GUIDE D'UTILISATION DU LOGICIEL

MCXDOS

POUR LES CARTES DE LA GAMME MCX

GUIDE D'UTILISATION DU LOGICIEL MCXDOS POUR LES CARTES MCX ET MCX-*Lite*

MARQUES DEPOSEES

- *ACKSYS* est une marque déposée de *ACKSYS*.
- IBM PC, AT sont des marques déposées de International Business Machines Corporation.
- MS-DOS est une marque déposée de Microsoft Inc.
- S-DOS est une marque déposée de 3S.

NOTICE

ACKSYS ne garantit en aucune façon le contenu du présent document et dégage son entière responsabilité quant à la rentabilité et la conformité du matériel aux besoins de l'utilisateur.

ACKSYS ne pourra en aucun cas être tenu pour responsable des erreurs éventuellement contenues dans ce document, ni des dommages quelle qu'en soit leur importance, du fait de la fourniture, du fonctionnement ou de l'utilisation du matériel.

ACKSYS se réserve le droit de réviser périodiquement ce document, ou d'en changer le contenu, sans aucune obligation pour ACKSYS d'en aviser qui que ce soit.

Société ACKSYS
3 & 5 rue du Stade
BP 80
78302 POISSY CEDEX

Téléphone : 33 (1) 39-11-62-81
Télécopie : 33 (1) 39-11-47-96

TABLE DES MATIERES

I. INTRODUCTION 1

I.1 PRESENTATION DE MCXDOS 1
I.2 CONTENU DU MANUEL 1

II. CHARGEMENT DU DOS DANS LA CARTE 3

II.1 ETAPES PRELIMINAIRES 3
II.1.1 Vérification de la configuration des cavaliers 3
II.1.2 Chargement du fichier INTERMCX.COM 4
II.2 CHARGEMENT DEPUIS UNE DISQUETTE 4
II.3 CHARGEMENT DEPUIS UN FICHIER 5
II.4 CHARGEMENT DEPUIS UNE PARTITION DU DISQUE C: 6
II.5 UTILISATION D'UN DISQUE VIRTUEL 8
II.6 OPTIONS DE LANCEMENT DU PROGRAMME MCXDOS.EXE 9
II.7 OPTIONS DE LANCEMENT DU PROGRAMME MCX2HOST.COM 10
II.8 RESET DE LA CARTE 10
II.9 DEVELOPPEMENT D'APPLICATIONS 11
II.10 LE PROGRAMME INTERMCX.COM 13
II.11 DESCRIPTION DES FONCTIONS DE INTERMCX - INT 8FH 14
II.12 LE PROGRAMME MCX2HOST.COM 19
II.13 LES FONCTIONS DE MCX2HOST - INT 8EH 19
II.14 EXEMPLE DE PROGRAMMATION DE LA CARTE MCX-XX OU MCX-*LITE*/S 22

III. LES UTILITAIRES 31

III.1 BOOTFILE.COM 31
III.2 REBOOT.COM 31
III.3 RESETMCX.COM 31
III.4 EMULMCC.COM 31
III.5 DOSSETUP.EXE 31
III.6 AUTOMCX.COM 32
III.6.1 Chargement d'applications développées sous MCXDOS 33
III.6.2 Utilisation simultanée de plusieurs cartes MCX 33
III.6.3 Accès au disques du PC 33
III.7 MCXSPY.COM 34

IV. EXTENSION BIOS PCMCIA 35

V. NOTES 37

VI. FICHE ERREUR 39

I. INTRODUCTION

I.1 Présentation de MCXDOS

MCXDOS est un logiciel composé principalement des programmes MCXDOS.EXE, INTERMCX.COM et MCX2HOST.COM, permettant de charger MS-DOS sur les cartes ACKSYS MCX et MCX-*Lite*.

Dans ce mode de fonctionnement, la carte se comporte comme un véritable P.C. utilisant les ressources disque, écran et clavier du P.C. hôte. Le chargement du DOS dans la carte s'effectue également depuis les ressources disque du P.C. hôte (disquette, fichier ou partition du disque dur).

