.

### 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. (0 ~ 2)  
 Modifie : Tout sauf BC, DE, HL.

## 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 : E.  
 Note : La pile doit être sur la plage 0C000h~0FFFFh. Par conséquent, le point d'accès recherche l'adresse d'entrée comme suit.

Exemple de programme pour obtenir l'adresse d'entrée du BIOS étendue du MSX-JE:



```

HOKVLD equ 0fb20h      ;Extended BIOS flag
EXTBIO equ 0ffc0h      ;Extended BIOS hook
;
        ld    a, (HOKVLD)
        rra
        jr    nc, NO_MJE
        ld    hl, MJETBL
        call  GETSLT      ;Get Slot address of MJETBL into [B]
        ld    de, 16*256+0
        push  hl
        call  EXTBIO
        pop   de
        or    a
        sbc   hl, de
        jr    z, NO_MJE
        .
        .
        .

```

En sortie de EXTBIO, une table est créée comme suit à l'adresse MJETBL et HL pointe l'octet derrière la table. Lorsque la fonction MSX-JE n'existe pas, HL ne change pas. La table qui suit est un exemple avec deux MSX-JE.

|                                     |                                                           |   |                               |
|-------------------------------------|-----------------------------------------------------------|---|-------------------------------|
| Valeur de MJETBL :                  |                                                           | } | Descripteur du premier MSX-JE |
|                                     | Vecteur de capacité                                       |   |                               |
|                                     | Numéro du Slot                                            |   |                               |
|                                     | Octet de poids faible de l'adresse de la table des sauts. |   |                               |
|                                     | Octet de poids fort de l'adresse d'entrée                 | } | Descripteur du second MSX-JE  |
|                                     | Vecteur de capacité                                       |   |                               |
|                                     | Numéro du Slot                                            |   |                               |
|                                     | Octet de poids faible de l'adresse d'entrée               |   |                               |
|                                     | Octet de poids fort de l'adresse d'entrée                 |   |                               |
| Adresse indiquée par HL en sortie : | .                                                         |   |                               |
|                                     | .                                                         |   |                               |
|                                     | .                                                         |   |                               |

### Fonction 1 (Inquiry)

Rôle : Renvoie la taille de la zone de travail.

Entrée : A = 1

Sortie : BC = Limite supérieure de la zone de travail utilisée par MSX-JE (MAX).

DE = Limite inférieure de la zone de travail utilisée par MSX-JE (MIN).

HL = Taille minimale requise pour utiliser la fonction d'apprentissage (MIN2).

Modifie : .

Notes :

- Une application peut déterminer la taille de la zone de travail requise pour MSX-JE avec n de la façon suivante.

| Par rapport aux valeurs renvoyées | Taille de la zone de travail                                         |
|-----------------------------------|----------------------------------------------------------------------|
| $n < \text{MIN}$                  | (L'application ne peut pas utiliser MSX-JE)                          |
| $\text{MIN} \leq n < \text{MIN2}$ | MIN (l'application ne peut pas utiliser la fonction d'apprentissage) |
| $\text{MIN} \leq n < \text{MAX}$  | n                                                                    |
| $\text{MAX} \leq n$               | MAX                                                                  |

- Il est possible qu'une application ne puisse pas utiliser la fonction d'apprentissage. De plus, si vous pouvez toujours sécuriser 2560 octets de travail, vous n'avez pas besoin d'appeler cette fonction.
- Avec cette fonction, l'adresse de la zone de travail n'est pas transmise au MSX-JE en tant que paramètre. Par conséquent, MSX-JE ne peut pas utiliser la zone de travail.

### Fonction 2 (Invoke)

Rôle : Démarrer MSX-JE. (Initialise une zone de travail sécurisée pour une application.)

Entrée :  $A = 2$   
DE = Taille de la zone de travail  
HL = Adresse de la zone de travail

Sortie : Rien.

Modifie : .

Notes :

### Fonction 3 (Release)

Rôle : Quitter MSX-JE. (Libérer une zone de travail sécurisée par une application.)

Entrée :  $A = 3$   
HL = Adresse de la zone de travail

Sortie : Rien.

Modifie : .

Notes :

### Fonction 4 (Clear)

Rôle : Efface la cache pour la conversion Kana-Kanji. (Efface le texte en cours de saisi.)

Entrée :  $A = 4$   
HL = Adresse de la zone de travail.

Sortie : Rien.

Modifie : .

Notes :

## Fonction 5 (Set TTB)

Rôle : .

Entrée : A = 5

Sortie : BC = Adresse TTB.  
DE = Adresse des données du texte à reconvertir.  
HL = Adresse de la zone de travail.

Modifie : .

Notes :

## 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 étendus 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 de l'octet derrière la table.

Modifie : Tout.

Note : Format de la table d'information sur les Bios étendus.

|                                     |                                                  |
|-------------------------------------|--------------------------------------------------|
| Adresse indiquée par HL en entrée : |                                                  |
|                                     | Numéro du Slot du périphérique                   |
|                                     | Octet de poids faible de l'adresse des sauts     |
|                                     | Octet de poids fort de l'adresse des sauts       |
|                                     | Identifiant du fabricant (Voir liste ci-dessous) |
| Table du périphérique suivant :     | Octet réservé                                    |
|                                     | Numéro du Slot du périphérique                   |
|                                     | Octet de poids faible de l'adresse des sauts     |
|                                     | Octet de poids fort de l'adresse des sauts       |
|                                     | Identifiant du fabricant (Voir liste ci-dessous) |
|                                     | Octet réservé                                    |

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 disques 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.

## 14 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.

### 14.1 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 :

```
MNROM    equ    0FCC1h
ENASLT    equ    00024h
READYR    equ    0409Bh
;
BACK2BASIC:
    ld      a, (MNROM)
    ld      hl, 0        ; Bios en page 0
    call    ENASLT
    ld      h, 040h      ; Basic en page 1
    call    ENASLT
    jp      READYR
```

Lorsque l'on travaille sous Basic, ou que l'on ne touche pas aux Slot, il suffit d'effectuer un saut à la READYR mais sous DOS, il faut effectuer un saut à la Disk-ROM principale à BASENT.

## 14.2 Quel type de MSX ?

Pour savoir si l'ordinateur utilisé est un MSX1, MSX2 ou autre, 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.

## 14.3 Afficher un texte 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 :

```
CHPUT:    equ    00A2h
PRINT:    ld      hl, DATA
          call   ECRIRE
          ret
ECRIRE:    ld      a, (hl)
          cp      0
          ret     z
          call   CHPUT
          inc     hl
          jr      ECRIRE
DATA:     db      'COUCOU', 0
```

Dans cet exemple, l'adresse du premier caractère de la chaîne à afficher est chargée dans HL. Puis la routine ECRIRE est appelée pour afficher le texte. Le 0 en fin de la chaîne sert à indiquer la fin. Cette routine comme le PRINT du Basic gère les codes CTRL et ESC.

Vous devrez utiliser les variables système CSRX (0F3DDh) et CSRY (0F3DCh) pour indiquer l'abscisse et l'ordonnée du curseur si besoin. TTYPOS (0F661h) peut aussi être utile pour afficher à la suite d'un texte précédent.

## 14.4 Système a disquettes ou pas ?

Pour savoir si l'ordinateur possède une interface de disque ou non, il faut vérifier l'état du premier octet du Hook « H.PHYD » à l'adresse 0FFA7h. Celui-ci contient 0C9h (code du RET en Z80) lorsqu'il n'est pas modifié par une Disk-ROM. Donc 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 pas) ou une interface disque.

De plus, lorsqu'un lecteur de disquette est présent, certaines variables système sont modifiées :

| Adresse | Contenu                                                       |
|---------|---------------------------------------------------------------|
| 0FB21h  | Nombre de lecteurs connectés au premier contrôleur (1 ou 2)   |
| 0FB22h  | Numéro de Slot du premier contrôleur                          |
| 0FB23h  | Nombre de lecteurs connectés au second contrôleur (1 ou 2)    |
| 0FB24h  | Numéro de Slot du second contrôleur                           |
| 0FB25h  | Nombre de lecteurs connectés au troisième contrôleur (1 ou 2) |
| 0FB26h  | Numéro de Slot du troisième contrôleur                        |
| 0FB27h  | Nombre de lecteurs connectés au quatrième contrôleur (1 ou 2) |
| 0FB28h  | Numéro de Slot du quatrième contrôleur                        |

Un MSX peut donc gérer jusqu'à 8 disques simultanément (4 contrôleurs de lecteurs de disquettes, chacun d'eux ne pouvant gérer que deux lecteurs). S'il y a moins de quatre contrôleurs, les variables système correspondantes au contrôleur inexistant contiennent 0.

En cas d'absence de lecteur de disquette, cette zone mémoire n'est pas initialisée et peut contenir donc n'importe quoi.

## 14.5 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, puisque 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 :

```

PLAY      equ      073e5h

          DB        0feh          ; Entête pour les
          DW        START,END,START ; fichiers binaires

          org       0c000h

START:
          ld        hl,DATA
          call      PLAY
          ret

;
DATA:     db        22h,'O5L6DABDACF',22h
          db        22h,'O3L4DEDE',22h
          db        22h,'O7L8ADEFGE',22h,0

END:

```

Attention à ne pas oublier le 0 en fin de chaîne. Dans l'exemple ci-dessus, on présume que la Main-ROM est sélectionnée sur la plage 4000h~7FFFh.

## 14.6 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 machine pour donner un look professionnel à un programme en Basic. Dans ce 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 l'inconvénient d'être plutôt lente car les valeurs de plus de 8 bits doivent être envoyées en plusieurs fois.

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 (0F7F8h+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.

```

VALTYP     equ      0f663h

          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  00A2h
        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"MULTVAR2.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-7

db     0feh
dw     START, END, START
```

START:

```

        cp    3
        ret   nz          ; Retour au Basic si ce n'est pas une chaine
;
        ld    a,(de)      ; Longueur de la chaine
        or    a
        ret   z          ; Retour au Basic si longueur = 0

CONVC:   ld    b,a        ; Nombre de caractères dans B pour DJNZ
        inc   de
        ld    a,(de)      ; Prend un caractère dans la chaine
        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 chaine
NOCHAR:  djnz  CONVC
        ret              ; retour au Basic
END:

```

Après avoir tapé et assemblé puis sauvegardé cette routine sous le nom « MIN-MAJ.BIN », entrez le programme Basic suivant et faites RUN :

```

10 CLS
20 CLEAR 200,&HC000
30 BLOAD"MIN-MAJ.BIN":DEFUSR=&HC000
40 A$="TRANSFORMATION":A$=USR(A$)
50 PRINT A$

```

## 14.7 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). Ceci permet de créer nos propres instructions.

Lors de l'appel au Hook, le registre HL du CPU contient le pointeur de l'interpréteur Basic dans le cache 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 050h, 072h, 069h 06Eh, 074h (« PRINT »).

Pour créer votre instruction, vous disposez des trois routines suivantes en ROM :

### 00010h ou 04666h (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 = ZF à 1 si il s'agit d'un code de fin de ligne (00h ou « : »).  
                  CF 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)

### 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.
- Note :          Appelle le Hook H.FRME (0FF66h).

## 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

RETBASIC:
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
;
CHGCPU    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.

## 14.8 Les codes de contrôle [CTRL]

Le MSX peut effectuer une action en enfonçant simultanément la touche « CTRL » avec une autre. Ces actions peuvent être détectées lors d'une lecture du clavier mais aussi effectuées en mode texte en envoyant les codes à l'affichage. Voici la liste des actions disponibles avec les codes correspondants :

| Touches | Code     | Effet                                                                          |
|---------|----------|--------------------------------------------------------------------------------|
| CTRL+B  | 2        | Place le curseur au début du mot précédent                                     |
| CTRL+C  | 3        | Interrompt le programme (même effet que BREAK)                                 |
| CTRL+E  | 5        | Efface toute la ligne à droite du curseur                                      |
| CTRL+F  | 6        | Place le curseur au début du mot suivant                                       |
| CTRL+G  | 7        | Émission d'un bref bip sonore                                                  |
| CTRL+H  | 8        | Déplace le curseur à gauche d'un caractère et l'efface (même effet que BS)     |
| CTRL+I  | 9        | Tabulation (même effet que TAB)                                                |
| CTRL+J  | 10 (0Ah) | Descend le curseur d'une ligne                                                 |
| CTRL+K  | 11 (0Bh) | Déplace le curseur en haut à gauche de l'écran (même effet que EFE/HOME)       |
| CTRL+L  | 12 (0Ch) | Efface l'écran                                                                 |
| CTRL+M  | 13 (0Dh) | Effectue une entrée (même effet que RETURN)                                    |
| CTRL+N  | 14 (0Eh) | Place le curseur en fin de ligne                                               |
| CTRL+R  | 18 (12h) | Mode insertion (même effet que INS)                                            |
| CTRL+U  | 21 (15h) | Efface la ligne où se trouve le curseur                                        |
| CTRL+X  | 24 (18h) | Même effet que la touche SELECT                                                |
| CTRL+[  | 27 (1Bh) | Même effet que la touche ESC                                                   |
| CTRL+\  | 28 (1Ch) | Déplace le curseur d'un caractère à droite (même effet que la flèche droite)   |
| CTRL+]  | 29 (1Dh) | Déplace le curseur d'un caractère à gauche (même effet que la flèche gauche)   |
| CTRL+^  | 30 (1Eh) | Déplace le curseur d'un caractère vers le haut (même effet que la flèche haut) |
| CTRL+_  | 31 (1Fh) | Déplace le curseur d'un caractère vers le bas (même effet que la 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
```

## 14.9 Les codes d'échappement [ESC]

Le MSX peut effectuer une action en enfonçant simultanément la touche « ESC » avec une autre. Ces



actions peuvent être aussi effectuées en mode texte en envoyant les codes à l'affichage comme pour les codes de contrôle. Ces codes sont compatibles avec les terminaux VT-52 ou HEATH-19. En voici la liste :

| Touches         | Code      | Effet                                                       |
|-----------------|-----------|-------------------------------------------------------------|
| ESC+A           | 27+65     | Monte le curseur d'une ligne, stoppe en haut de l'écran     |
| ESC+B           | 27+66     | Descend le curseur d'une ligne, stoppe en bas de l'écran    |
| ESC+C           | 27+67     | Déplace le curseur vers la droite, stoppe en fin de ligne   |
| ESC+D           | 27+68     | Déplace le curseur vers la gauche, stoppe en début de ligne |
| ESC+H           | 27+72     | Place le curseur en haut à droite (HOME)                    |
| ESC+Y+ligne+col | 27+89+42  | Curseur en X, Y (LOCATE) voir ci-dessous                    |
| ESC+j           | 27+106    | Efface l'écran sans déplacer le curseur                     |
| ESC+E           | 27+69     | Efface l'écran et place le curseur en haut de l'écran (CLS) |
| ESC+J           | 27+74     | Efface jusqu'à la fin de l'écran                            |
| ESC+K           | 27+75     | Efface toute la ligne à droite du curseur                   |
| ESC+I           | 27+108    | Efface toute la ligne                                       |
| ESC+L           | 27+76     | Insère une ligne                                            |
| ESC+M           | 27+77     | Détruit une ligne                                           |
| ESC+x+4         | 27+120+52 | Définit un curseur de type entier                           |
| ESC+x+5         | 27+120+53 | Éteint le curseur                                           |
| ESC+y+4         | 27+121+52 | Définit un curseur de type barre                            |
| ESC+y+5         | 27+121+53 | Allume le curseur                                           |

La combinaison 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*AI CI"
```

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"LI GNE";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.

## 14.10 Afficher les caractères étendus

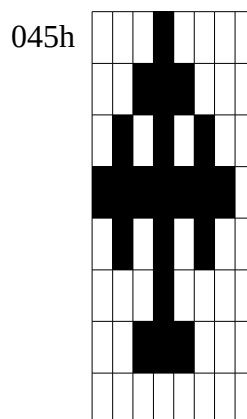
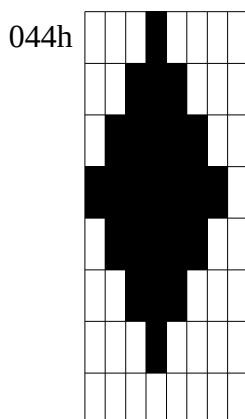
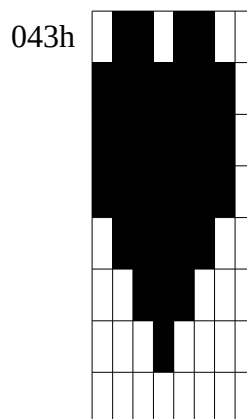
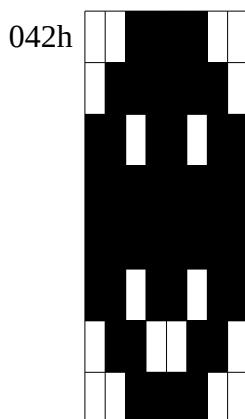
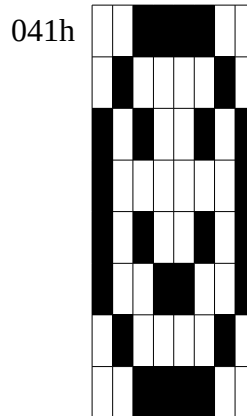
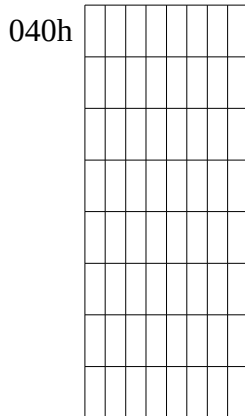
Certains MSX possèdent des caractères supplémentaires qui se trouvent physiquement dans la police de caractères entre 0 et 31. Comme ces numéros correspondent aux codes de contrôles, ces 32 caractères sont affichables avec un code ASCII entre 64 (040h) et 95 (05Fh) précédé du code 1.

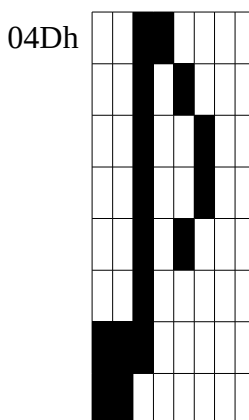
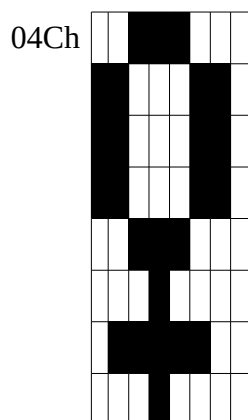
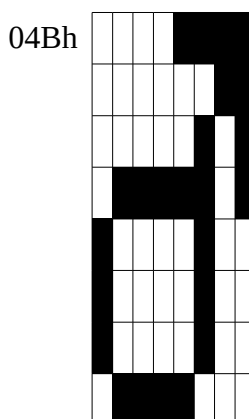
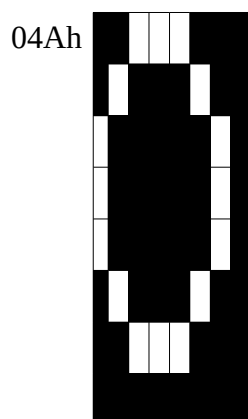
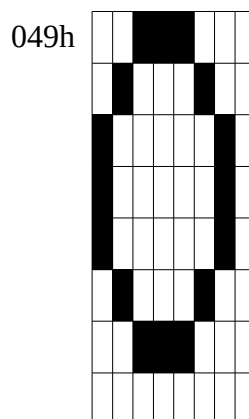
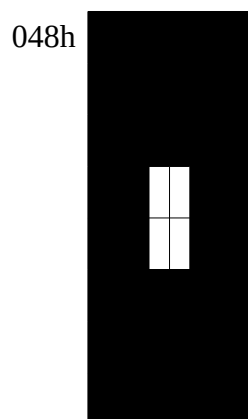
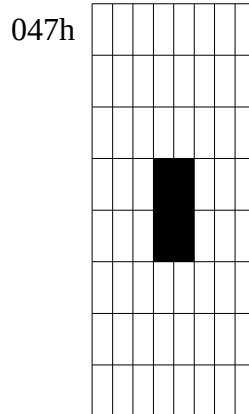
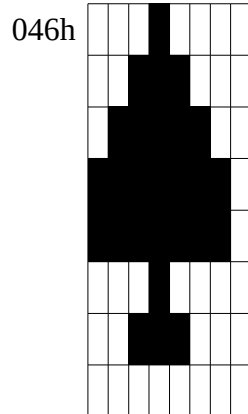
Par exemple, en SCREEN 1, la ligne :

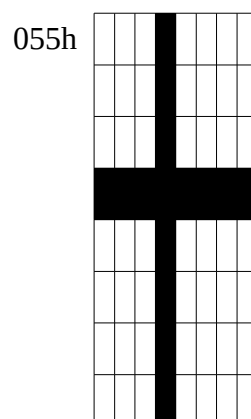
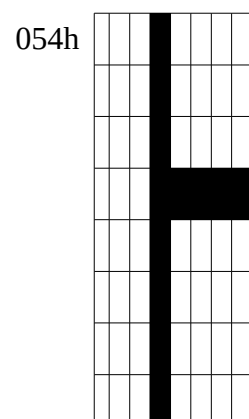
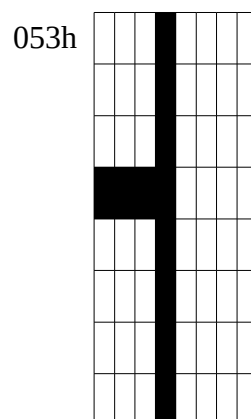
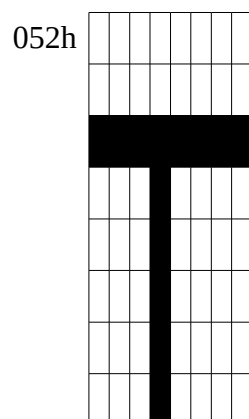
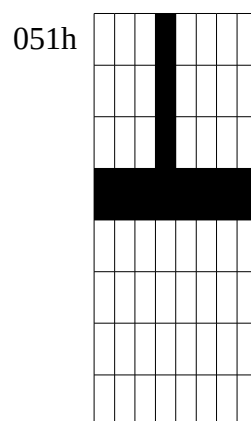
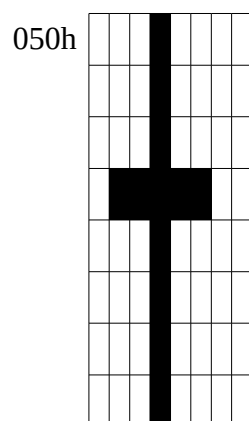
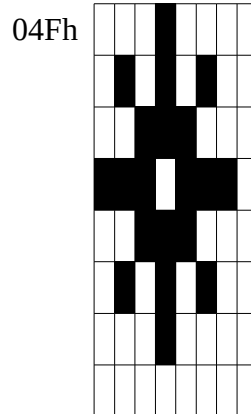
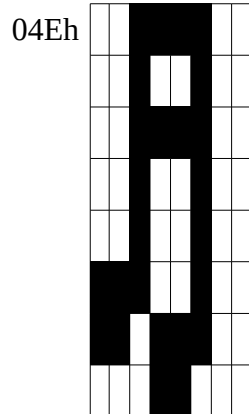
```
PRINT CHR$(1)+"A"
```

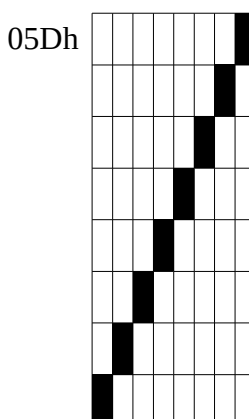
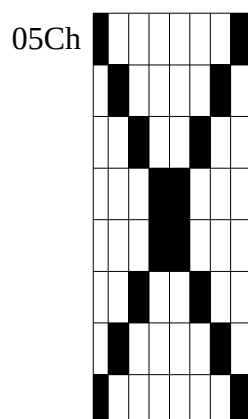
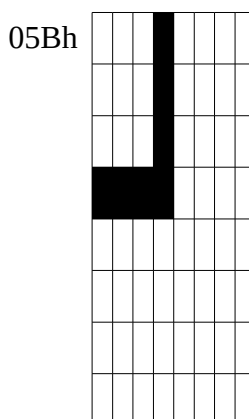
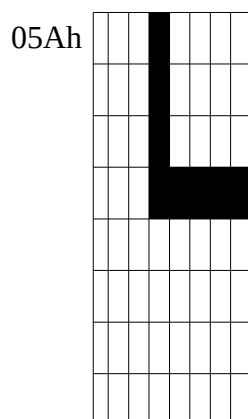
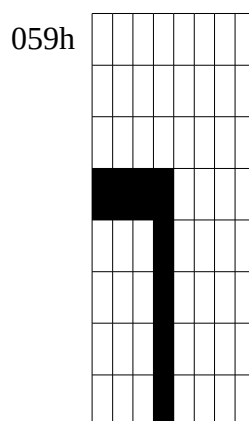
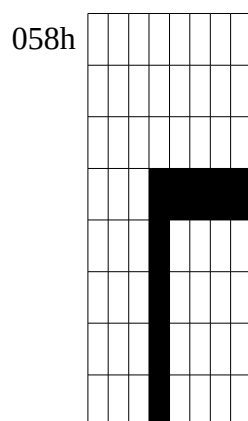
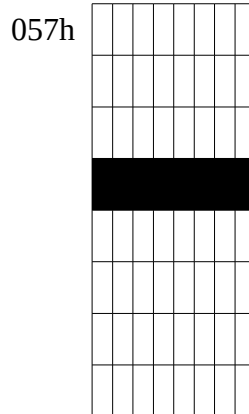
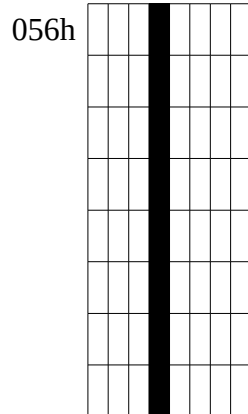
fera s'afficher à l'écran une petite tête sympathique. Notez que tous ces caractères se trouvent définis dans des matrices 8 sur 8. On perdra donc, en SCREEN 0, les 2 bits de poids faible, et le caractère sera amputé de ses 2 colonnes les plus à droite.

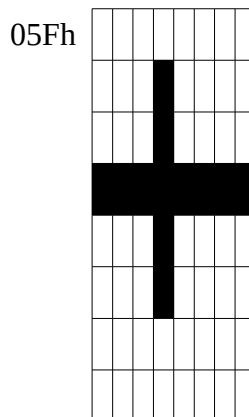
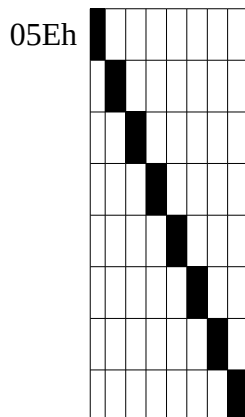
Voici les matrices de ces caractères pour les MSX occidentaux :











Voir les [Codes des caractères MSX](#) dans les annexes page 592 pour plus de détails.

### 14.11 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 plage 1 (04000h à 05FFFh). 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          ; Saute à LOOP si pas de 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 plage 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.

## 14.12 Ajouter des mots clés à l'instruction CALL du 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 instructions Basic supplémentaires que l'on peut exécuter avec l'instruction CALL (ou « \_ ») suivi d'un nom et éventuellement de paramètres.

L'adresse de la routine de traitement de ces instructions est donnée par le contenu de l'entête d'une ROM aux adresses 04004H et 04005h. A chaque fois que le Basic trouve un CALL, il appelle les routines. 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.

```
ENASLT: equ 00024h
FILVRM: equ 00056h
MNROM:  equ 0fcc1h

      org 0c000h
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
      jp 0c100h
RETOUR:
      ld a,(MNROM)
      ld hl,04000h
      jp ENASLT
ds 0c100h-$ ; Remplit l'espace entre les 2 routines
LC100:
      push hl
      ld hl,WORD
      ld de,0fd89h
```



```

LOOP2:
    ld    a, (DE)
    ld    b, (HL)
    cp    b
    jr    nz, SYNTAXERR
    cp    0
    inc    hl
    inc    de
    jr    z, OUT
    jr    LOOP2
SYNTAXERR:
    pop    hl
    xor    a
    ccf
    ret
OUT:
    pop    hl
    push   hl
    ld    a, (hl)
    cp    028h
    jr    nz, SYNTAXERR
    inc    hl
    ld    a, (hl)
    cp    022h
    jr    nz, SYNTAXERR
    inc    hl
    ld    b, (hl)
    inc    hl
    ld    a, (hl)
    cp    022h
    jr    nz, SYNTAXERR
    inc    hl
    ld    a, (hl)
    cp    029h
    jr    nz, SYNTAXERR
LOOP3:
    inc    hl
    ld    a, (hl)
    cp    020h
    jr    z, LOOP3
    cp    0
    jr    Z, SUITE
    cp    03Ah
    jr    nz, SYNTAXERR
SUITE:
    push   hl
    ld    hl, 0
    ld    a, b
    ld    bc, 00800h
    call   FILVRM
    pop    hl
    pop    bc
    xor    a
    ret
WORD:
    db     'FILLSCREEN', 0

```

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 : »

### 14.13 Manipuler la souris

Je vous propose un sous-programme en assembleur qui permet d'utiliser la souris depuis le Basic.

```
EXTROM      equ      0015fh
NEWPAD      equ      001Adh
GTTRIG      equ      000d8h
NWRVRM      equ      00177h

;
Souris:     org      0c000h

            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        00h
Y:          db        00h
```

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
```

## 14.14 Développer un programme en cartouche

Après avoir trouvé la RAM et initialisé les variables système de base, le MSX cherche les entêtes des ROM dans tous les Slot sur les plages mémoire 4000h~7FFFh et 8000h~BFFFh. La recherche s'effectue dans l'ordre croissant. Lorsqu'un Slot primaire est étendu, la recherche se fait dans les Slot secondaire correspondants avant de passer au Slot primaire suivant.

Lorsque le système trouve une entête, il sélectionne le Slot de la ROM uniquement sur la plage mémoire correspondante à l'adresse indiquée à INIT puis, exécute le programme en ROM à cette même adresse (En résumé, il effectue un appel inter-Slot).

Une entête se compose de 16 octets et doit être placés à l'adresse 4000h ou 8000h.

Format de l'entête d'une ROM :

| Entête | Nom       | Rôle                                                                                                                                                                                                                                |
|--------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| +0     | ID        | Les deux premiers octets doivent être 041H et 042H (« AB ») pour indiquer qu'il s'agit d'une ROM additionnelle.                                                                                                                     |
| +2     | INIT      | Ces deux octets sont prévus pour contenir l'adresse de la routine à appeler pour initialiser une zone de travail ou des ports E/S.<br>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 MSX-Basic au moyen de l'instruction CALL.<br>Indiquer 0000h si la ROM ne contient pas d'instruction Basic à ajouter. |
| +6     | DEVICE    | Adresse d'exécution d'un programme servant à contrôler un périphérique intégré à la cartouche. Par exemple, une interface pour disque.<br>Indiquer 0000h si la cartouche ne contient pas de matériel à installer.                   |
| +8     | TEXT      | Pointeur du programme Basic contenu en ROM.<br>Indiquer 0000h si la ROM ne contient pas de programme Basic.                                                                                                                         |
| +10~15 | Reserved  | Réservé. Ces octets doivent être mis à zéro.                                                                                                                                                                                        |

Description :

### INIT

C'est la première adresse prise en compte. Lorsque cette adresse est supérieure 0000h, le système sélectionne le Slot de la ROM sur la plage mémoire correspondante à l'adresse indiquée puis, exécute le programme en ROM à cette même adresse.

Au moment de l'exécution du programme INIT, bien que ce ne soit pas spécifié par le standard, le registre C contient toujours le numéro de Slot de la ROM sous la forme F000SSPP. La routine doit se terminer par un RET et afin le double registre SP retrouve la valeur qu'il avait au moment de l'appel à l'adresse INIT sinon le système ne pourra pas continuer à scruter les ROM dans les slots suivants. À la place d'une routine d'initialisation, la ROM peut très bien contenir un jeu et ne jamais rendre la main.

En général :

Si votre ROM a une taille de 32Ko et que INIT est comprise entre 4010h~7FFEh, vous pourrez sélectionner la deuxième partie (8000h~BFFFh) dès l'exécution de la ROM de la façon suivante.

```

ld    a,c
ld    h,080h ; La routine ENASLT ne tient pas compte du registre L
call  ENASLT ; Sélection du Slot de la ROM sur la page 8000h~BFFFh

```

Le standard conseille *la façon suivante*:

```

call  0138h
rrca
rrca
rrca
and    3          ; Garde les bits correspondants au Slot primaire (8000h-BFFFh)
ld     c,a
ld     b,0
ld     hl,0FCC1h
add    hl,bc
ld     a,(hl)
and    80h
or     c
ld     c,a
inc    hl
inc    hl
inc    hl
inc    hl
ld     a,(hl)
and    0Ch        ; Garde les bits correspondants au Slot secondaire (8000h-BFFFh)
or     c
ld     h,080h ; La routine ENASLT ne tient pas compte du registre L
call  ENASLT ; Sélection du Slot de la ROM sur la page 8000h~BFFFh

```

Si votre ROM a une taille de 32Ko et que INIT est comprise entre 8010h~BFFEh, vous pourrez sélectionner la première partie (4000h~7FFFh) dès l'exécution de la ROM de la façon suivante.

```

ld     a,c
ld     h,040h ; La routine ENASLT ne tient pas compte du registre L
call  ENASLT ; Sélection du Slot de la ROM sur la page 4000h~7FFFh

```

Le standard conseille *la façon suivante*:

```

call  0138h
rrca
rrca
rrca
rrca
and    3          ; Garde les bits correspondants au Slot primaire (4000h-7FFFh)
ld     c,a
ld     b,0
ld     hl,0FCC1h
add    hl,bc
ld     a,(hl)
and    80h
or     c
ld     c,a
inc    hl
inc    hl
inc    hl
inc    hl
ld     a,(hl)
and    0Ch        ; Garde les bits correspondants au Slot secondaire (4000h-7FFFh)
or     c
ld     h,040h ; La routine ENASLT ne tient pas compte du registre L
call  024h ; Sélection du Slot de la ROM sur la page 4000h~7FFFh

```

## STATEMENT

Le programme de traitement de l'instruction doit résider sur la plage 4000h~7FFFh.

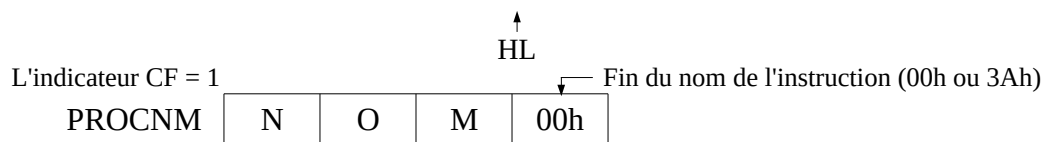
Une instruction appelée par CALL doit avoir le format suivant :

CALL <Nom de l'instruction> [(variable[, variable][,...])]

Le nom de l'instruction peut faire jusqu'à 15 caractères. Lorsque l'interpréteur du Basic trouve l'instruction CALL, il copie le nom de l'instruction dans la zone de travail PROCNM (0FD89h) puis cherche dans les Slot par ordre croissant une adresse STATEMENT supérieure à 0000h afin de transmettre le contrôle pour cette instruction. À ce moment là, le registre double HL contient l'adresse du paramètre qui suit le nom de l'instruction dans le listing. L'instruction peut être traitée. En sortie, HL doit indiquer l'instruction suivante à traiter et l'indicateur Carry doit indiquer si il y a eu une erreur.

Voici un exemple de procédure avec l'instruction CALL NOM (0, 0) suivit de A=0 :

1. Le listing contient donc CALL NOM (0, 0) : A=0



2. Traitement de l'instruction par la routine à l'adresse indiquée à STATEMENT.

Si le nom ne correspond pas alors laissez HL tel quel et mettez Carry à 1 avant de rendre la main à l'interpréteur (par un RET).

Si nom correspond, exécutez la routine de l'instruction et ses paramètres puis, pointez l'instruction suivante avec HL et mettez Carry à 0 si il n'y a pas erreur dans les paramètres.

3. Fin du traitement.

CALL NOM (0, 0) : A=0

↑ HL

Carry = 0 si il n'y a pas erreur

Rendez la main à l'interpréteur (par un RET).

Note : Evitez de donner un nom déjà existant à votre instruction car selon la position de la ROM dans les Slot, elle ne pourrait ne pas être prise en compte ou même provoquer une erreur à cause des paramètres.

## DEVICE

Cette adresse doit être comprise entre 04000h et 05FFFh. Le système peut contrôler jusqu'à 4 périphériques par cartouche. Le nom du périphérique doit faire 15 caractères maximum. Lorsque l'interpréteur Basic rencontre un nom de périphérique, il le copie dans la zone de travail PROCNM (0FD89h) puis met le registre A à 255 et cherche dans les Slot par ordre croissant une adresse DEVICE supérieure à 0000h afin de transmettre le contrôle du périphérique correspondant.

Voici un exemple de procédure avec l'instruction OPEN "NOM: " :

1. Le listing contient OPEN "NOM: "

A = Numéro de l'instruction (voir le tableau plus bas)

L'indicateur CF = 1

|        |                                         |   |   |     |
|--------|-----------------------------------------|---|---|-----|
| PROCNM | Fin du nom du périphérique (00h ou 3Ah) |   |   |     |
|        | N                                       | O | M | 00h |

2. Appel à la routine de contrôle du périphérique à l'adresse indiquée à DEVICE.

Si le nom ne correspond pas alors mettez le registre A à 255 et Carry à 1 avant de rendre la main à l'interpréteur (par un RET).

Si nom ne correspond, exécutez la routine de contrôle puis, pointez l'instruction suivante avec HL et mettez Carry à 0 si il n'y a pas erreur dans les paramètres.

3. Fin du traitement.

A = Identifiant du périphérique (0~3)

Carry = 0 si il n'y a pas erreur

Rendez la main à l'interpréteur (par un RET).

Numéro de l'instruction

| Registre A | Instruction d'appel     |
|------------|-------------------------|
| 0          | OPEN                    |
| 2          | CLOSE                   |
| 4          | Accès aléatoire         |
| 6          | Sortie séquentielle     |
| 8          | Entrée séquentielle     |
| 10         | Fonction LOC            |
| 12         | Fonction LOF            |
| 14         | Fonction EOF            |
| 16         | Fonction FPOS           |
| 18         | Caractère de sauvegarde |

## TEXT

Ce pointeur TEXT indique le début du programme Basic à exécuter automatiquement au démarrage du MSX. Le premier octet du programme est toujours zéro. Le programme ne peut avoir une taille de 16Ko maximum et doit se situer entre 08000h et 0BFFFh. Il doit aussi être format codé (« tokenized ») et non pas au format texte ASCII. De plus, les adresses correspondantes aux numéros de lignes du programme doivent indiquer les adresses de destination réelles dans le programme.

Méthode pour mettre un programme Basic en ROM :

1. Un programme Basic commence généralement à l'adresse 08000h. Il faut donc le décaler au minimum vers l'adresse 08012h pour pouvoir insérer l'entête de la ROM. Pour cela, entrer la ligne suivante sous Basic :

```
POKE &HF676,&H13: POKE &HF677,&H80: POKE &H8012,0: NEW
```

2. Charger le programme Basic à mettre en ROM en entrant l'instruction suivante.