La mise au point d'une application peut alors être faite en même temps sur votre machine et sur votre carte, le passage des ressources P.C. vers celle-ci étant activé par une combinaison de touches sur votre clavier. Il devient donc extrêmement facile de développer dans la carte car vous pouvez utiliser les mêmes compilateurs et les mêmes logiciels de mise au point que ceux que vous avez l'habitude d'utiliser sur votre P.C.

MCXDOS est livré avec une librairie de primitives qui donne accès aux périphériques MCX étrangers à la structure P.C. classique, notamment aux unités de communication série.

MCXDOS ne peut fonctionner que sur un système MS-DOS 5 ou supérieur, équipé d'un écran couleur VGA. C'est un logiciel résident qui occupe environ 45 Ko de mémoire système. Les programmes utilisant un mode d'affichage graphique ne sont pas supportés par la carte sous MCXDOS. De même, certains programmes qui adressent directement le port clavier du P.C. au lieu d'utiliser les fonctions standards du BIOS et du DOS, ne pourront pas fonctionner sur la carte avec MCXDOS.

I.2 Contenu du manuel

Au travers 3 chapitres largement documentés, ce manuel vous apportera toutes les notions jugées indispensables pour exploiter les performances de votre carte et du logiciel MCXDOS. En voici un bref aperçu :

Le premier chapitre vous propose différentes procédures possibles pour charger MCXDOS dans votre carte.

Le second chapitre vous présente les différentes fonctions de INTERMCX.COM et MCX2HOST.COM ainsi qu'un exemple de programmation des cartes MCX et MCX-*Lite/s* côté PC et côté carte.

Enfin, le troisième chapitre vous présente les utilitaires associés à MCXDOS.

Chaque chapitre a été étudié avec soin; par conséquent, une lecture séquentielle est recommandée.

II. CHARGEMENT DU DOS DANS LA CARTE

Trois possibilités vous sont proposées pour charger le système d'exploitation dans votre carte :

- Depuis le lecteur de disquette A: du système hôte
- Depuis un fichier de chargement
- Depuis une partition du disque C: du système hôte

II.1 Etapes préliminaires

Avant de lancer MCXDOS, les 2 étapes suivantes sont nécessaires :

II.1.1 Vérification de la configuration des cavaliers

Assurez-vous que les cavaliers de votre carte sont positionnés comme indiqué ci-dessous :

ST1 en 1-2	**ST2 en 2-3**	**ST3 en 2-3**	**ST4 en 1-2**	**ST5 en 1-2**

NOTE IMPORTANTE :

Pendant le test à la mise sous tension (voir manuel intitulé « Manuel d'installation et caractéristiques techniques des cartes de la gamme MCX »), les LEDS 0 à 7 s'allument successivement et lentement pour s'arrêter définitivement par un « HALT » CPU lorsqu'une erreur est rencontrée. Le code de l'erreur est alors affiché. **Attention, en mode MCXDOS, le code 94h indique le chargement du DOS dans la carte**. **Ce code n'est donc pas un code d'erreur**.

Lorsque le test a été effectué correctement toutes les leds sont éteintes, alors que dans le mode de compatibilité MCC, les LEDS 0 à 7 s'allument rapidement de bas en haut, puis dans le sens contraire, indiquant que la carte attend maintenant son code de départ (« `RUN 01` » par exemple).

II.1.2 Chargement du fichier INTERMCX.COM

Avant de lancer MCX, il est nécessaire de charger le programme INTERMCX.COM. Celui-ci assure l'interface logicielle entre la carte et les applications côté host. Il utilise l'interruption 8Fh. Les paramètres suivants doivent être spécifiés au lancement :

- Adresse de la boîte aux lettres
- Adresse du port I/O
- Numéro d'IRQ utilisée

Exemple : Boîte aux lettres en D000h, Port I/O en 340h, IRQ 15 :

```
INTERMCX D000h, 340h, 15
```

Quelle que soit la méthode utilisée pour charger MS-DOS dans la carte, le support utilisé doit être formaté « système », comme pour n'importe quel P.C. Il peut contenir un fichier CONFIG.SYS et **doit** contenir un fichier AUTOEXEC.BAT permettant le lancement du programme **MCX2HOST.COM**. Ce programme est absolument nécessaire pour assurer le partage des ressources du P.C. Dès son lancement, la carte a accès à tous les disques logiques du P.C. hôte.