```
LOAD"Name.BAS"
```

3. Puis sauvegarder le programme en entrant l'instruction suivante.

```
SAVE"Name2.BAS"
```

4. Ensuite, créer l'entête de la ROM avec un éditeur hexadécimal et ajouter 00 00.

```
000000h: 41 42 00 00 00 00 00 00 12 80 00 00 00 00 00 00
000010h: 00 00
```

5. Ajouter les données du fichier "Name2.BAS" à la suite par un copier/coller sans oublier de remplacer le premier octet FFh du programme par 00h. Ensuite, remplissez de 00h ou autres derrière le programme afin de faire un fichier 16384 octets précisément. Pour finir, sauvegardez par exemple sous le nom "Name.rom". Vous aurez juste à graver la ROM avec ce fichier pour créer votre cartouche.



Le système dispose aussi de variables pour les programmes en cartouche. Voici un descriptif celles-ci :

– **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 combinaisons.

Format de ces variables :

| 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 définis 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 autres bits de poids faible sont inutilisés. Ils sont en général à 0 mais pas toujours.

Voici un petit programme Basic qui donne un descriptif des cartouches insérées :

```
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~0FD88h)

SLTWRK est un tableau de 128 octets qui permet de stocker des données temporaires ou de réserver une zone de travail en RAM pour les applications en ROM. Ce tableau est composé de 8 octets par Slot (4 x 2 octets par plage mémoire) de la façon suivante.

|            | Partie basse | Partie Haute |                                                                       |
|------------|--------------|--------------|-----------------------------------------------------------------------|
| SLTWRK     |              |              | Indicateur de la zone de travail de l'application en Slot 0-0, page 0 |
| SLTWRK+2   |              |              | Indicateur de la zone de travail de l'application en Slot 0-0, page 1 |
| SLTWRK+3   |              |              | Indicateur de la zone de travail de l'application en Slot 0-0, page 2 |
| .          | .            | .            | .                                                                     |
| .          | .            | .            | .                                                                     |
| .          | .            | .            | .                                                                     |
| SLTWRK+124 |              |              | Indicateur de la zone de travail de l'application en Slot 3-3, page 2 |
| SLTWRK+126 |              |              | Indicateur de la zone de travail de l'application en Slot 3-3, page 3 |

Vous pouvez utiliser les deux octets correspondants à votre application pour y stocker des valeurs quelconques mais si vous avez besoin de plus de RAM que ces deux octets, utilisez-les pour y placer un numéro de Slot (au format indiqué plus bas) sur le premier octet ou un pointeur pour indiquer la zone de travail de l'application.

Le pointeur indique la première adresse de la zone à réserver. Celle-ci doit se trouver dans la zone libre dont le début est indiqué par la variable BOTTOM (0FC48h) et la fin par MEMSIZ (0F672h). (Attention : Prenez en compte que la taille pile est variable.)

Le numéro de Slot permet de réserver toute la RAM sur la plage 0 et 1 (0000h~7FFFh). Le format est le suivant.

| 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 de la RAM à réserver.

SS0 et SS1 = Numéro de Slot secondaire de la RAM à réserver.

RES = Bit réservé.

APP = 1 si la RAM est utilisée par une application ; 0 dans le cas contraire.

RMD = 1 si la RAM est utilisée par l'instruction CALL MEMINI ; 0 si ce n'est pas le cas.

F = 0 pour Slot primaire ; 1 pour Slot secondaire.

Effectuer le calcul suivant pour connaître l'emplacement réservé pour votre application en ROM.

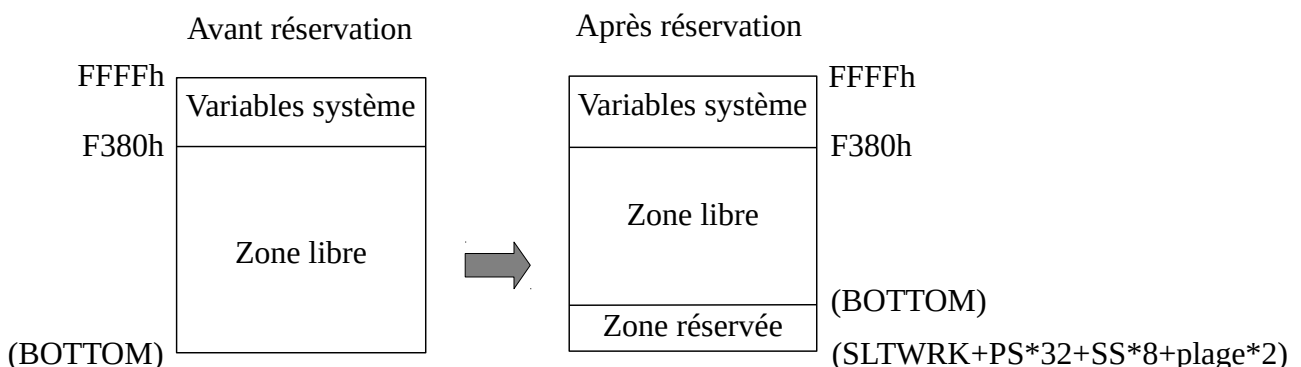
Adresse = SLTWRK + 32 \* Slot primaire + 8 \* Slot étendu + 2 \* plage mémoire (de 0 à 3)

Note : Le tableau n'est pas valable pour les 7 premiers octets car le système les réserve de la façon suivante.

SLTWRK = Numéro de Slot de la zone réservée par CALL MEMINI. (MSX2~)

SLTWRK+1 = 2 \* 3 octets pour les modes Kanji ou Hangul des MSX japonais ou coréens.  
(SLTWRK+4 indique le Slot de la ROM du Kanji Basic)

Voici un exemple de réservation d'une zone de travail sous un programme Basic :



Dans cet exemple, la variable correspondante à l'application en ROM doit contenir l'adresse 08000h comme pointeur. L'adresse à mettre à BOTTOM dépend de la taille de la zone de travail. N'oubliez pas non plus de changer la variable TXTTAB (0F676h) et d'écrire 0 à l'adresse indiquée par BOTTOM sinon la zone réservée sera écrasée lors du chargement d'un programme en Basic.

### 14.15 Trouver le Slot de la RAM depuis une ROM

Pour connaître le Slot de la Main-RAM lorsque le MSX a un disque installé, il suffit de lire les variables RAMAD0 à RAMAD3 (0F341h~0F344h) qui sont installées par la Disk-ROM. Celle-ci indique le Slot pour chacune des 4 plages mémoire. Par contre, lorsque votre programme est lancée depuis une ROM, étant donnée que la Disk-ROM n'est probablement pas encore installée, il faut donc créer une routine soit-même pour chercher la RAM.

Pour trouver la RAM sur la plage mémoire 3 (0C000h~0FFFFh), c'est relativement simple puisque le système a déjà sélectionné le Slot sur cette plage quelque soit le MSX. Il suffit donc de lire l'état des registre des Slot primaires et secondaire et ne garder que les bits utiles.

Voici un programme d'exemple qui renvoie le numéro de Slot (au format F000SSPP) de la RAM sur la plage mémoire 3 (0C000h~0FFFFh) dans le registre A :

```
RSLREG equ 00138h ; Lecture du registre des Slot primaires
EXPTBL equ 0FCC1h

SEEK_RAMAD3:
    call RSLREG
    and 0c0h ; Garde les bits 7 et 6 correspondant à la plage 3
    rlca
    rlca ; Décalage des bits 7 et 6 vers les bits 1 et 0
    ld e,a ; Garde la valeur du Slot primaire dans le reg. E
    ld a,(0ffffh) ; Lecture du registre des Slot secondaire du
    cpl ; Slot primaire sélectionné puis inversion des bits
    and 0c0h ; Garde les bits 7 et 6 correspondant à la plage 3
    rrca
    rrca
    rrca
    rrca ; Décalage des bits 7 et 6 vers les bits 3 et 2
    or e ; Récupère la valeur du Slot primaire
    ld hl,EXPTBL+3
    ld d,0
    add hl,de
    or (hl) ; Bit7=1 si le Slot est étendu
    ret
```

Ce programme est valable pour tous les MSX à partir de 16Ko et à condition que votre ROM ne démarre pas sur la plage mémoire 3 (0C000h~0FFFFh).

Pour trouver la RAM de la plage mémoire 2 (08000h~0BFFFh), nous pouvons employer la même

méthode à condition que la ROM ne démarre pas sur la plage 2.

Voici un programme d'exemple qui renvoie le numéro de Slot de la RAM sur la plage mémoire 2 (08000h~0BFFFh) dans le registre A :

```
RSLREG equ 00138h ; Lecture du registre des Slot primaires
EXPTBL equ 0FCC1h

SEEK_RAMAD2:
    call RSLREG
    and 030h ; Garde les bits 5 et 4 correspondant à la plage 2
    rlca
    rlca
    rlca
    rlca ; Décalage des bits 5 et 4 vers les bits 1 et 0
    ld e,a ; Garde la valeur du Slot primaire dans le registre E
    ld a,(0ffffh) ; Lecture du registre des Slot secondaire du
    cpl ; Slot primaire sélectionné puis inversion des bits
    and 030h ; Garde les bits 5 et 4 correspondant à la plage 3
    rrca
    rrca ; Décalage des bits 5 et 4 vers les bits 3 et 2
    or e ; Récupère la valeur du Slot primaire
    ld hl,EXPTBL+2
    ld d,0
    add hl,de
    or (hl) ; Bit7=1 si le Slot est étendu
    ret
```

Ce programme est valable pour tous les MSX à partir de 32Ko.

Pour trouver la RAM sur la plage 0 (0000h~3FFFh) ou 1 (4000h~7FFFh), c'est plus compliqué, il faut faire une recherche dans tous les Slot et de préférence dans l'ordre croissant.

Voici un exemple :

```
; Routine de recherche de la RAM sur la plage mémoire 0 ou 1
; Sortie: A=Numéro de Slot et Carry = 0, Carry = 1 si Ram non trouvée

EXPTBL equ 0fcc1h ; Indicateurs des Slot secondaires
RDSLT: equ 0000ch
WRSLT: equ 00014h
RAMSLT: equ 0c000h ; Slot de la RAM trouvée
KBUF: equ 0c001h ; Longueur 4 octets (0C001h~0C004h)

    org 04010h ; Adresse qui peut être 0000h~0BEFFh selon
               ; la plage sur laquelle la ROM se trouve

    call Ram_srch
    ld (RAMSLT),a
    ret

Ram_srch:
    ld b,4 ; Slot primaire
Ram_srch_loop:
    ld a,b
```

```

dec    a
xor    3
ld     (RAMSLT),a      ; Numéro du Slot secondaire en cours
ld     e,a

ld     hl,EXPTBL
ld     d,0
add    hl,de
ld     a,(hl)
ld     (KBUF),a        ; Stocke l'indicateur de Slot secondaire
exx

ld     a,(KBUF)         ; Restitue l'indicateur de Slot secondaire
rlca
ld     b,1
ld     a,(RAMSLT)
jr     nc,PrimSLT

ld     b,4              ;Slot secondaire
Ram_srch_loop2:
ld     a,b
dec    a
xor    3
rlca
rlca
ld     c,a
ld     a,(RAMSLT)
or     c
or     080h             ; Met le bit 7 à 1
PrimSLT:
ld     (KBUF+1),a
ld     hl,0000h         ; Plage 0 (mettre 4000h chercher la plage 1)
push   bc
call   RDSLT
ld     (KBUF+2),a
pop    bc
cp     041h
jr     nz,no_header    ; Saute si le premier octet = "A" (Rom?)

inc    hl
ld     a,(KBUF+1)
push   bc
call   RDSLT
pop    bc
dec    hl
cp     042h
jr     z,no_ram        ; Saute si le second octet <> "B"
no_header:
ld     a,(KBUF+1)
push   bc
call   RDSLT           ; Lit le premier octet
pop    bc
ld     e,041h
ld     a,(KBUF+1)
push   bc
call   WRSLT           ; Ecrit "A" au premier octet
pop    bc
ld     a,(KBUF+1)
push   bc
call   RDSLT           ; Lit le premier octet

```

```

        pop    bc
        cp     041h
        jr     z,Ram_found      ; Saute si le premier octet = "A"
no_ram:
        djnz   Ram_srch_loop2 ; Vers le Slot suivant si pas de RAM
        exx
        djnz   Ram_srch_loop  ; Vers le Slot suivant si pas de RAM
        scf
        ; Met Carry à 1
        ret

Ram_found:
        ld     a, (KBUF+2)
        ld     e, a
        ld     a, (KBUF+1)
        push   af
        or     080h
        call   WRSLT           ; Restitue le premier octet de la RAM
        pop    af             ; A=Slot de la Ram trouvée (sans le bit 7)
        or     a               ; Met Carry à 0
        ret

```

Ce programme est valable pour tous les MSX à partir de 64Ko. Cependant, il faut tenir compte qu'il est préférable, voir même nécessaire, de choisir la RAM interne pour le MSX Turbo R en mode R800. Si une extension de mémoire est insérée dans un port cartouche, elle sera sélectionnée car elle se trouvera dans un Slot inférieur.

Voilà vous avez trouvé de la mémoire vive. Maintenant, il faut faire attention au deux cas suivants :

1. Sur les MSX1 Toshiba HX-20, HX-21, HX-22 (Japanese) et les MSX2, un disque virtuel peut être implanté par l'instruction CALL MEMINI du Basic 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 un RAM e disque virtuel sous la forme FXXXSSPP en binaire.

F        indique le type de Slot. (0 pour Slot Primaire ; 1 pour Slot étendu.)

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 0005FH 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.

2. Une application utilise peut-être déjà la mémoire trouvée. Si tel est le cas, le bit 5 de SLTWRK

sera à 1. Si le bit est à 0, si vous décidez vous même d'utiliser la mémoire vive trouvée pour votre application, vous devez mettre le bit 5 de SLTWRK à 1.

Voici le format des variables du tableau SLTWRK :

| 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 = 1 si la RAM utilisée par une application ; 0 dans le cas contraire.

RMD = 1 si la RAM est utilisée par l'instruction CALL MEMINI ; 0 dans le cas contraire.

F = 0 pour Slot primaire ; 1 pour Slot secondaire.

Note : Voir l'explication sur SLTWRK à « [Développer un programme en cartouche](#) », chapitre 14.14 page 564 pour plus de précision.

## Annexes

### *A – Liste des labels par ordre alphabétique*

Les listes suivantes sont sous forme de label que vous pourrez récupérer pour les insérer directement dans vos programmes en assembleur.

Liste des labels pour les routines du Bios de la Main-ROM :

| <u>Nom</u> |     | <u>Adresse</u> | <u>Nom</u> |     | <u>Adresse</u> |
|------------|-----|----------------|------------|-----|----------------|
| BASRVN     | equ | 0002Bh         | DOWNC      | equ | 00108h         |
| BEEP       | equ | 000C0h         | DSPFNK     | equ | 000CFh         |
| BIGFIL     | equ | 0016Bh         | ENASCR     | equ | 00044h         |
| BREAKX     | equ | 000B7h         | ENASLT     | equ | 00024h         |
| CALATR     | equ | 00087h         | EOL        | equ | 00168h         |
| CALBAS     | equ | 00159h         | ERAFNK     | equ | 000CCh         |
| CALLF      | equ | 00030h         | EXTROM     | equ | 0015Fh         |
| CALPAT     | equ | 00084h         | FETCHC     | equ | 00114h         |
| CALSLT     | equ | 0001Ch         | FILVRM     | equ | 00056h         |
| CGTABL     | equ | 00004h         | FNKSB      | equ | 000C9h         |
| CHGCAP     | equ | 00132h         | FORMAT     | equ | 00147h         |
| CHGCPU     | equ | 00180h         | GETCPU     | equ | 00183h         |
| CHGCLR     | equ | 00062h         | GETVCP     | equ | 00150h         |
| CHGET      | equ | 0009Fh         | GETVC2     | equ | 00153h         |
| CHGMOD     | equ | 0005Fh         | GETYPR     | equ | 00028h         |
| CHGSND     | equ | 00135h         | GICINI     | equ | 00090h         |
| CHKNEW     | equ | 00165h         | GRPPRT     | equ | 0008Dh         |
| CHKSLZ     | equ | 00162h         | GSPSIZ     | equ | 0008Ah         |
| CHPUT      | equ | 000A2h         | GTASPC     | equ | 00126h         |
| CHRGTR     | equ | 00010h         | GTPAD      | equ | 000DBh         |
| CHSNS      | equ | 0009Ch         | GTPDL      | equ | 000DEh         |
| CKCNTC     | equ | 000BDh         | GTSTCK     | equ | 000D5h         |
| CLRSPPR    | equ | 00069h         | GTTRIG     | equ | 000D8h         |
| CLS        | equ | 000C3h         | INIFNK     | equ | 0003Eh         |
| CNVCHR     | equ | 000ABh         | INIGRP     | equ | 00072h         |
| DCOMPR     | equ | 00020h         | INIMLT     | equ | 00075h         |
| DISSCR     | equ | 00041h         | INITIO     | equ | 0003Bh         |



|        |     |        |        |     |        |
|--------|-----|--------|--------|-----|--------|
| INITXT | equ | 0006Ch | RSLREG | equ | 00138h |
| INIT32 | equ | 0006Fh | SCALXY | equ | 0010Eh |
| INLIN  | equ | 000B1h | SCANL  | equ | 0012Fh |
| ISCNTC | equ | 000BAh | SCANR  | equ | 0012Ch |
| ISFLIO | equ | 0014Ah | SETATR | equ | 0011Ah |
| KEYINT | equ | 00038h | SETC   | equ | 00120h |
| KILBUF | equ | 00156h | SETGRP | equ | 0005Eh |
| LDIRMV | equ | 00059h | SETMLT | equ | 00081h |
| LDIRVM | equ | 0005Ch | SETRD  | equ | 00050h |
| LEFTC  | equ | 000FFh | SETTXT | equ | 00078h |
| LFTQ   | equ | 000F6h | SETT32 | equ | 0007Bh |
| LPTOUT | equ | 000A5h | SETWRT | equ | 00053h |
| LPTSTT | equ | 000A8h | SNSMAT | equ | 00141h |
| MAPXYC | equ | 00111h | STARUP | equ | 00000h |
| MSXVER | equ | 0002Dh | STMOTR | equ | 000F3h |
| NMI    | equ | 00066h | STOREC | equ | 00117h |
| NRDVRM | equ | 00174h | STRTMS | equ | 00099h |
| NSETCX | equ | 00123h | SUBROM | equ | 0015Ch |
| NSETRD | equ | 0016Eh | SYNCHR | equ | 00008h |
| NSTWRT | equ | 00171h | TAPIN  | equ | 000E4h |
| NWRVRM | equ | 00177h | TAPIOF | equ | 000E7h |
| OUTLDP | equ | 0014Dh | TAPION | equ | 000E1h |
| OUTDO  | equ | 00018h | TAPOOF | equ | 000F0h |
| PCMPLY | equ | 00186h | TAPOON | equ | 000EAh |
| PCMREC | equ | 00189h | TAPOUT | equ | 000EDh |
| PNTINI | equ | 00129h | TDOWNC | equ | 0010Bh |
| PINLIN | equ | 000AEh | TOTEXT | equ | 000D2h |
| POSIT  | equ | 000C6h | TUPC   | equ | 00105h |
| PUTQ   | equ | 000F9h | UPC    | equ | 00102h |
| PYSDIO | equ | 00144h | VDP.DR | equ | 00006h |
| QINLIN | equ | 000B4h | VDP.DW | equ | 00007h |
| RDPSG  | equ | 00096h | WRRES  | equ | 0017Dh |
| RDRES  | equ | 0017Ah | WRTPSG | equ | 00093h |
| RDSLT  | equ | 0000Ch | WRSLT  | equ | 00014h |
| RDVDP  | equ | 0013Eh | WRTVDP | equ | 00047h |
| RDVRM  | equ | 0004Ah | WRTVRM | equ | 0004Dh |
| READC  | equ | 0011Dh | WRTVRM | equ | 00109h |
| RIGHTC | equ | 000FCh | WSLREG | equ | 0013Bh |

Liste des labels pour les routines du Bios de la Sub-ROM :

| <u>Nom</u> |     | <u>Adresse</u> | <u>Nom</u> |     | <u>Adresse</u> |
|------------|-----|----------------|------------|-----|----------------|
| ATRSCN     | equ | 00051h         | INIPLT     | equ | 00141h         |
| BASE       | equ | 00169h         | INITXTS    | equ | 000D5h         |
| BASEF      | equ | 0016Dh         | INIT32S    | equ | 000D9h         |
| BEEPS      | equ | 0017Dh         | INSLN0     | equ | 00125h         |
| BLTDM      | equ | 001A9h         | KNJPRT     | equ | 001BDh         |
| BLTDV      | equ | 001A1h         | KYKLOK     | equ | 00135h         |
| BLTMD      | equ | 001A5h         | LEFTCS     | equ | 000ADh         |
| BLTMV      | equ | 00199h         | MAPXYCS    | equ | 00091h         |
| BLTVM      | equ | 00195h         | NEWPAD     | equ | 001ADh         |
| BLTVD      | equ | 0019Dh         | NVBXFL     | equ | 000CDh         |
| BLTVV      | equ | 00191h         | NVBXLN     | equ | 000C9h         |
| BOXLIN     | equ | 00081h         | PAINT      | equ | 00069h         |
| CALATRS    | equ | 000FDh         | PROMPT     | equ | 00181h         |
| CALPATS    | equ | 000F9h         | PSET       | equ | 0006Dh         |
| CHGCLRS    | equ | 00111h         | PUTCHR     | equ | 00139h         |
| CHGMDP     | equ | 001B5h         | PUTSPR     | equ | 00151h         |
| CHGMODS    | equ | 000D1h         | RDVRMS     | equ | 0010Dh         |
| CLRSPRS    | equ | 000F5h         | READCS     | equ | 00095h         |
| CLRTXT     | equ | 00119h         | REDCLK     | equ | 001F5h         |
| CLSS       | equ | 00115h         | RIGHTCS    | equ | 000A5h         |
| COLOR      | equ | 00155h         | RSTPLT     | equ | 00145h         |
| DELLNO     | equ | 00121h         | SCALXYS    | equ | 0008Dh         |
| DOBOXF     | equ | 00079h         | SCANLS     | equ | 000C5h         |
| DOGRPH     | equ | 00085h         | SCANRS     | equ | 000C1h         |
| DOLINE     | equ | 0007Dh         | SCOPY      | equ | 0018Dh         |
| DOWNCS     | equ | 000B5h         | SCREEN     | equ | 00159h         |
| DSPFNKS    | equ | 0011Dh         | SDFSCR     | equ | 00185h         |
| GETPAT     | equ | 00105h         | SETATRS    | equ | 00099h         |
| GETPLT     | equ | 00149h         | SETCS      | equ | 0009Dh         |
| GETPUT     | equ | 001B1h         | SETGRPS    | equ | 000EDh         |
| GLINE      | equ | 00075h         | SETMLTS    | equ | 000F1h         |
| GRPPRTS    | equ | 00089h         | SETPAG     | equ | 0013Dh         |
| GSPSIZS    | equ | 00101h         | SETPLT     | equ | 0014Dh         |
| INIGRPS    | equ | 000DDh         | SETS       | equ | 00179h         |
| INIMLTS    | equ | 000E1h         | SETSCR     | equ | 00189h         |

|         |     |        |        |     |        |
|---------|-----|--------|--------|-----|--------|
| SETTXTS | equ | 000E5h | VDPF   | equ | 00165h |
| SETT32S | equ | 000E9h | VDPSTA | equ | 00131h |
| TDOWNCS | equ | 000B1h | VPEEK  | equ | 00175h |
| TLEFTC  | equ | 000A9h | VPOKE  | equ | 00171h |
| TRIGHT  | equ | 000A1h | WIDTHS | equ | 0015Dh |
| TUPCS   | equ | 000B9h | WRTCLK | equ | 001F9h |
| UPCS    | equ | 000BDh | WRTVDP | equ | 0012Dh |
| VDP     | equ | 00161h |        |     |        |

Liste des labels pour les variables système :

| Nom    | Adresse    | Nom    | Adresse     |
|--------|------------|--------|-------------|
| ACPAGE | equ 0FAF6h | CHRCNT | equ 0FAF9h  |
| ARG    | equ 0F847h | CLIKFL | equ 0FBD9h  |
| ARYTAB | equ 0F6C4h | CLIKSW | equ 0F3DBh  |
| ARYTA2 | equ 0F7B5h | CLINEF | equ 0F935h  |
| ASPCT1 | equ 0F40Bh | CLMLST | equ 0F3B2h  |
| ASPCT2 | equ 0F40Dh | CLOC   | equ 0F92Ah  |
| ASPECT | equ 0F931h | CLPRIM | equ 0F3B2h  |
| ATRBAS | equ 0F928h | CMASK  | equ 0F92Ch  |
| ATRBYT | equ 0F3F2h | CNPNTS | equ 0F936h  |
| AUTFLG | equ 0F6AAh | CNSDFG | equ 0F3DEh  |
| AUTINC | equ 0F6ADh | CODSAV | equ 0FBCCCh |
| AUTLIN | equ 0F6ABh | CONLO  | equ 0F66Ah  |
| AVCSAV | equ 0FAF7h | CONSAV | equ 0F668h  |
| BAKCLR | equ 0F3EAh | CONTXT | equ 0F666h  |
| BASROM | equ 0FBB1h | CONTYP | equ 0F669h  |
| BDRCLR | equ 0F3EBh | CPCNT  | equ 0F939h  |
| BOTTOM | equ 0FC48h | CPCNT8 | equ 0F93Bh  |
| BRDATR | equ 0FCB2h | CPLOTF | equ 0F938h  |
| BUF    | equ 0F55Eh | CRCSUM | equ 0h93Dh  |
| BUFMIN | equ 0F55Dh | CRTCNT | equ 0F3B1h  |
| CAPST  | equ 0FCABh | CS120  | equ 0F3FCh  |
| CASPRV | equ 0FCB1h | CS240  | equ 0F401h  |
| CENCNT | equ 0F933h | CSAVEA | equ 0F942h  |
| CGPBAS | equ 0F924h | CSAVEM | equ 0F944h  |
| CGPNT  | equ 0F91Fh | CSCLXY | equ 0F941h  |

|        |     |        |        |     |        |
|--------|-----|--------|--------|-----|--------|
| CSRSW  | equ | 0FCA9h | FBUFFR | equ | 0F7C5h |
| CSRX   | equ | 0F3DDh | FILNAM | equ | 0F866h |
| CSRY   | equ | 0F3DCh | FILNM2 | equ | 0F871h |
| CSTCNT | equ | 0F93Fh | FILTAB | equ | 0F860h |
| CSTYLE | equ | 0FCAAh | FLBMEM | equ | 0FCAEh |
| CURLIN | equ | 0F41Ch | FLGINP | equ | 0F6A6h |
| CXOFF  | equ | 0F945h | FNKFLG | equ | 0FBCEh |
| CYOFF  | equ | 0F947h | FNKSTR | equ | 0F87Fh |
| DAC    | equ | 0F7F6h | FNKSWI | equ | 0FBCDh |
| DATLIN | equ | 0F6A3h | FORCLR | equ | 0F3E9h |
| DATPTR | equ | 0F6C8h | FRCNEW | equ | 0F3F5h |
| DECCNT | equ | 0F7F4h | FRETOP | equ | 0F69Bh |
| DECTMP | equ | 0F7F0h | FSTPOS | equ | 0FBCAh |
| DECTM2 | equ | 0F7F2h | FUNACT | equ | 0F7BAh |
| DEFTBL | equ | 0F6CAh | GETPNT | equ | 0F3FAh |
| DEVICE | equ | 0FD99h | GRPACX | equ | 0FCB7h |
| DIMFLG | equ | 0F662h | GRPACY | equ | 0FCB9h |
| DORES  | equ | 0F664h | GRPATR | equ | 0F3CDh |
| DONUM  | equ | 0F665h | GRPCGP | equ | 0F3CBh |
| DOT    | equ | 0F6B5h | GRPCOL | equ | 0F3C9h |
| DPPAGE | equ | 0FAF5h | GRPHED | equ | 0FCA6h |
| DRWANG | equ | 0FCBDh | GRPNAM | equ | 0F3C7h |
| DRWFLG | equ | 0FCBBh | GRPPAT | equ | 0F3CFh |
| DRWSCL | equ | 0FCBCh | GXPOS  | equ | 0FCB3h |
| DSCPTR | equ | 0F699h | GYPOS  | equ | 0FCB5h |
| DSCTMP | equ | 0F698h | HEADER | equ | 0F40Ah |
| ENDBUF | equ | 0F660h | HIGH   | equ | 0F408h |
| ENDFOR | equ | 0F6A1h | HIMEM  | equ | 0FC4Ah |
| ENDPRG | equ | 0F40Fh | HOLD   | equ | 0F83Eh |
| ENSTOP | equ | 0FBB0h | HOLD2  | equ | 0F836h |
| ERRFLG | equ | 0F414h | HOLD8  | equ | 0F806h |
| ERRLIN | equ | 0F6B3h | INSFLG | equ | 0FCA8h |
| ERRTXT | equ | 0F6B7h | INTCNT | equ | 0FCA2h |
| ESCCNT | equ | 0FCA7h | INTFLG | equ | 0FC9Bh |
| EXBRSA | equ | 0FAF8h | INTVAL | equ | 0FCA0h |
| EXPTBL | equ | 0FCC1h | JIFFY  | equ | 0FC9Eh |
| FACLO  | equ | 0F7F8h | KANAMD | equ | 0FCADh |

|        |     |        |        |     |        |
|--------|-----|--------|--------|-----|--------|
| KANAST | equ | 0FCACH | NAMBAS | equ | 0F922h |
| KBFMIN | equ | 0F41Eh | NEWKEY | equ | 0FBE5h |
| KBUF   | equ | 0F41Fh | NLONLY | equ | 0F87Ch |
| KEYBUF | equ | 0FBF0h | NOFUNS | equ | 0F7B7h |
| LFPROG | equ | 0F954h | NTMSXP | equ | 0F417h |
| LINL32 | equ | 0F3AFh | NULBUF | equ | 0F862h |
| LINL40 | equ | 0F3AEh | OLDKEY | equ | 0FBDAh |
| LINLEN | equ | 0F3B0h | OLDLIN | equ | 0F6BEh |
| LINTTB | equ | 0FBB2h | OLDSCR | equ | 0FCB0h |
| LINWRK | equ | 0FC18h | OLDTXT | equ | 0F6C0h |
| LOGOPR | equ | 0FB02h | ONEFLG | equ | 0F6BBh |
| LOHADR | equ | 0F94Bh | ONELIN | equ | 0F6B9h |
| LOHCNT | equ | 0F94Dh | ONGSBF | equ | 0FBD8h |
| LOHDIR | equ | 0F94Ah | OPRTYP | equ | 0F664h |
| LOHMSK | equ | 0F949h | PADX   | equ | 0FC9Dh |
| LOW    | equ | 0F406h | PADY   | equ | 0FC9Ch |
| LOWLIM | equ | 0FCA4h | PARM1  | equ | 0F6E8h |
| LPTPOS | equ | 0F415h | PARM2  | equ | 0F750h |
| MAXDEL | equ | 0F92Fh | PATBAS | equ | 0F926h |
| MAXFIL | equ | 0F85Fh | PATWRK | equ | 0FC40h |
| MAXUPD | equ | 0F3ECh | PDIREC | equ | 0F953h |
| MCLFLG | equ | 0F958h | PLYCNT | equ | 0FB40h |
| MCLLEN | equ | 0FB3Bh | PRMFLG | equ | 0F7B4h |
| MCLPTR | equ | 0FB3Ch | PRMLN  | equ | 0F6E6h |
| MCLTAB | equ | 0F956h | PRMLN2 | equ | 0F74Eh |
| MEMSIZ | equ | 0F672h | PRMPRV | equ | 0F74Ch |
| MINDEL | equ | 0F92Dh | PRMSTK | equ | 0F6E4h |
| MINUPD | equ | 0F3EFh | PROCNM | equ | 0FD89h |
| MLTATR | equ | 0F3D7h | PRSCNT | equ | 0FB35h |
| MLTCGP | equ | 0F3D5h | PTRFIL | equ | 0F864h |
| MLTCOL | equ | 0F3D3h | PTRFLG | equ | 0F6A9h |
| MLTNAM | equ | 0F3D1h | PUTPNT | equ | 0F3F8h |
| MLTPAT | equ | 0F3D9h | QUEBAK | equ | 0F971h |
| MNROM  | equ | 0FCC1h | QUETAB | equ | 0F959h |
| MODE   | equ | 0FAFCh | QUEUEN | equ | 0FB3Eh |
| MOVCNT | equ | 0F951h | QUEUES | equ | 0F3F3h |
| MUSICF | equ | 0FB3Fh | RAWPRT | equ | 0F418h |

|         |     |        |        |     |        |
|---------|-----|--------|--------|-----|--------|
| RDPRIM  | equ | 0F380h | SAVENT | equ | 0FCBFh |
| REPCNT  | equ | 0F3F7h | SAVSP  | equ | 0FB36h |
| REQSTP  | equ | 0FC6Ah | SAVSTK | equ | 0F6B1h |
| RG0SAV  | equ | 0F3DFh | SAVTXT | equ | 0F6AFh |
| RG1SAV  | equ | 0F3E0h | SAVVOL | equ | 0FB39h |
| RG2SAV  | equ | 0F3E1h | SCNCNT | equ | 0F3F6h |
| RG3SAV  | equ | 0F3E2h | SCRMOD | equ | 0FCAFh |
| RG4SAV  | equ | 0F3E3h | SFTKEY | equ | 0FBEBh |
| RG5SAV  | equ | 0F3E4h | SKPCNT | equ | 0F94Fh |
| RG6SAV  | equ | 0F3E5h | SLTATR | equ | 0FCC9h |
| RG7SAV  | equ | 0F3E6h | SLTTBL | equ | 0FCC5h |
| RG8SAV  | equ | 0F3E7h | SLTWRK | equ | 0FD09h |
| RG9SAV  | equ | 0F3E8h | STATFL | equ | 0F3E7h |
| RG10SAV | equ | 0F3E9h | STKTOP | equ | 0F674h |
| RG11SAV | equ | 0F3EAh | STREND | equ | 0F6C6h |
| RG12SAV | equ | 0F3EBh | SUBFLG | equ | 0F6A5h |
| RG13SAV | equ | 0F3ECh | SWPTMP | equ | 0F7BCh |
| RG14SAV | equ | 0F3EDh | TEMP   | equ | 0F6A7h |
| RG15SAV | equ | 0F3EEh | TEMP2  | equ | 0F6BCh |
| RG16SAV | equ | 0F3EFh | TEMP3  | equ | 0F69Dh |
| RG17SAV | equ | 0FFF0h | TEMP8  | equ | 0F69Fh |
| RG18SAV | equ | 0FFF1h | TEMP9  | equ | 0F7B9h |
| RG19SAV | equ | 0FFF2h | TEMPPT | equ | 0F678h |
| RG20SAV | equ | 0FFF3h | TEMPST | equ | 0F67Ah |
| RG21SAV | equ | 0FFF4h | TOCNT  | equ | 0FB03h |
| RG22SAV | equ | 0FFF5h | TRCFLG | equ | 0F7C4h |
| RG23SAV | equ | 0FFF5h | TRGFLG | equ | 0F3E8h |
| ROMA    | equ | 0FAFAh | TRPTBL | equ | 0FC4Ch |
| RNDX    | equ | 0F857h | TTYPOS | equ | 0F661h |
| RSFCB   | equ | 0FB04h | TXTATR | equ | 0F3B9h |
| RSIQLN  | equ | 0FB06h | TXTCGP | equ | 0F3B7h |
| RS2IQ   | equ | 0FAF5h | TXTCOL | equ | 0F3B5h |
| RTPROG  | equ | 0F955h | TXTNAM | equ | 0F3B3h |
| RTYCNT  | equ | 0FC9Ah | TXTPAT | equ | 0F3BBh |
| RUNBNF  | equ | 0FCBEh | TXTTAB | equ | 0F676h |
| RUNFLG  | equ | 0F866h | T32ATR | equ | 0F3C3h |
| SAVEND  | equ | 0F87Dh | T32CGP | equ | 0F3C1h |

|        |     |        |
|--------|-----|--------|
| T32COL | equ | 0F3BFh |
| T32NAM | equ | 0F3BDh |
| T32PAT | equ | 0F3C5h |
| USRTAB | equ | 0F39Ah |
| VALTYP | equ | 0F663h |
| VARTAB | equ | 0F6C2h |
| VCBA   | equ | 0FB41h |
| VCBB   | equ | 0FB66h |
| VCBC   | equ | 0FB8Bh |
| VLZADR | equ | 0F419h |

|        |     |        |
|--------|-----|--------|
| VLZDAT | equ | 0F41Bh |
| VOICAQ | equ | 0F975h |
| VOICBQ | equ | 0F9F5h |
| VOICBC | equ | 0FA75h |
| VOICEN | equ | 0FB38h |
| WINWID | equ | 0FCA5h |
| WRPRIM | equ | 0F385h |
| XSAVE  | equ | 0FAFEh |
| YSAVE  | equ | 0FB00h |