II.2 Chargement depuis une disquette

La carte peut charger MS-DOS depuis une disquette système placée dans le lecteur A: du système hôte. Pour assurer le partage des ressources, le fichier AUTOEXEC.BAT de la disquette doit contenir la ligne de lancement du programme MCX2HOST. Ce programme résident assure l'interface avec le système hôte, ainsi que la gestion du clavier français. Le gestionnaire de clavier du DOS (KEYB.COM) **ne doit pas être chargé** dans la carte, et ne doit par conséquent pas figurer dans les fichiers CONFIG.SYS et AUTOEXEC.BAT. En revanche, l'utilisation de DOSKEY ne pose pas de problème.

REMARQUE : Le nombre de fichiers pouvant être ouverts simultanément, spécifié par la commande FILES = n du fichier CONFIG.SYS, doit être augmenté, côté P.C., du nombre de fichiers spécifié côté carte. Par exemple, si le CONFIG.SYS du P.C. contient la ligne 'Files = 30' et le CONFIG.SYS de la carte (sur la disquette) la ligne 'Files = 20', changez au niveau du P.C. pour 'Files = 50'.

Exemple :

- Config.sys

```
FILES= 30
BUFFERS= 10
LASTDRIVE= F:
COUNTRY= 033,,A:\DOS\COUNTRY.SYS
DEVICE= A:\DOS\ANSI.SYS
```

- Autoexec.bat

```
ECHO OFF
PROMPT Mcx $P$G
PATH=A:\DOS; A:\UTILS
MCX2HOST
REM LES UNITES LOGIQUES DU PC A: à F: SONT MAINTENANT ACCESSIBLES
C:
APPLI.EXE
...
```

REMARQUES:
La commande PROMPT Mcx PG permet de différencier facilement l'écran MCX de l'écran P.C. La commande SET COMSPEC indique à l'interpréteur de commandes de recharger sa zone temporaire depuis le disque dur et permet d'ôter la disquette du lecteur.

Pour lancer le chargement, exécutez le programme INTERMCX.COM (voir ci-dessus), introduisez la disquette dans le lecteur A: puis lancez le programme MCXDOS. Lorsque la carte a terminé son chargement, tapez ALT-X pour donner le contrôle à la carte.

ATTENTION ! Terminez toute application en mode graphique avant de taper ALT-X. Le P.C. doit impérativement être en mode TEXTE.

II.3 Chargement depuis un fichier

Cette méthode est plus facile à mettre en oeuvre que l'utilisation d'une partition du disque dur, et elle offre la même rapidité de chargement, c'est pourquoi elle devra être utilisée en priorité. Le fichier de chargement, ou « bootfile », est vu par la carte en tant que lecteur de disquettes A: Ce fichier est crée à partir d'une disquette classique qui doit, comme dans le cas précédent, être « bootable » et contenir un fichier AUTOEXEC.BAT permettant le lancement de MCX2HOST.COM. Il n'est cependant plus possible pour la carte d'accéder au lecteur de disquette réel du P.C., celui-ci étant remplacé par le fichier de chargement.

Le fichier de chargement doit être créé dans le répertoire à partir duquel MCXDOS sera lancé. Copiez vos programmes (principalement les drivers) sur une disquette système, ainsi que le programme MCX2HOST.COM lancé par le fichier AUTOEXEC.BAT.

Lorsque la disquette système est prête, testez la en utilisant la méthode précédente, puis lancez, depuis le répertoire du disque dur contenant MCXDOS, le programme BOOTFILE.COM suivi de la désignation de l'unité dans laquelle se trouve la disquette. (Ex : BOOTFILE B:). Le programme crée une image de la disquette dans un fichier nommé MCXBOOT. Ce fichier peut être renommé, à l'aide de la commande RENAME, dans le cas ou vous désirez utiliser plusieurs fichiers de chargement.

MCXDOS peut maintenant être lancé, en indiquant le nom du fichier de boot, précédé du caractère @ (exemple: MCXDOS @mcxboot2). Pour passer du P.C. hôte à la carte et réciproquement , tapez ALT-X.

II.4 Chargement depuis une partition du disque C:

Cette méthode nécessite une bonne compréhension du processus mis en oeuvre et ne doit être appliquée, si nécessaire, qu'après parfaite maîtrise des deux autres méthodes.