Liste des labels des Hooks :

| <u>Nom</u> |     | <u>Adresse</u> |
|------------|-----|----------------|
| H.ATTR     | equ | 0FE1Ch         |
| H.BAKU     | equ | 0FEADh         |
| H.BINL     | equ | 0FE76h         |
| H.BINS     | equ | 0FE71h         |
| H.BUFL     | equ | 0FF8Eh         |
| H.CHGE     | equ | 0FDC2h         |
| H.CHPU     | equ | 0FDA4h         |
| H.CHRG     | equ | 0FF48h         |
| H.CLEA     | equ | 0FED0h         |
| H.CMD      | equ | 0FE0Dh         |
| H.COMP     | equ | 0FF57h         |
| H.COPY     | equ | 0FE08h         |
| H.CRDO     | equ | 0FEE9h         |
| H.CRUN     | equ | 0FF20h         |
| H.CRUS     | equ | 0FF25h         |
| H.CVD      | equ | 0FE49h         |
| H.CVI      | equ | 0FE3Fh         |
| H.CVS      | equ | 0FE44h         |
| H.DEVN     | equ | 0FEC1h         |
| H.DGET     | equ | 0FFE8h         |
| H.DIRD     | equ | 0FF11h         |
| H.DOGR     | equ | 0FEF3h         |
| H.DSKC     | equ | 0FEEh          |

| <u>Nom</u> |     | <u>Adresse</u> |
|------------|-----|----------------|
| H.DSKF     | equ | 0FE12h         |
| H.RETU     | equ | 0FF4Dh         |
| H.RSET     | equ | 0FE26h         |
| H.RSLF     | equ | 0FE8Fh         |
| H.RUNC     | equ | 0FECBh         |
| H.DSKI     | equ | 0FE17h         |
| H.DSKO     | equ | 0FDEFh         |
| H.DSPC     | equ | 0FDA9h         |
| H.DSPF     | equ | 0FDB3h         |
| H.EOF      | equ | 0FEA3h         |
| H.ERAC     | equ | 0FDAEh         |
| H.ERAF     | equ | 0FDB8h         |
| H.ERRF     | equ | 0FF02h         |
| H.ERRO     | equ | 0FFB1h         |
| H.ERP      | equ | 0FEFDh         |
| H.EVAL     | equ | 0FF70h         |
| H.FIEL     | equ | 0FE2Bh         |
| H.FILE     | equ | 0FE7Bh         |
| H.FILO     | equ | 0FE85h         |
| H.FINE     | equ | 0FF1Bh         |
| H.FING     | equ | 0FF7Ah         |
| H.FINI     | equ | 0FF16h         |
| H.FINP     | equ | 0FF5Ch         |

|         |     |         |         |     |        |
|---------|-----|---------|---------|-----|--------|
| H.FORM  | equ | 0FFACh  | H.MKI\$ | equ | 0FE30h |
| H.FPOS  | equ | 0FEA8h  | H.SETF  | equ | 0FE53h |
| H.FRET  | equ | 0FF9Dh  | H.SETS  | equ | 0FDF4h |
| H.FRME  | equ | 0FF66h  | H.SNGF  | equ | 0FF39h |
| H.FRQI  | equ | 0FF93h  | H.STKE  | equ | 0FEDAh |
| H.GEND  | equ | 0FEC6h  | H.MKS\$ | equ | 0FE35h |
| H.SAVD  | equ | 0FE94h  | H.NAME  | equ | 0FDF9h |
| H.SAVE  | equ | 0FE6Ch  | H.NEWS  | equ | 0FF3Eh |
| H.SCNE  | equ | 0FF98h  | H.NMI   | equ | 0FDD6h |
| H.SCRE  | equ | 0FFC0h  | H.NODE  | equ | 0FEB7h |
| H.GETP  | equ | 0FE4Eh  | H.NOFO  | equ | 0FE58h |
| H.GONE  | equ | 0FF43h  | H.NOTR  | equ | 0FF34h |
| H.INDS  | equ | 0FEA8h  | H.NTFL  | equ | 0FE62h |
| H.INIP  | equ | 0FDC7h  | H.NTFN  | equ | 0FF2Fh |
| H.INLI  | equ | 0FDE5h  | H.NTPL  | equ | 0FF6Bh |
| H.IPL   | equ | 0FE03h  | H.NULO  | equ | 0FE5Dh |
| H.ISFL  | equ | 0FEDFh  | H.OKNO  | equ | 0FF75h |
| H.ISMI  | equ | 0FF7Fh  | H.ONGO  | equ | 0FDEAh |
| H.ISRE  | equ | 0FF2Ah  | H.OUTD  | equ | 0FEE4h |
| H.KEYC  | equ | 0FDCCCh | H.PARD  | equ | 0FEB2h |
| H.KEYI  | equ | 0FD9Ah  | H.PHYD  | equ | 0FFA7h |
| H.KILL  | equ | 0FDFEh  | H.PINL  | equ | 0FDDb  |
| H.KYEA  | equ | 0FDD1h  | H.PLAY  | equ | 0FFC5h |
| H.LIST  | equ | 0FF89h  | H.POSD  | equ | 0FEBCh |
| H.LOC   | equ | 0FE99h  | H.PRGE  | equ | 0FEF8h |
| H.LOF   | equ | 0FE9Eh  | H.PRTF  | equ | 0FF52h |
| H.LOPD  | equ | 0FED5h  | H.PTRG  | equ | 0FFA2h |
| H.LPTO  | equ | 0FFB6h  | H.QINL  | equ | 0FDE0h |
| H.LPTS  | equ | 0FFBBh  | H.READ  | equ | 0FF07h |
| H.LSET  | equ | 0FE21h  | H.TIMI  | equ | 0FD9Fh |
| H.MAIN  | equ | 0FF0Ch  | H.TOTE  | equ | 0FDBDh |
| H.MERG  | equ | 0FE67h  | H.TRMN  | equ | 0FF61h |
| H.MKD\$ | equ | 0FE3Ah  | H.WIDT  | equ | 0FF84h |



## B – Les registres du VDP

Contenu des registres des VDP du MSX :

|               |     |     |     |     |     |     |     |     |
|---------------|-----|-----|-----|-----|-----|-----|-----|-----|
| Registre 0 :  | 0   | DG  | IE2 | IE1 | M5  | M4  | M3  | EV  |
| Registre 1 :  | 0   | BL  | IE0 | M1  | M2  | 0   | SI  | MAG |
| Registre 2 :  | 0   | N16 | N15 | N14 | N13 | N12 | N11 | N10 |
| Registre 3 :  | C13 | C12 | C11 | C10 | C9  | C8  | C7  | C6  |
| Registre 4 :  | 0   | 0   | F16 | F15 | F14 | F13 | F12 | F11 |
| Registre 5 :  | S14 | S13 | S12 | S11 | S10 | S9  | S8  | S7  |
| Registre 6 :  | 0   | 0   | P16 | P15 | P14 | P13 | P12 | P11 |
| Registre 7 :  | TC3 | TC2 | TC1 | TC0 | BD3 | BD2 | BD1 | BD0 |
| Registre 8 :  | MS  | LP  | TP  | CB  | VR  | 0   | SPD | BW  |
| Registre 9 :  | LN  | 0   | S1  | S0  | IL  | E0  | NT  | DC  |
| Registre 10 : | 0   | 0   | 0   | 0   | 0   | C16 | C15 | C14 |
| Registre 11 : | 0   | 0   | 0   | 0   | 0   | 0   | S16 | S15 |
| Registre 12 : | T23 | T22 | T21 | T20 | BC3 | BC2 | BC1 | BC0 |
| Registre 13 : | ON3 | ON2 | ON1 | ON0 | OF3 | OF2 | OF1 | OF0 |
| Registre 14 : | 0   | 0   | 0   | 0   | 0   | V16 | V15 | V14 |
| Registre 15 : | 0   | 0   | 0   | 0   | ST3 | ST2 | ST1 | ST0 |
| Registre 16 : | 0   | 0   | 0   | 0   | CC3 | CC2 | CC1 | CC0 |
| Registre 17 : | AI1 | 0   | RS5 | RS4 | RS3 | RS2 | RS1 | RS0 |
| Registre 18 : | V3  | V2  | V1  | V0  | H3  | H2  | H1  | H0  |
| Registre 19 : | IL7 | IL6 | IL5 | IL4 | IL3 | IL2 | IL1 | IL0 |
| Registre 20 : | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| Registre 21 : | 0   | 0   | 1   | 1   | 1   | 0   | 1   | 1   |
| Registre 22 : | 0   | 0   | 0   | 0   | 0   | 1   | 0   | 1   |
| Registre 23 : | DO7 | DO6 | DO5 | DO4 | DO3 | DO2 | DO1 | DO0 |

|               |   |     |     |     |     |     |     |     |
|---------------|---|-----|-----|-----|-----|-----|-----|-----|
| Registre 25 : | 0 | CMD | VDS | YAE | YJK | WTE | MSK | SP2 |
| Registre 26 : | 0 | 0   | HO8 | HO7 | HO6 | HO5 | HO4 | HO3 |
| Registre 27 : | 0 | 0   | 0   | 0   | 0   | HO2 | HO1 | HO0 |

} V9958

|               |     |     |     |     |     |     |     |     |
|---------------|-----|-----|-----|-----|-----|-----|-----|-----|
| Registre 32 : | SX7 | SX6 | SX5 | SX4 | SX3 | SX2 | SX1 | SX0 |
| Registre 33 : | 0   | 0   | 0   | 0   | 0   | 0   | 0   | SX8 |
| Registre 34 : | SY7 | SY6 | SY5 | SY4 | SY3 | SY2 | SY1 | SY0 |
| Registre 35 : | 0   | 0   | 0   | 0   | 0   | 0   | SY9 | SY8 |

|               |     |     |     |     |     |     |     |     |
|---------------|-----|-----|-----|-----|-----|-----|-----|-----|
| 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 des bits :

| 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 = Sprite agrandis ; 0 = Sprite 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 Sprite (6 bits de poids fort sur 17)      |
| RS5 ~ RS0  | 17 | N° de registre lors d'un adressage indirect                             |
| SI         | 1  | 1 = Sprite 16*16 ; 0 = Sprite 8*8                                       |
| SPD        | 8  | 1 = Sprite interdits ; 0 = Sprite 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 de la table des attributs de Sprite (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 |

## C – Les cartes de la mémoire vidéo

La carte mémoire vidéo à l'initialisation pour chaque mode graphique :

### SCREEN 0 en 40 colonne                      MSX1 à MSX turbo R

|                                    |               |                     |
|------------------------------------|---------------|---------------------|
| Table des formes de caractère :    | 00800h~00FFFh | (2048 octets)       |
| Table des palettes de couleur :    | 00400h~0041Fh | (32 octets) (MSX2~) |
| Table des positions de caractère : | 00000h~003BFh | (960 octets)        |

### SCREEN 0 en 80 colonne                      MSX2 à MSX turbo R

|                                    |               |                           |
|------------------------------------|---------------|---------------------------|
| Table des formes de caractère :    | 01000h~017FFh | (2048 octets)             |
| Table des palettes de couleur :    | 00F00h~00F1Fh | (32 octets) (MSX2~)       |
| Table des couleurs en 24 lignes :  | 00800h~008EFh | (240 octets - 24 lignes)  |
| Table des positions de caractère : | 00000h~0077Fh | (1920 octets - 24 lignes) |

### SCREEN 1                                              MSX1 à MSX turbo R

|                                    |               |                     |
|------------------------------------|---------------|---------------------|
| Table des formes de Sprite :       | 03800h~03FFFh | (2048 octets)       |
| Table des palettes de couleur :    | 02020h~0203Fh | (32 octets) (MSX2~) |
| Table des couleurs de caractère :  | 02000h~0201Fh | (32 octets)         |
| Table des attributs de Sprite :    | 01B00h~01B7Fh | (128 octets)        |
| Table des positions de caractère : | 01800h~01AFFh | (768 octets)        |
| Table des formes de caractère :    | 00000h~005FFh | (2048 octets)       |

### SCREEN 2                                              MSX1 à MSX turbo R

|                                 |               |                     |
|---------------------------------|---------------|---------------------|
| Table des formes de Sprite :    | 03800h~03FFFh | (2048 octets)       |
| Table des palettes de couleur : | 02020h~0203Fh | (32 octets) (MSX2~) |
| Table des couleurs de motif :   | 02000h~037FFh | (6144 octets)       |
| Table des attributs de Sprite : | 01B00h~01B7Fh | (128 octets)        |
| Table des positions de motif :  | 01800h~01AFFh | (768 octets)        |
| Table des formes de motif :     | 00000h~017FFh | (6144 octets)       |

### SCREEN 3

#### MSX1 à MSX turbo R

|                                 |               |               |
|---------------------------------|---------------|---------------|
| Table des formes de Sprite :    | 03800h~03FFFh | (2048 octets) |
| Table de la palette (MSX2~) :   | 02020h~0203Fh | (32 octets)   |
| Table des attributs de Sprite : | 01B00h~01B7Fh | (128 octets)  |
| Table des positions de motif :  | 00800h~00AFFh | (768 octets)  |
| Table des formes de motif :     | 00000h~005FFh | (2048 octets) |

### SCREEN 4

#### MSX2 à MSX turbo R

|                                 |               |               |
|---------------------------------|---------------|---------------|
| Table des formes de Sprite :    | 03800h~03FFFh | (2048 octets) |
| Table des couleurs de motif :   | 02000h~037FFh | (6144 octets) |
| Table des palettes de couleur : | 01E80h~01E9Fh | (32 octets)   |
| Table des attributs de Sprite : | 01E00h~01E7Fh | (128 octets)  |
| Table des couleurs de Sprite :  | 01C00h~0D7FFh | (512 octets)  |
| Table des positions de motif :  | 01800h~01AFFh | (768 octets)  |
| Table des formes de motif :     | 00000h~017FFh | (6144 octets) |

### SCREEN 5

#### MSX2 à MSX turbo R

|                                 |               |                             |
|---------------------------------|---------------|-----------------------------|
| Table des formes de Sprite :    | 07800h~05FFFh | (2048 octets)               |
| Table des palettes de couleur : | 07680h~0769Fh | (32 octets)                 |
| Table des attributs de Sprite : | 07600h~0767Fh | (128 octets)                |
| Table des couleurs de Sprite :  | 07400h~075FFh | (512 octets)                |
| Table Bitmap :                  | 00000h~069FFh | (27136 octets à 212 lignes) |

### SCREEN 6

#### MSX2 à MSX turbo R

|                                 |               |                             |
|---------------------------------|---------------|-----------------------------|
| Table des formes de Sprite :    | 07800h~05FFFh | (2048 octets)               |
| Table des palettes de couleur : | 07680h~0769Fh | (32 octets)                 |
| Table des attributs de Sprite : | 07600h~0767Fh | (128 octets)                |
| Table des couleurs de Sprite :  | 07400h~075FFh | (512 octets)                |
| Table Bitmap :                  | 00000h~069FFh | (27136 octets à 212 lignes) |

## SCREEN 7

MSX2 à MSX turbo R

|                                 |               |                             |
|---------------------------------|---------------|-----------------------------|
| Table des palettes de couleur : | 0FA80h~0FA9Fh | (32 octets)                 |
| Table des attributs de Sprite : | 0FA00h~0FA7Fh | (128 octets)                |
| Table des couleurs de Sprite :  | 0F800h~0F9FFh | (512 octets)                |
| Table des formes de Sprite :    | 0F000h~0F7FFh | (2048 octets)               |
| Table Bitmap :                  | 00000h~0D3FFh | (54272 octets à 212 lignes) |

## SCREEN 8

MSX2 à MSX turbo R

|                                 |               |                             |
|---------------------------------|---------------|-----------------------------|
| Table des palettes de couleur : | 0FA80h~0FA9Fh | (32 octets)                 |
| Table des attributs de Sprite : | 0FA00h~0FA7Fh | (128 octets)                |
| Table des couleurs de Sprite :  | 0F800h~0F9FFh | (512 octets)                |
| Table des formes de Sprite :    | 0F000h~0F7FFh | (2048 octets)               |
| Table Bitmap :                  | 00000h~0D3FFh | (54272 octets à 212 lignes) |

## SCREEN 9

MSX2 coréen

Le mode SCREEN 9 est en fait le mode graphique 4 ou 5 selon le nombre de colonne. Voir les descriptions du SCREEN 5 ou SCREEN 6. Notez que les Sprite ne sont pas inutilisés par le Basic.

## SCREEN 10 / 11

MSX2+ et MSX turbo R

|                                 |               |                             |
|---------------------------------|---------------|-----------------------------|
| Table des palettes de couleur : | 0FA80h~0FA9Fh | (32 octets)                 |
| Table des attributs de Sprite : | 0FA00h~0FA7Fh | (128 octets)                |
| Table des couleurs de Sprite :  | 0F800h~0F9FFh | (512 octets)                |
| Table des formes de Sprite :    | 0F000h~0F7FFh | (2048 octets)               |
| Table YEA :                     | 00000h~0D3FFh | (54272 octets à 212 lignes) |

## SCREEN 12

MSX2+ et MSX turbo R

|                                 |               |                             |
|---------------------------------|---------------|-----------------------------|
| Table des palettes de couleur : | 0FA80h~0FA9Fh | (32 octets)                 |
| Table des attributs de Sprite : | 0FA00h~0FA7Fh | (128 octets)                |
| Table des couleurs de Sprite :  | 0F800h~0F9FFh | (512 octets)                |
| Table des formes de Sprite :    | 0F000h~0F7FFh | (2048 octets)               |
| Table YJK :                     | 00000h~0D3FFh | (54272 octets à 212 lignes) |

## ***D – Codes des caractères MSX***

Les codes de caractères MSX sont tous basés sur le code ASCII 7 bits. Les 32 premiers et le 7Fh (en bleu dans les tableaux) sont des codes de contrôle qui ont les effets suivants lorsqu'ils sont rencontrés lors d'une entrée ou une sortie de caractère.

Beep = Emet le son « bip ».

BS = Déplace le curseur vers la gauche et efface le caractère sur son passage.

CLS = Efface l'écran.

CTRL+B = Place le curseur sur le mot précédent.

CTRL+C = Quitter une commande (MSX-DOS)

CTRL+D = Free.

CTRL+E = Efface la ligne à droite du curseur.

CTRL+F = Place le curseur sur le mot suivant.

CTRL+N = Désactive l'écho vers l'imprimante et place le curseur en fin de ligne.

CTRL+P = Activer l'écho vers l'imprimante. (MSX-DOS)

CTRL+N = Désactiver l'écho vers l'imprimante. (MSX-DOS)

CTRL+O = Free.

CTRL+P = Free.

CTRL+Q = Free.

CTRL+S = Stopper temporairement l'affichage des caractères à l'écran. (MSX-DOS)

CTRL+T = Free.

CTRL+U = Efface la ligne où se trouve le curseur.

CTRL+V = Free.

CTRL+W = Free.

CTRL+X = Free.

CTRL+Y = Free.

CTRL+Z = Free.

DEL = Supprimer le caractère à gauche du curseur et déplace les suivants d'un caractère jusqu'au code null ou de fin de ligne.

GRAPH = Permet d'afficher un caractère graphique étendu.

HOME = Positionnement du curseur tout en haut à gauche.

LF = Descend le curseur d'une ligne.

INS = Passage en mode insertion / écraser.

RET = Place le curseur au début de la ligne suivante.

SEL = Touche de sélection.

TAB = Déplace le curseur vers la colonne suivante.

↑ = Curseur vers le haut.

→ = Curseur vers la droite.

← = Curseur vers la gauche.

↓ = Curseur vers le bas.



Les caractères suivants sont disposés de la façon suivante selon la région où a été vendu l'ordinateur.

Jeu de caractères des 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     | HONE | 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      | ä      | ç      | è   | ë      | è      | ï    | î   | ï   | ä      | å      |
| 9 | é      | æ      | Æ      | ø      | ö      | ò      | ù      | û      | ü   | ö      | Ü      | ¢    | £   | ¥   | ₣      | f      |
| A | á      | í      | ó      | ú      | ñ      | ñ      | ä      | ö      | ç   | ı      | ı      | ½    | ¼   | ı   | <      | >      |
| B | ä      | ä      | ï      | ï      | ö      | ö      | ö      | ü      | ü   | ¼      | ı      | ı    | ı   | ı   | ı      | ı      |
| C | ■      | ■      | ■      | ■      | ■      | ■      | ■      | ■      | ■   | ■      | ■      | ■    | ■   | ■   | ■      | ■      |
| D | ■      | ■      | ■      | ■      | ■      | ■      | ■      | ■      | ■   | ■      | ■      | ■    | ■   | ■   | ■      | ■      |
| E | α      | β      | Γ      | Π      | Σ      | σ      | μ      | ı      | ı   | ı      | ı      | ı    | ı   | ı   | ı      | ı      |
| F | ≡      | ±      | ≥      | ≤      | ı      | J      | ÷      | ×      | ×   | ×      | ×      | ×    | ×   | ×   | ×      | ×      |

|   | 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 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 en mode texte, chacun doit être précédé du caractère 01h. Ils sont aussi spécifiques au MSX occidentaux.

|   | 0      | 1      | 2      | 3      | 4      | 5      | 6      | 7      | 8   | 9      | A      | B    | C   | D   | E      | F      |
|---|--------|--------|--------|--------|--------|--------|--------|--------|-----|--------|--------|------|-----|-----|--------|--------|
| 0 | NUL    | GRAPH  | CTRL E | CTRL C | CTRL D | CTRL E | CTRL F | BEEP   | BS  | TAB    | LF     | HOME | CLS | RET | CTRL W | 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      | 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 | タ      | チ      | ツ      | テ      | ト      | ナ      | ニ      | ヌ      | ネ   | ノ      | ハ      | ヒ    | フ   | ヘ   | ホ      | マ      |
| D | ミ      | ム      | メ      | モ      | ヤ      | ユ      | ヨ      | ラ      | リ   | ル      | レ      | ロ    | ワ   | ン   | ゝ      | °      |
| E | た      | ち      | つ      | て      | と      | な      | に      | ぬ      | ね   | の      | は      | 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 | 一 | 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 en mode texte, chacun doit être précédé du caractère 01h. Ils sont spécifiques au MSX japonais.

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.

## 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 W | CTRL 0 |
| 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 | é      | â      | à      | ç      | ê      | ë      | è      | ï      | î   | ó      | ô      | ù    | û   | ö   | °      | no     |
| 9 | p      | q      | r      | s      | t      | u      | v      | w      | x   | y      | z      | {    |     | }   | ~      | DEL    |
| A |        | !      | "      | #      | \$     | %      | &      | '      | (   | )      | *      | +    | ,   | -   | .      | /      |
| B | +      | 1      | 2      | 3      | 4      | 5      | 6      | 7      | 8   | 9      | :      | ;    | <   | =   | >      | ?      |
| C | @      | ء      | آ      | أ      | ق      | ك      | ل      | ب      | ة   | ت      | ث      | ج    | ح   | خ   | د      |        |
| D | ذ      | ر      | ز      | س      | ش      | ص      | ض      | ط      | ظ   | ع      | ف      | ]    | \   | [   | ^      | _      |
| E | —      | ف      | ق      | ك      | ل      | ب      | ة      | ت      | ث   | ج      | ح      | خ    | د   | ذ   | ر      | ز      |
| F | .      | ء      | آ      | أ      | ق      | ك      | ل      | ب      | ة   | ت      | ث      | ج    | ح   | خ   | د      | DEL    |

- 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.

## E – Exemples de matrices de clavier

Le caractère à droite correspond à celui obtenu avec la touche SHIFT maintenue.

Clavier « AZERTY » des Philips NMS-8250/55/80 et Sony HB-F500/700/900 :

| 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    | Accents | = +   | : /   | ; .   | # £      | 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.

Les Philips ont la touche « Cap lock » à la place de « Accents ».

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 » du Sony HB-501P :

| 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    | Accents | / ?   | . >   | , <   | £ ~      | ' "   |
| 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   | CODE  | CAPS  | GRAPH | CTRL     | SHIFT |
| 07    | RET   | SELECT | BS      | STOP  | TAB   | ESC   | F5 F10   | F4 F9 |
| 08    | →     | ↓      | ↑       | ←     | DEL   | 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    | →     | ↓      | ↑     | ←     | DEL   | INS   | Home CLS | SPACE |
| 09    | 4     | 3      | 2     | 1     | 0     | /     | +        | *     |
| 10    | .     | ,      | -     | 9     | 8     | 7     | 6        | 5     |
| 11    |       |        |       |       | 取消    |       | 実行       |       |

Les lignes 09 et 10 sont celles du pavé numérique.

« 取消 » veut dire « Annuler » et « 実行 » veut dire « Exécuter ».

## Clavier russe du Yamaha KYBT1 YIS-503II et KYBT2 YIS-503IIR :

| 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 Ъ    | Z 3   | ] } Щ | T Т   | X Ь   | D Д      | L Л   |
| 05    | Q Я   | N Н    | ~ Ч   | C Ц   | M М   | G Г   | E Е      | 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    | →     | ↓      | ↑     | ←     | DEL   | 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 Ъ    | Z 3   | ] } Щ | T Т   | X Ь   | D Д      | L Л   |
| 05    | Q Я   | N Н    | ~ Ч   | C Ц   | M М   | G Г   | E Е      | 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    | →     | ↓      | ↑     | ←     | DEL   | 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 Ъ    | Z 3   | ] Щ   | T Т   | X Ь   | D Д      | L Л   |
| 05    | Q Я   | N Н    | Ч     | C Ц   | M М   | G Г   | E Е      | 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    | →     | ↓      | ↑     | ←     | DEL   | INS   | Home CLS | SPACE |
| 09    | 4     | 3      | 2     | 1     | 0     | /     | +        | *     |
| 10    | .     | ,      | -     | 9     | 8     | 7     | 6        | 5     |

Les deux dernières lignes sont celles du pavé numérique.

## ***F – 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 de 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.

## G – Les ports d'entrée/sortie

Les ports d'entrée/sortie permettent de faire des accès directs aux matériels qui composent le MSX. Sur MSX1, il était interdit de faire des accès directs pour garder la compatibilité. Puis la norme s'est assouplie à la sortie des MSX2 en tolérant les accès au PPI et au PSG. Ensuite la norme des MSX2+ conseillait même de faire des accès directs au VDP pour gagner en rapidité.

Dans les faits, les accès au PPI, au PSG et au VDP ne causent aucun problème sur toutes les générations de MSX, à condition de faire attention à certains timings qui sont un peu différents selon la génération du VDP par exemple. Cela est d'autant plus vrai lorsque le CPU a un « mode turbo ».

Le Z80 ne peut gérer que 256 ports d'entrée/sortie maximum. La norme MSX a réservé tous les ports de 0 à 63 (03Fh) pour les utilisateurs et ceux au dessus pour les constructeurs de matériel officiel.

Voici une liste non-exhaustive des matériels utilisant les ports d'entrée/sortie.

| Ports    | Matériel                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |      |                |                |                           |                           |                         |                        |      |      |          |      |      |                |                |                           |                           |                   |                     |  |      |      |      |      |      |      |      |      |          |   |   |   |   |                  |                  |                         |                        |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|----------------|----------------|---------------------------|---------------------------|-------------------------|------------------------|------|------|----------|------|------|----------------|----------------|---------------------------|---------------------------|-------------------|---------------------|--|------|------|------|------|------|------|------|------|----------|---|---|---|---|------------------|------------------|-------------------------|------------------------|
| 00h~03h  | Interface MIDI JVC « MSX MIDI interface. »<br>Port 00h : Bit 1 = Activer / désactiver l'horloge (écriture), bit 0 = Signal de l'horloge.<br>Port 02h : Accès au registre de contrôle / Lecture du registre de statut. (MC6850 ACIA)<br>Port 03h : Données. (MC6850 ACIA)<br>Note : ces ports se reflètent de 08h à 0Bh.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |      |                |                |                           |                           |                         |                        |      |      |          |      |      |                |                |                           |                           |                   |                     |  |      |      |      |      |      |      |      |      |          |   |   |   |   |                  |                  |                         |                        |
| 00h~01h  | Prise MIDI (sortie) du Music Module.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |      |                |                |                           |                           |                         |                        |      |      |          |      |      |                |                |                           |                           |                   |                     |  |      |      |      |      |      |      |      |      |          |   |   |   |   |                  |                  |                         |                        |
| 00h~01h  | <div>Cartouche Robot Arm SVI-2000C. (Ecriture)</div> <table><tr><td></td><td>bit7</td><td>bit6</td><td>bit5</td><td>bit4</td><td>bit3</td><td>bit2</td><td>bit1</td><td>bit0</td></tr><tr><td>Port 00h</td><td>-</td><td>-</td><td>Fermer la main</td><td>Ouvrir la main</td><td>Rotation du bras à droite</td><td>Rotation du bras à gauche</td><td>Bras vers l'avant</td><td>Bras vers l'arrière</td></tr></table> <div>Bits 0 et 1 = Axe 2<br/>Bits 2 et 3 = Axe 1<br/>Bits 4 et 5 = Axe 5</div> <table><tr><td></td><td>bit7</td><td>bit6</td><td>bit5</td><td>bit4</td><td>bit3</td><td>bit2</td><td>bit1</td><td>bit0</td></tr><tr><td>Port 01h</td><td>-</td><td>-</td><td>-</td><td>-</td><td>Poignet à droite</td><td>Poignet à gauche</td><td>Avant-bras vers le haut</td><td>Avant-bras vers le bas</td></tr></table> <div>Bits 0 et 1 = Axe 3<br/>Bits 2 et 3 = Axe 4<br/>Attention : Ne pas mettre les deux bits d'un même axe à 1 car cela peut endommager les transistors. Veillez aussi à stopper les moteurs avant qu'ils arrivent en fin de course. Cela peut abîmer les engrenages.</div> |      | bit7           | bit6           | bit5                      | bit4                      | bit3                    | bit2                   | bit1 | bit0 | Port 00h | -    | -    | Fermer la main | Ouvrir la main | Rotation du bras à droite | Rotation du bras à gauche | Bras vers l'avant | Bras vers l'arrière |  | bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 | Port 01h | - | - | - | - | Poignet à droite | Poignet à gauche | Avant-bras vers le haut | Avant-bras vers le bas |
|          | bit7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | bit6 | bit5           | bit4           | bit3                      | bit2                      | bit1                    | bit0                   |      |      |          |      |      |                |                |                           |                           |                   |                     |  |      |      |      |      |      |      |      |      |          |   |   |   |   |                  |                  |                         |                        |
| Port 00h | -                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | -    | Fermer la main | Ouvrir la main | Rotation du bras à droite | Rotation du bras à gauche | Bras vers l'avant       | Bras vers l'arrière    |      |      |          |      |      |                |                |                           |                           |                   |                     |  |      |      |      |      |      |      |      |      |          |   |   |   |   |                  |                  |                         |                        |
|          | bit7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | bit6 | bit5           | bit4           | bit3                      | bit2                      | bit1                    | bit0                   |      |      |          |      |      |                |                |                           |                           |                   |                     |  |      |      |      |      |      |      |      |      |          |   |   |   |   |                  |                  |                         |                        |
| Port 01h | -                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | -    | -              | -              | Poignet à droite          | Poignet à gauche          | Avant-bras vers le haut | Avant-bras vers le bas |      |      |          |      |      |                |                |                           |                           |                   |                     |  |      |      |      |      |      |      |      |      |          |   |   |   |   |                  |                  |                         |                        |
| 00h~01h  | <div>Cartouche « Sensor Kid ».</div> <div>Écriture :</div> <table><tr><td></td><td>bit7</td><td>bit6</td><td>bit5</td><td>bit4</td><td>bit3</td><td>bit2</td><td>bit1</td><td>bit0</td></tr><tr><td>Port 00h</td><td>OUT1</td><td>OUT0</td><td>-</td><td>CS</td><td>AI</td><td></td><td>RS</td><td>CLK</td></tr></table> <div>CLK = 1 pour spécifier la lecture du bit suivant de l'entrée analogique sélectionnée ;<br/>RS (Range Selection) = Signal vers la broche RS de la puce MB4052 ;<br/>AI = Numéro de l'entrée analogique à sélectionner. (0 = Capteur de luminosité, 1 = Capteur sonore, 2 = Thermomètre, 3 = Inutilisé) ;</div>                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |      | bit7           | bit6           | bit5                      | bit4                      | bit3                    | bit2                   | bit1 | bit0 | Port 00h | OUT1 | OUT0 | -              | CS             | AI                        |                           | RS                | CLK                 |  |      |      |      |      |      |      |      |      |          |   |   |   |   |                  |                  |                         |                        |
|          | bit7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | bit6 | bit5           | bit4           | bit3                      | bit2                      | bit1                    | bit0                   |      |      |          |      |      |                |                |                           |                           |                   |                     |  |      |      |      |      |      |      |      |      |          |   |   |   |   |                  |                  |                         |                        |
| Port 00h | OUT1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | OUT0 | -              | CS             | AI                        |                           | RS                      | CLK                    |      |      |          |      |      |                |                |                           |                           |                   |                     |  |      |      |      |      |      |      |      |      |          |   |   |   |   |                  |                  |                         |                        |

|         |                                                                                                                                                                                                                                                                                                                                                 |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|         | <p>CS (Chip Select) = Signal vers la broche CS de la puce MB4052 ;<br/> OUT0 et OUT1 = Port 0 et port 1 de la prise Sortie à 0V ou 5V.</p> <p><b>Lecture :</b><br/> Port 00h = Le bit 0 donne l'état du bit actuel de l'entrée analogique sélectionnée.<br/> Port 01h = Le bit 0 et 1 indique l'état du port 0 et port 1 de la prise Sortie</p> |
| 02h~03h | Interface FAC MIDI. (Ces ports se reflètent jusqu'à 07h)                                                                                                                                                                                                                                                                                        |
| 04h~05h | Entrée MIDI du Music Module.                                                                                                                                                                                                                                                                                                                    |
| 04h~05h | Ports d'accès à la puce audio SAA1099 de la cartouche Soundstar de Supersoniqs.<br>Port 4 pour les données et le 5 pour spécifier le registre dans lequel écrire.                                                                                                                                                                               |
| 0Ah     | DAC du Music Module.                                                                                                                                                                                                                                                                                                                            |
| 0Fh     | MegaRam Zemina. (Cartouches Black Box, Deluxe Box et Golden Box)<br>Bit 4 et 5 = 0 et 1 pour mode RAM, 1 et 0 pour mode ROM (par défaut).<br>Bit 6 et 7 = 0 et 1 pour Mapper 8k, 1 et 0 pour Mapper 16k.                                                                                                                                        |
| 10h~11h | PSG de plusieurs MegaFlashROM basées sur un FPGA.                                                                                                                                                                                                                                                                                               |
| 14h~17h |                                                                                                                                                                                                                                                                                                                                                 |
| 18h~19h | Lecteur de code-barre Philips NMS-1170/20                                                                                                                                                                                                                                                                                                       |
| 20h~28h | Modem Philips NMS1251 (Ports paramétrables soit sur la plage 20h~28h, soit sur 30h~38h à l'aide de cavaliers)                                                                                                                                                                                                                                   |
| 20h~28h | Modem Miniware M4000 (Ports paramétrables soit sur la plage 20h~28h, soit sur 30h~38h à l'aide de cavaliers)                                                                                                                                                                                                                                    |
| 21h~27h | MP3 Player Sunrise                                                                                                                                                                                                                                                                                                                              |
| 27h~2Fh | Serial interface Philips NMS 1210/1211/1212 (Ports paramétrables soit sur la plage 27h~2Fh, soit sur 37h~3Fh à l'aide de cavaliers)                                                                                                                                                                                                             |
| 28h~29h | Interface Ethernet DenYoNet                                                                                                                                                                                                                                                                                                                     |
| 2Ah~2Bh | Registres de paramétrage de la cartouche PlaySoniq                                                                                                                                                                                                                                                                                              |
| 30h~32h | Interface Beer IDE                                                                                                                                                                                                                                                                                                                              |
| 30h~38h | Modem Philips NMS1251 (Ports paramétrables soit sur la plage 20h~28h, soit sur 30h~38h à l'aide de cavaliers)                                                                                                                                                                                                                                   |
| 30h~38h | Modem Miniware M4000 (Ports paramétrables soit sur la plage 20h~28h, soit sur 30h~38h à l'aide de cavaliers)                                                                                                                                                                                                                                    |
| 30h~38h | Interface SCSI GREEN/MAK                                                                                                                                                                                                                                                                                                                        |
| 30h~38h | Interface CD-ROM Philips NMS 0210                                                                                                                                                                                                                                                                                                               |
| 37h~3Fh | MSX interface Modem Philips NMS 1250/1255                                                                                                                                                                                                                                                                                                       |
| 37h~3Fh | Serial interface Philips NMS 1210/1211/1212 (Ports paramétrables soit sur la plage 27h~2Fh, soit sur 37h~3Fh à l'aide de cavaliers)                                                                                                                                                                                                             |
| 3Ch     | Accès au registre de contrôle du Musical Memory Mapper.                                                                                                                                                                                                                                                                                         |
| 3Fh     | Accès au registre de contrôle du SN76489 du Musical Memory Mapper.                                                                                                                                                                                                                                                                              |
|         |                                                                                                                                                                                                                                                                                                                                                 |
| 40h~4Fh | <p>Accès aux ports E/S étendus.</p> <p>L'écriture d'un identifiant au registre 40h permet d'accéder au matériel correspondant via les ports 41h~4Fh. Ceci permet d'étendre le nombre de matériel accessible via les ports d'E/S. Voici la liste des identifiants réservés :</p>                                                                 |