Pour partager les ressources, sous DOS, il n'est pas possible de faire « booter » la carte sur la partition principale du disque du système: d'une part les drivers et programmes à lancer côté carte sont probablement différents de ceux du système hôte, d'autre part le programme MCX2HOST indispensable au partage des ressources ne peut être lancé **que** depuis la carte.

La solution consiste à charger MS-DOS depuis la première partition DOS étendue du disque **C:** du système hôte (la version actuelle ne permet pas le chargement depuis le deuxième disque physique **D:**). Pour le système hôte, cette partition est l'unité logique **D:** si le système ne possède qu'un disque dur, ou l'unité logique **E:** si le système possède deux disques (l'unité logique **D:** étant dans ce cas la partition principale du deuxième disque dur). Pour la carte, jusqu'au lancement de MCX2HOST, cette partition sera vue par celle-ci comme la partition DOS principale, unité **C:**. Après lancement de MCX2HOST, elle reprend son nom d'unité logique réelle **D:** ou **E:**, alors que **C:** désigne désormais la véritable partition principale. Il convient donc d'être extrêmement prudent lors de la création des fichiers de chargement.

Création de la partition de boot MCX :

- A l'aide du programme FDISK de MS-DOS, créez sur le disque **C:** une partition principale, puis une partition étendue. Cette dernière ne doit contenir qu'une seule unité logique.

- Après avoir relancé la machine, à la sortie de FDISK, formatez les unités logiques **C:** (partition principale) et **D:** (partition étendue).

- Créez le répertoire D:\MCXDOS pour y copier le contenu de la disquette de distribution MCXDOS.

- Depuis ce répertoire, tapez la commande :

  SYSMCX mbadr ioadr

  Avec mbadr = Adresse de la boîte aux lettres. Ex: D000h
  ioadr = Adresse du port I/O. Ex: 280h

  Cette commande copie le système sur le disque dur. Elle peut prendre quelques secondes.

- Créez dans le répertoire principal de la partition étendue **D:** le fichier CONFIG.SYS contenant les drivers à exécuter par la carte. N'oubliez pas que lors de l'exécution du CONFIG.SYS, la carte ne voit **que** cette partition, et qu'elle la voit en tant qu'unité **C:** (voir exemple ci-dessous).

- Créez dans ce même répertoire un fichier AUTOEXEC.BAT contenant UNIQUEMENT la ligne suivante :

  MCXEXEC

- Créez enfin le fichier MCXEXEC.BAT, qui doit contenir dans les premières lignes le lancement du programme MCX2HOST.COM. Dès que ce programme à été lancé, la carte a accès aux unités logiques du système hôte, sous leur véritable nom d'unité : la partition étendue sur laquelle elle a « booté » en tant que **C:** redevient **D:**, alors que **C:** désigne désormais la véritable partition principale du P.C. hôte. Comme dans les cas précédents, il est judicieux d'introduire dans ce fichier une commande du type :

  PROMPT Mcx pg

  Ceci permettra de distinguer plus aisément l'écran MCX de l'écran du P.C. hôte.

- Vous pouvez maintenant lancer INTERMCX (voir ci-dessus) puis :

  MCXDOS -p2

  L'option -p2 indique à la carte de « booter » sur la partition DOS étendue. Après quelques secondes, vous pouvez taper ALT-X pour « donner la main » à la carte.

Exemple de fichier CONFIG.SYS :

```
BREAK= ON
FILES= 30
BUFFERS= 10
LASTDRIVE= F
SHELL= C:\DOS\COMMAND.COM C:\DOS\ /E:1024 /p
DEVICE C:\DOS\ANSI.SYS
```

Remarques :
Le répertoire DOS contenant ANSI.SYS doit en réalité se trouver sur D: En revanche, la commande SHELL indique à MS-DOS le chemin d'accès à l'interpréteur de commandes. Ce chemin doit être valide **avant et après** le lancement de MCX2HOST. Par conséquent, les unités C: et D: doivent toutes deux comporter un répertoire \DOS contenant COMMAND.COM.
La commande LASTDRIVE = F donne à la carte l'accès aux unités logiques du système hôte (après lancement de MCX2HOST) jusqu'à l'unité F si elle existe. Indiquez le nom de la dernière unité logique valide sur votre système.