|         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|         | <div> <div> 1 = ASCII / Microsoft<br/>2 = Canon<br/>3 = Casio<br/>4 = Fujitsu<br/>5 = General<br/>6 = Hitachi<br/>7 = Kyocera<br/>8 = Panasonic (MSX2+)<br/>9 = Mitsubishi<br/>10 = NEC<br/>11 = Yamaha<br/>12 = JVC<br/>13 = Philips<br/>14 = Pioneer<br/>15 = Sanyo<br/>16 = Sharp </div> <div> 17 = Sony<br/>18 = Spectravideo<br/>19 = Toshiba<br/>20 = Mitsumi<br/>21 = Telematica<br/>22 = Gradiente<br/>23 = Sharp Brazil<br/>24 = GoldStar<br/>25 = Daewoo<br/>26 = Samsung<br/>128 = Panasonic (Image Scanner)<br/>165 = WOPR3<br/>170 = SuperSoniqs (Darky)<br/>171 = SuperSoniqs (Darky second setting)<br/>212 = 1chipMSX / Zemmex Neo (KdL firmware)<br/>254 = ASCII (MPS2) </div> </div> <p>L'écriture de 0 (ou une valeur non utilisée) au port 40h désactive les ports E/S étendus.</p> <p><u>Exemple d'utilisation avec les MSX2+ Panasonic :</u></p> <ul style="list-style-type: none"> <li>- Ecrire la valeur 8 au port 40h pour activer les ports d'E/S étendus des MSX2+ Panasonic.</li> <li>- Lire le port 040h. Si la valeur lue est 247 (8 en bits inversés), c'est donc qu'il s'agit d'un MSX2+ Panasonic ayant un Z80 avec mode turbo (5.37MHz).</li> <li>- Vous pouvez donc d'écrire 0 au port 41h pour activer le mode turbo ou 1 pour le désactiver. La lecture du bit 0 de ce port, vous indiquera le mode actuel et le bit 7, l'état de interrupteur du logiciel interne (0 pour Off).</li> </ul> |
| 48h~49h | Cartouche Franky. (SN76489 et VDP)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 50h~6Fh | Ports réservés pour le système.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 5Eh~5Fh | Interface Ethernet GR8NET.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 5Fh     | SD-512 interface. (Accessible en écriture seulement)<br>Bits 0~3 = Numéro de page de 16Ko répété sur les 4 pages. Ignorés lorsque le bit 7 = 0.<br>Bit 7 = Activation du l'écriture dans la mémoire Flash.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 60h~6Fh | Graphics9000 / V9990.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 70h~73h | MIDI Saurus. (Bit <sup>2</sup> )                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 74h~7Bh | Ports réservés pour le système.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 78h~79h | Cartouche 80 colonnes SVI-727 Spectravideo.<br>78h est le port de sélection de registre et 79h le port d'envoi ou de lecture de donnée.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 7Ch~7Dh | MSX-Music (OPLL).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 7Eh~7Fh | MoonSound (OPL4).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 80h~83h | Interface RS-232C. (option)<br>Ports 80h~83h : Émetteur-récepteur asynchrone universel (8251).<br>Ports 84h~87h : Timer programmable (8253).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 88h~8Bh | Accès aux registres de contrôle, de statut, de la palette de couleur et de transfert de données d'un V9938 externe. (ex : Extension MSX2 ou 80 colonnes pour MSX1.)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 8Ch~8Dh | MSX Modem.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 8Eh~8Fh | Accès aux registres de contrôle de la MegaRam / MegaRam Disk.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

| 8Eh~8Fh  | <i>Ports réservés pour le système.</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |      |      |      |      |      |      |      |  |      |      |      |      |      |      |      |      |          |   |   |   |   |   |   |    |  |          |      |   |   |      |     |      |      |      |  |      |      |      |      |      |      |      |      |          |     |     |     |     |     |     |     |     |          |   |   |   |      |     |      |      |      |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------|------|------|------|------|------|--|------|------|------|------|------|------|------|------|----------|---|---|---|---|---|---|----|--|----------|------|---|---|------|-----|------|------|------|--|------|------|------|------|------|------|------|------|----------|-----|-----|-----|-----|-----|-----|-----|-----|----------|---|---|---|------|-----|------|------|------|
| 90h~91h  | Accès aux registres de contrôle de l'imprimante.<br>La lecture du bit 1 du port 90h indique si l'imprimante est occupée (1) ou non (0).<br>Une écriture au port 90h avec le bit 0 à 0 puis une autre avec le bit 0 à 1 permet d'envoyer la donnée actuelle du port 91h.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |      |      |      |      |      |      |      |  |      |      |      |      |      |      |      |      |          |   |   |   |   |   |   |    |  |          |      |   |   |      |     |      |      |      |  |      |      |      |      |      |      |      |      |          |     |     |     |     |     |     |     |     |          |   |   |   |      |     |      |      |      |
| 93h      | Accès aux registres de contrôle de la direction du bus de l'imprimante (port 91h).<br>Seuls les bits 1 et 0 sont utilisés. (00 pour état Z, 01/11 pour sortie, 10 pour entrée)<br>Note : Ce port est non standardisée et très peu répandus. Le S1985 peut le gérer.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |      |      |      |      |      |      |      |  |      |      |      |      |      |      |      |      |          |   |   |   |   |   |   |    |  |          |      |   |   |      |     |      |      |      |  |      |      |      |      |      |      |      |      |          |     |     |     |     |     |     |     |     |          |   |   |   |      |     |      |      |      |
| 92h~97h  | <i>Ports réservés pour le système.</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |      |      |      |      |      |      |      |  |      |      |      |      |      |      |      |      |          |   |   |   |   |   |   |    |  |          |      |   |   |      |     |      |      |      |  |      |      |      |      |      |      |      |      |          |     |     |     |     |     |     |     |     |          |   |   |   |      |     |      |      |      |
| 98h~99h  | Accès aux registres de contrôle et au(x) registre(s) de statut du VDP. (Voir le chapitre 7.4 « <a href="#">Comment accéder au VDP</a> » page 264)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |      |      |      |      |      |      |      |  |      |      |      |      |      |      |      |      |          |   |   |   |   |   |   |    |  |          |      |   |   |      |     |      |      |      |  |      |      |      |      |      |      |      |      |          |     |     |     |     |     |     |     |     |          |   |   |   |      |     |      |      |      |
| 9Ah~9Bh  | Accès aux registres de la palette de couleur et de transfert de données du VDP (v9938 / v9958 seulement) (Voir le chapitre 7.4 « <a href="#">Comment accéder au VDP</a> » page 264)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |      |      |      |      |      |      |      |  |      |      |      |      |      |      |      |      |          |   |   |   |   |   |   |    |  |          |      |   |   |      |     |      |      |      |  |      |      |      |      |      |      |      |      |          |     |     |     |     |     |     |     |     |          |   |   |   |      |     |      |      |      |
| 9Ch~9Fh  | <i>Ports réservés pour le système.</i> Les ports du VDP se reflètent sur ceux-là sur certains MSX. (Par exemple, les MSX faits avec le S3527 et le S1985.)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |      |      |      |      |      |      |      |  |      |      |      |      |      |      |      |      |          |   |   |   |   |   |   |    |  |          |      |   |   |      |     |      |      |      |  |      |      |      |      |      |      |      |      |          |     |     |     |     |     |     |     |     |          |   |   |   |      |     |      |      |      |
| A0h~A2h  | Accès aux registres du PSG des MSX. (AY-3-8910 / YM2149)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |      |      |      |      |      |      |      |  |      |      |      |      |      |      |      |      |          |   |   |   |   |   |   |    |  |          |      |   |   |      |     |      |      |      |  |      |      |      |      |      |      |      |      |          |     |     |     |     |     |     |     |     |          |   |   |   |      |     |      |      |      |
| A4h~A7h  | Les ports du PSG se reflètent sur ceux-là sur certains MSX. (Par exemple, les MSX faits avec le S3527 et le S1985.)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |      |      |      |      |      |      |      |  |      |      |      |      |      |      |      |      |          |   |   |   |   |   |   |    |  |          |      |   |   |      |     |      |      |      |  |      |      |      |      |      |      |      |      |          |     |     |     |     |     |     |     |     |          |   |   |   |      |     |      |      |      |
| A4h~A5h  | <p>Accès aux registres du convertisseur Analogique/Numérique du MSX turbo R.</p> <p>En lecture :</p> <table border="1"> <thead> <tr> <th></th><th>bit7</th><th>bit6</th><th>bit5</th><th>bit4</th><th>bit3</th><th>bit2</th><th>bit1</th><th>bit0</th></tr> </thead> <tbody> <tr> <td>Port A4h</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="2">CT</td></tr> <tr> <td>Port A5h</td><td>COMP</td><td>0</td><td>0</td><td>SMPL</td><td>SEL</td><td>FILT</td><td>MUTE</td><td>BUFF</td></tr> </tbody> </table> <p>CT = CT est incrémenté toutes les 63,5 µs.<br/> Si ADDA est à 1, la donnée écrite au port A4h est envoyée à répétition à chaque incrémentation du compteur. L'écriture au port A4h initialise le compteur.<br/> Si ADDA est à 0, la donnée écrite au port A4h est envoyée directement. Les écritures au port A4h n'initialise pas le compteur.</p> <p>BUFF = ADDA (voir plus bas)<br/> COMP = 0 lorsque la sortie du comparateur A/N est supérieur à la donnée du port A4h ;<br/> 1 lorsque la sortie du comparateur A/N est inférieur à la donnée du port A4h.</p> <p>En écriture :</p> <table border="1"> <thead> <tr> <th></th><th>bit7</th><th>bit6</th><th>bit5</th><th>bit4</th><th>bit3</th><th>bit2</th><th>bit1</th><th>bit0</th></tr> </thead> <tbody> <tr> <td>Port A4h</td><td>DA7</td><td>DA6</td><td>DA5</td><td>DA4</td><td>DA3</td><td>DA2</td><td>DA1</td><td>DA0</td></tr> <tr> <td>Port A5h</td><td>0</td><td>0</td><td>0</td><td>SMPL</td><td>SEL</td><td>FILT</td><td>MUTE</td><td>ADDA</td></tr> </tbody> </table> <p>DA0~DA7 = Donnée à envoyer au comparateur A/N.<br/> ADDA = 0 pour cache unique* (conversion A/N) ;<br/> 1 pour double cache (conversion A/N).<br/> MUTE = 1 pour couper le son* ; 0 pour activer le son.<br/> FILT = 1 pour activer le filtre ; 0 pour signal normal*.<br/> SEL = Sélection du signal à filtrer. 0 pour comparateur A/N* ; 1 pour l'ampli du micro.<br/> SMPL = Signal d'échantillonnage. 0 pour numériser* ; 1 pour attendre.<br/> (*) valeur par défaut.</p> <p>Exemples d'utilisation :</p> |      |      |      |      |      |      |      |  | bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 | Port A4h | 0 | 0 | 0 | 0 | 0 | 0 | CT |  | Port A5h | COMP | 0 | 0 | SMPL | SEL | FILT | MUTE | BUFF |  | bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 | Port A4h | DA7 | DA6 | DA5 | DA4 | DA3 | DA2 | DA1 | DA0 | Port A5h | 0 | 0 | 0 | SMPL | SEL | FILT | MUTE | ADDA |
|          | bit7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |  |      |      |      |      |      |      |      |      |          |   |   |   |   |   |   |    |  |          |      |   |   |      |     |      |      |      |  |      |      |      |      |      |      |      |      |          |     |     |     |     |     |     |     |     |          |   |   |   |      |     |      |      |      |
| Port A4h | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 0    | 0    | 0    | 0    | 0    | CT   |      |  |      |      |      |      |      |      |      |      |          |   |   |   |   |   |   |    |  |          |      |   |   |      |     |      |      |      |  |      |      |      |      |      |      |      |      |          |     |     |     |     |     |     |     |     |          |   |   |   |      |     |      |      |      |
| Port A5h | COMP                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 0    | 0    | SMPL | SEL  | FILT | MUTE | BUFF |  |      |      |      |      |      |      |      |      |          |   |   |   |   |   |   |    |  |          |      |   |   |      |     |      |      |      |  |      |      |      |      |      |      |      |      |          |     |     |     |     |     |     |     |     |          |   |   |   |      |     |      |      |      |
|          | bit7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |  |      |      |      |      |      |      |      |      |          |   |   |   |   |   |   |    |  |          |      |   |   |      |     |      |      |      |  |      |      |      |      |      |      |      |      |          |     |     |     |     |     |     |     |     |          |   |   |   |      |     |      |      |      |
| Port A4h | DA7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | DA6  | DA5  | DA4  | DA3  | DA2  | DA1  | DA0  |  |      |      |      |      |      |      |      |      |          |   |   |   |   |   |   |    |  |          |      |   |   |      |     |      |      |      |  |      |      |      |      |      |      |      |      |          |     |     |     |     |     |     |     |     |          |   |   |   |      |     |      |      |      |
| Port A5h | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 0    | 0    | SMPL | SEL  | FILT | MUTE | ADDA |  |      |      |      |      |      |      |      |      |          |   |   |   |   |   |   |    |  |          |      |   |   |      |     |      |      |      |  |      |      |      |      |      |      |      |      |          |     |     |     |     |     |     |     |     |          |   |   |   |      |     |      |      |      |

```

; Routine PCM Play
;
; Entrée: HL = Adresse des données à lire
;         BC = Longueur des données
;         E = Fréquence de l'échantillonnage
;           1 (15,75KHz), 2 (7,875KHz) ou 3 (5,25KHz)

PCMdac      equ    0A4h ; Convertisseur A/N (sortie)
PCMcnt      equ    0A4h ; Compteur (entrée)
PCMcntl     equ    0A5h ; Contrôle du PCM (sortie)

PCMplay:
    ld        a,00000011b
    out       (PCMcntl),a ; Mode conversion A/N
    di                          ; Pour garder de bons timings
PCMplayLoop:
    in        a,(PCMcnt)
    sub       e
    jr        c,PCMplayLoop
    ld        a,(hl)
    out       (PCMdac),a
    inc       hl
    dec       bc
    ld        a,c
    or        b
    jr        nz,PCMplayLoop ; Saute si le compteur = 0
    ei
    ret

; Routine PCM Rec (assembleur M80)
;
; Entrée: HL = Adresse des données à lire
;         BC = Longueur des données
;         E = Fréquence de l'échantillonnage
;           1 (15,75KHz), 2 (7,875KHz) ou 3 (5,25KHz)

PCMdac      equ    0A4h ; Convertisseur A/N
PCMcnt      equ    0A4h ; Compteur
PCMcntl     equ    0A5h ; Contrôle du PCM
PCMstat     equ    0A5h ; Statut du PCM

adconv      macro next_bit,strip ; Macro de conversion A/N 1 bit
    local adconv_not_change
    out       (PCMdac),a ; Sortie de la donnée
    db        0EDh,70h ; Instruction du Z80 non-documentée IN (C)
                    ; qui change les indicateurs du registre F
                    ; en fonction de ce qui est lu au port C.
    jp        m,adconv_not_change
    and       strip ; Mets le bit à 0 car plus petit que le
                    ; signal d'entrée analogique
adconv_not_change:
    or        next_bit ; Bit suivant à 1
    endm      ; Fin de création de la macro

PCMrec:
    ld        a,00001100b
    out       (PCMcntl),a ; Mode conversion A/N
    ld        d,0 ; Pour mettre le compteur à 0
    di                          ; Pour garder de bons timings
PCMrecLoop:
    in        a,(PCMcnt) ; Lecture du compteur
    cp        d
    jr        nz,PCMrecLoop ; Saut si la valeur est adéquate

```

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                 |                 |                 |                 |                 |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|
|          | <pre>add    a,e                ; Création de la valeur du and    11b                ; compteur suivant ld     d,a ; Plus rapide de 7µs qui ci-dessus exx ld     a,00011100b out    (PCMcntl),a ; Maintient la donnée ld     c,PCMstat ld     a,80h            ; Bit COMP à 1 adconv 01000000b,01111111b ; récupération de chaque bit adconv 00100000b,10111111b adconv 00010000b,11011111b adconv 00001000b,11101111b adconv 00000100b,11110111b adconv 00000010b,11111011b adconv 00000001b,11111101b adconv 00000000b,11111110b exx ld     (hl),a            ; Stocke la donnée (l'octet) ld     a,00001100b out    (PCMcntl),a ; Supprime la donnée maintenue inc    hl                ; Prochaine donnée dec    bc ld     a,c or     b jr     nz,PCMrecLoop     ; Saute si le compteur = 0 ld     a,00000011b out    (PCMcntl),a ; Mode conversion A/N et active le son ei ret</pre>                                                                                                                                                                                                                                                                                                                                                                                                                          |                 |                 |                 |                 |                 |
| A6h      | Ports réservés pour le système.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                 |                 |                 |                 |                 |
| A7h      | Accès au registre de contrôle des LED Pause et mode R800 du MSX turbo R.<br>bit 0 = 1 pour allumer la LED Pause ;<br>bit 1~6 = ignorés.<br>bit 7 = 1 pour allumer la LED Mode R800.<br>Ces deux bits sont sauvegardés à l'adresse 0FCB1h par le système du turbo R.<br>En lecture, le bit 0 indique l'état de la touche « Pause ».                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                 |                 |                 |                 |                 |
| A8h~ABh  | Accès au registre de l'interface de port programmable (PPI : 8255 / ULA9RA041).<br><br>Port A8h = Accès au registre de sélection des Slot primaires (via le port A du PPI).<br><div><div>bit7   bit6   bit5   bit4   bit3   bit2   bit1   bit0</div><table><tr><td>Port A8h</td><td>Slot primaire 3</td><td>Slot primaire 2</td><td>Slot primaire 1</td><td>Slot primaire 0</td></tr></table></div><br>Port A9h = Lecture d'une ligne de la matrice du clavier (via le port B du PPI).<br>Port AAh = Accès au registre de contrôle du clavier et du lecteur à cassette (via le port C du PPI).<br>bits 0~3 = Numéro de la matrice du clavier à lire via le port B ;<br>bit 4 = 0 pour faire tourner le moteur du lecteur cassette ;<br>bit 5 = 1 commander l'écriture sur cassette ;<br>bit 6 = 0 pour allumer la LED CAPS du clavier ;<br>bit 7 = 1, puis 0 peu après pour émettre un cliquetis (pour les touches).<br>Port ABh = Accès au registre de contrôle des ports du PPI.<br>La fonction des bits 0~6 dépendent de l'état du bit 7. Lorsque celui-ci est à 0, ils ont la fonction suivante :<br>bit 0 = Etat du bit à changer ;<br>bits 1~3 = Numéro du bit à changer au port C du PPI.<br>bits 4~6 = Inutilisés.<br>Lorsque le bit 7 est à 1, ils ont en théorie la fonction suivante mais en fait, ils | Port A8h        | Slot primaire 3 | Slot primaire 2 | Slot primaire 1 | Slot primaire 0 |
| Port A8h | Slot primaire 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Slot primaire 2 | Slot primaire 1 | Slot primaire 0 |                 |                 |

|         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|         | <p>ne doivent/peuvent pas être modifiés :</p> <p>bit 0 = Direction des bits de poids faible du port C. 0 pour sortie (MSX) ;</p> <p>bit 1 = Direction du port B. 0 pour sortie, 1 pour entrée (MSX) ;</p> <p>bit 2 = Mode des ports B et des bits de poids faible du port C.<br/>0 pour mode normal (MSX), 1 pour mode bidirectionnel ;</p> <p>bit 3 = Direction des bits de poids fort du port C. 0 pour sortie (MSX)</p> <p>bit 4 = Direction du port A. 0 pour sortie (MSX) ;</p> <p>bits 5~6 = Mode des ports A et des bits de poids fort du port C.<br/>00 pour mode normal (MSX), 01 pour mode à impulsion et 10 pour mode bidirectionnel ;</p>                                               |
| ACh~AFh | Les ports du PPI se reflètent sur ceux-là sur certains MSX. (Par exemple, les MSX faits avec le S3527 et le S1985.)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| ACh~AFh | Ports E/S du 1chipMSX.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| B0h~B3h | Accès aux registres de la cartouche Sony HBI-55 et Yamaha UDC-01. Ces cartouches contiennent 4Ko de SRAM. (External PPI 8255)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| B4h~B5h | <p>Accès aux registres de l'horloge en temps réel. (RP-5C01) (Voir les routines REDCLK et WRTCLK de la Sub-ROM.)</p> <p>Port B4h = Numéro de registre sur 4 bits (écriture seule) ;</p> <p>Port B5h = Donnée sur 4 bits à lire ou à écrire.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| B6h~B7h | <i>Ports réservés pour le système.</i> Les ports de la RTC se reflètent sur ceux-là sur certains MSX. (Par exemple, les MSX faits avec le S1985.)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| B8h~BBh | Interface pour crayon optique Sanyo.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| BCh~BFh | Contrôleur VHD.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| C0h~C1h | <p>MSX Audio (canal maitre) / Music Module.</p> <p>Port C0h = Numéro de registre (écriture seule) ;</p> <p>Port C1h = Donnée du canal maitre (écriture et lecture partielle).</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| C2h~C3h | <p>MSX Audio (canal esclave) / Music Module (ports pour une deuxième cartouche)</p> <p>Port C2h = Numéro de registre (écriture seule) ;</p> <p>Port C3h = Donnée du canal maitre (écriture et lecture partielle).</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| C0h~C3h | Moonsound / OPL4 (ports alternatifs).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| C4h~C7h | Moonsound / OPL4.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| C8h~CCh | Accès aux registres de la « MSX Interface ». (Interface Série Asynchrone)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| D0h~D7h | Contrôleur de lecteur de disquette. (WD2793 Microsol)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| D8h~D9h | <p>Kanji Rom niveau 1 ou Hangul + Kanji Rom coréenne.</p> <p>Kanji Rom :</p> <p>D8h est le port pour écrire le code « Ku » du caractère auquel l'on veut lire les données.</p> <p>D9h est le port pour écrire le code « Ten » du caractère auquel l'on veut lire les données. La lecture des données se fait au port D9h. Il y a 32 octets par caractère dont 16 (2 x 8) pour le haut du caractère et 16 pour le bas.</p> <p>Hangul + Kanji Rom :</p> <p>Le code du caractère auquel l'on veut lire les données fait 16 bits. Il doit être écrit au port D8h (bits de poids faible) et D9h (bits de poids fort).</p> <p>Le bit 15 indique qu'il s'agit un caractère Hangul (1) ou un Kanji (0).</p> |



|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |       |       |       |       |       |  |          |       |       |                                                |       |                                                                                                                                                                                                                        |       |                                       |     |                                        |    |                                         |    |    |    |    |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|-------|-------|-------|-------|-------|--|----------|-------|-------|------------------------------------------------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------------------------------------|-----|----------------------------------------|----|-----------------------------------------|----|----|----|----|
|          | Un caractère Hangul est codé en 3 parties :<br><br>bits 10~14 pour la première partie<br>bits 5~9 pour la seconde partie<br>bits 0~4 pour la troisième partie                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |       |       |       |       |       |       |  |          |       |       |                                                |       |                                                                                                                                                                                                                        |       |                                       |     |                                        |    |                                         |    |    |    |    |
| DAh~DBh  | Kanji Rom niveau 2.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |       |       |       |       |       |       |  |          |       |       |                                                |       |                                                                                                                                                                                                                        |       |                                       |     |                                        |    |                                         |    |    |    |    |
| DCh      | Kanji Rom étendue.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |       |       |       |       |       |       |  |          |       |       |                                                |       |                                                                                                                                                                                                                        |       |                                       |     |                                        |    |                                         |    |    |    |    |
| DCh~DDh  | Support des manettes pour les jeux Sega avec la cartouche PlaySoniq.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |       |       |       |       |       |  |          |       |       |                                                |       |                                                                                                                                                                                                                        |       |                                       |     |                                        |    |                                         |    |    |    |    |
| E4h~E5h  | Accès aux registres de réglage du S1990. (MSX turbo R)<br>E4h = Register number set<br>E5h = Register data read/write <table><tr><td>Register</td><td></td></tr><tr><td>5</td><td>Bit 6 = 1 pour logiciel interne (lecture seul)</td></tr><tr><td>6</td><td>Bit 7 = 1 pour Hard reset, 0 pour Soft reset (écriture seul)<br/>Bit 6 = 1 pour mode ROM, 0 pour DRAM (lecture/écriture)<br/>Bit 5 = 1 pour Z80, 0 pour R800 (lecture/écriture)<br/>Bit 4~0 = Restent à 0. (lecture seul)</td></tr><tr><td>13</td><td>Contient 3 normalement (lecture seul)</td></tr><tr><td>14</td><td>Contient 47 normalement (lecture seul)</td></tr><tr><td>15</td><td>Contient 139 normalement (lecture seul)</td></tr></table> |       |       |       |       |       |       |  | Register |       | 5     | Bit 6 = 1 pour logiciel interne (lecture seul) | 6     | Bit 7 = 1 pour Hard reset, 0 pour Soft reset (écriture seul)<br>Bit 6 = 1 pour mode ROM, 0 pour DRAM (lecture/écriture)<br>Bit 5 = 1 pour Z80, 0 pour R800 (lecture/écriture)<br>Bit 4~0 = Restent à 0. (lecture seul) | 13    | Contient 3 normalement (lecture seul) | 14  | Contient 47 normalement (lecture seul) | 15 | Contient 139 normalement (lecture seul) |    |    |    |    |
| Register |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |       |       |       |       |       |       |  |          |       |       |                                                |       |                                                                                                                                                                                                                        |       |                                       |     |                                        |    |                                         |    |    |    |    |
| 5        | Bit 6 = 1 pour logiciel interne (lecture seul)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |       |       |       |       |       |  |          |       |       |                                                |       |                                                                                                                                                                                                                        |       |                                       |     |                                        |    |                                         |    |    |    |    |
| 6        | Bit 7 = 1 pour Hard reset, 0 pour Soft reset (écriture seul)<br>Bit 6 = 1 pour mode ROM, 0 pour DRAM (lecture/écriture)<br>Bit 5 = 1 pour Z80, 0 pour R800 (lecture/écriture)<br>Bit 4~0 = Restent à 0. (lecture seul)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |       |       |       |       |       |  |          |       |       |                                                |       |                                                                                                                                                                                                                        |       |                                       |     |                                        |    |                                         |    |    |    |    |
| 13       | Contient 3 normalement (lecture seul)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |       |       |       |       |       |       |  |          |       |       |                                                |       |                                                                                                                                                                                                                        |       |                                       |     |                                        |    |                                         |    |    |    |    |
| 14       | Contient 47 normalement (lecture seul)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |       |       |       |       |       |  |          |       |       |                                                |       |                                                                                                                                                                                                                        |       |                                       |     |                                        |    |                                         |    |    |    |    |
| 15       | Contient 139 normalement (lecture seul)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |       |       |       |       |       |       |  |          |       |       |                                                |       |                                                                                                                                                                                                                        |       |                                       |     |                                        |    |                                         |    |    |    |    |
| E6h~E7h  | Accès au registre du Timer du S1990. Ce Timer est incrémenté toutes les 3,911 µs.<br>E6h = 8 bits de poids faible de la valeur du Timer. (lecture)<br>Une écriture à ce port remet le Timer à zéro.<br>E7h = 8 bits de poids fort de la valeur du Timer. (lecture)<br><br>Exemple d'utilisation :<br><br>; Entrée: B = temps d'attente (B x 3,911 µs).<br>; Sortie: Les registres A, C et F seront modifiés<br>; Notes: B doit être supérieur à 0.<br>; Les interruptions doivent être désactivées.<br><br>TIMERLSB equ 0E6h<br>TIMERMSB equ 0E7h<br><br>Wait:<br>in a, (TIMERLSB)<br>ld c, a<br>WaitLoop:<br>in a, (TIMERMSB)<br>sub c<br>cp b<br>jr c, WaitLoop ; Saute si le temps n'est pas écoulé<br>ret    |       |       |       |       |       |       |  |          |       |       |                                                |       |                                                                                                                                                                                                                        |       |                                       |     |                                        |    |                                         |    |    |    |    |
| F3h      | Accès au registre servant à connaître le mode d'affichage actuel. (MSX2+ seulement)<br>La lecture de ce registre permet de connaître l'état des bits suivants du VDP. <table><tr><td>bit 7</td><td>bit 6</td><td>bit 5</td><td>bit 4</td><td>bit 3</td><td>bit 2</td><td>bit 1</td><td>bit 0</td></tr><tr><td>YAE</td><td>YUV</td><td>TP</td><td>M1</td><td>M2</td><td>M5</td><td>M4</td><td>M3</td></tr></table> M1~5 = indiquent le mode d'écran actuel<br>TP = 1 lorsque la couleur 0 est transparente, 0 si égale à la couleur de la palette<br>YUV = 0 pour mode RGB, 1 pour YJK                                                                                                                            |       |       |       |       |       |       |  | bit 7    | bit 6 | bit 5 | bit 4                                          | bit 3 | bit 2                                                                                                                                                                                                                  | bit 1 | bit 0                                 | YAE | YUV                                    | TP | M1                                      | M2 | M5 | M4 | M3 |
| bit 7    | bit 6                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |  |          |       |       |                                                |       |                                                                                                                                                                                                                        |       |                                       |     |                                        |    |                                         |    |    |    |    |
| YAE      | YUV                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | TP    | M1    | M2    | M5    | M4    | M3    |  |          |       |       |                                                |       |                                                                                                                                                                                                                        |       |                                       |     |                                        |    |                                         |    |    |    |    |

|         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|         | YAE = 1 lorsque les bits d'attribut sont activés                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| F4h     | Accès au registre « Reset status » qui sert d'indicateur pour que le système du MSX2+ ou plus récent s'initialise de façon adaptée. Vous ne devez pas modifier ce registre.<br>bit 5 = 1 pour que le R800 ne soit pas initialisé ;<br>bit 7 = 1 pour une initialisation logicielle.<br>Note : Les bits sont inversés sur MSX2+.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| F5h     | Accès au registre de contrôle des ports du système. (MSX2~)<br>Ce registre n'est accessible qu'en écriture seule. Il sert à désactiver (bit à 0) certains matériels intégrés au MSX. Notez que le Bios désactive certains matériels pendant l'initialisation du MSX.<br>bit 0 = Kanji Rom niveau 1 ;<br>bit 1 = Kanji Rom niveau 2 (MSX2+ seulement) ;<br>bit 2 = MSX-Audio. Si la valeur lu au port 0C0H est 255, c'est qu'il n'y en a pas en interne. ;<br>bit 3 = Super impose. Effectuez un ET logique de la valeur des bits 3 lue aux ports 77h et F7h, savoir si le MSX possède la super impose en interne. ;<br>bit 4 = MSX Interface. Si la valeur lu au port 0C0H est 255, c'est qu'il n'y en a pas en interne. MSX Interface est un LSI de communication à usage général qui n'est pas encore utilisé actuellement. ;<br>bit 5 = RS-232C aux ports E/S mappé ; Si le registre de statut du i8351 vaut xx000101b après une initialisation interne, c'est qu'il y a une RS-232C en interne. ;<br>bit 6 = Crayon optique. Si la valeur lu au port 0BBH est 255, c'est qu'il n'y en a pas en interne. ;<br>bit 7 = RTC. Ce bit est mis à 0 pendant l'initialisation et reste ainsi lorsqu'une cartouche contient une RTC. Vous pouvez ignorer ce bit dans vos programmes. |
| F6h     | Bus du port entrée/sortie des couleurs.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| F7h     | Accès au registre de contrôle A/V des MSX NTSC. Ce registre n'est accessible qu'en écriture sauf le bit 3 qui n'est accessible qu'en lecture.<br>bit 0 = Mettre à 1 pour mixer des sorties audio droite et gauche ;<br>bit 1 = Sortie audio gauche ;<br>bit 2 = Mettre à 1 pour régler l'entrée vidéo sur la prise RGB21 ;<br>bit 3 = Indicateur pour savoir si l'entrée vidéo est désactiver ou non ;<br>bit 4 = Contrôle AV, 1 pour TV ;<br>bit 5 = Contrôle Ym, 1 pour TV ;<br>bit 6 = Inverse l'état du bit 4 du registre 9 du VDP ;<br>bit 7 = Inverse l'état du bit 5 du registre 9 du VDP.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| F8h     | Accès au registre de contrôle A/V des MSX PAL.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| F8h~FBh | <i>Ports réservés pour le système.</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| F8h~FBh | Accès aux 8 bits de poids fort des registres 16 bits du Memory Mapper de la cartouche PlaySoniq. (Désactivé par défaut)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| FCh~FFh | Accès aux registres du <a href="#">Memory Mapper</a> (paragraphe 3.5 page 125).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

# Lexique

## - *Bruit blanc*

Le bruit blanc du PSG est un son produit par des fréquences sonores pseudo-aléatoires.

## - *Chaine ASCIIZ*

Suite de caractères codé ASCII dont le dernier code est zéro.

## - *DTA (Disk Transfer Area)*

La DTA est une adresse qui pointe un cache en mémoire utilisé pour transférer des données des disques sous MSX-DOS.

## - *Main-RAM*

Main-RAM désigne les 64Ko de RAM dans les Slots que le système a sélectionné au démarrage.

## - *Mapper*

Mapper désigne un dispositif qui permet d'étendre la mémoire grâce à un système de changement de page. « Memory Mapper » désigne le système de changement de page de la RAM au standard MSX2. « Rom Mapper » désigne le système de changement de page utilisé dans les Megarom par exemple. Il en existe plusieurs et aucun sont standardisés.

## - *Megarom*

Megarom désigne les cartouches ayant une Rom d'au moins 1 méga-bits (128Ko~). Celles-ci utilisent un Mapper destiné étendre la mémoire à plus de 64Ko. Il peut exister des cartouches de 64Ko, voir moins, munies d'un Mapper.

## - *MSX-Engine*

« MSX-Engine » est le nom donné à une puce, fabriqué par Toshiba, qui intègre différents composants couramment utilisés dans les MSX. Les plus récentes intègrent même le PSG et le CPU. ASCII a fabriqué aussi le même genre de puce. Bien que le nom de ces dernières soit « MSX-System », en Europe, on dit souvent « MSX-Engine ».

## - *Pattern*

« Pattern » est un terme anglais utilisé en graphisme ou musique. Cela signifie « patron », c'est à dire une portion (un motif ou un refrain) qui peut être reproduite là où l'on veut selon les paramètres indiqués dans une table ou une chaine MML.

## - *PPI (Programmable Port Interface)*

Sur MSX, le PPI est une puce utilisée pour commuter les Slot, pour gérer le clavier (cliquetis inclus) et le lecteur cassette.

## - *PSG (Programmable Sound Generator)*

PSG signifie : Générateur de Son Programmable. Programmable Sound Generator est une appellation de General Instrument. Yamaha nomme ce genre de puce SSG (Software-controlled Sound Generator) et Texas Instrument DCSG (Digital Complex Sound Generator). Bref, tout ceci désigne un puce dédiée au effet sonore et à la musique. Le PSG des MSX gère aussi les manettes de jeu, la souris, les molettes, la tablette graphique, etc.

- ***RAM (Random Access Memory)***

La RAM est la mémoire vive de l'ordinateur. Elle sert à y mettre les programmes et les données le temps que l'ordinateur est sous tension.

- ***ROM (Read Only Memory)***

Une ROM est une « mémoire morte ». Elle sert à y mettre des programmes ou des données permanentes que l'ordinateur pourra exécuter ou lire même après une extinction ou un redémarrage.

- ***RTC (Real Time Clock)***

Horloge en temps réel. Dans les ordinateurs MSX2 et plus récents, il y a une puce qui gère la date et l'heure en temps réel et ce même lorsque la machine est éteinte. Cette puce est alimentée par pile(s).

- ***Sprite***

Sprite est un terme anglais de l'époque que l'on traduit par lutin. Aujourd'hui, on appellerait ça plutôt un calque. Dans le cas du MSX, c'est une sorte petit calque d'une taille de 8 x 8 ou 16 x 16 pixels.

- ***Slot***

Slot, que l'on traduit par fente, est un port cartouche physique ou non. Les ports cartouches d'un MSX sont des Slot primaires. Ce n'est pas forcément le cas pour les bus d'extension.

- ***Vblank***

La Vblank est la partie non visible de la VRAM qui est scrutée par le VDP après chaque affichage de l'avant-plan. Le VDP envoie un signal d'interruption au Z80 à chaque début de scrutation de la Vblank.

- ***VRAM (Video Random Access Memory)***

La VRAM est la mémoire vive du processeur vidéo. Celle-ci contient toutes les données graphiques.

- ***VDP (Video Display processor)***

VDP veut dire : processeur de l'affichage vidéo. C'est un processeur dédiée à l'affichage à l'écran.