Exemple de fichier MCXEXEC.BAT :

```
@echo off
C:\MCXDOS\MCX2HOST
PATH C:\DOS;D:\UTIL;
PROMPT $E[1;34mMcx $P$G $E[;32m
...
```

II.5 Utilisation d'un disque virtuel

L'utilisation d'un disque virtuel en RAM sur la carte, avec le programme RAMDRIVE.SYS de MS-DOS par exemple, pose le problème suivant : Ramdrive.sys attribue au disque virtuel le premier nom d'unité disponible. Au moment du traitement de CONFIG.SYS, le DOS de la carte ne connaît que l'unité **C:**. Le disque virtuel se voit donc attribuer le nom d'unité **D:**. Après exécution de MCX2HOST.COM, les accès aux unités logiques sont redirigés vers le système hôte, y compris pour l'unité **D:**, et le disque virtuel n'est plus accessible.

Il est possible de résoudre ce problème à l'aide du programme LASTUNIT.SYS. Celui-ci permet de réserver les noms des unités que le P.C. doit partager avec la carte. Par exemple, si le système dispose d'un disque dur contenant deux unités logiques **C:** et **D:**, d'un disque virtuel **E:** et d'un disque réseau **F:**, la ligne de commande

```
DEVICE LASTUNIT.SYS = F
```

placée dans le CONFIG.SYS de la carte permet de réserver les noms d'unités jusqu'à **F:**. Ainsi, au lancement de RAMDRIVE.SYS, celui-ci utilisera le prochain nom d'unité disponible, c'est à dire **G:** (à condition d'avoir un lastdrive au moins égal à G). De plus, pour éviter que **G:** soit redirigé vers le système hôte, il doit être exclu lors du lancement de MCX2HOST, par la commande :

```
MCX2HOST  -G
```

Exemple : Installation d'un disque virtuel **H:** sur la carte. Les unités **C:** à **G:** restent accessibles par la carte si elles existent sur le système hôte.

- Config.sys

```
BREAK = ON
FILES = 30
BUFFERS = 10
LASTDRIVE = H:
DEVICE C:\DOS\ANSI.SYS
DEVICE C:\MCXDOS\LASTUNIT.SYS = G
DEVICE C:\DOS\RAMDRIVE.SYS 1024 /e
```

- Mcxexec.bat

```
@echo off
C:\MCXDOS\MCX2HOST -H
VERIFY OFF
PROMPT $p$g
 ...
```

II.6 Options de lancement du programme MCXDOS.EXE

MCXDOS [-?] [-d] [-k scan stat] [-f id] [-pn] [@pathname]

-? Aide sur les options de lancement
-d Supprime MCXDOS de la mémoire
-k Définition de la touche d'activation :
scan représente le « scan code » de la touche
(Ex: A = 16, B = 48, C = 46, D = 32, E = 18...)
stat est une combinaison de :
1 = Shift de droite
2 = Shift de gauche
4 = Control
8 = Alt
-f Définition du code d'identification (id = 0 à 15) pour l'INT 2FH
-m Autorise le partage de la souris
-p Permet de booter sur la partition n (si pas de disquette dans A:)
-q Utilisation d'un clavier QWERTY
@pathname = Chemin d'accès et nom du fichier de boot

Exemples :

- Boot sur fichier mcxboot2, Activation par CTL-ALT-A:

```
mcxdos -k 16 12 @mcxboot
```

- Boot sur partition étendue, Activation par ALT-RIGHT SHIFT-M :

```
mcxdos -k 39 9 -p2
```

- Boot sur disquette dans le lecteur A:, Activation par CTL-V, clavier QWERTY, partage de la souris :

```
mcxdos -k 47 4 -q -m
```

- Suppression de MCXDOS de la mémoire. La carte n'est pas ré initialisée :

```
mcxdos -d
```

II.7 Options de lancement du programme MCX2HOST.COM

```
MCX2HOST  /k  -n
```

/k Autorise l'emploi d'un clavier connecté directement sur la carte[1]. Attention, après activation de MCXDOS par ALT-X, les deux claviers sont actifs simultanément. Il peut en résulter une désynchronisation entre les indicateurs à leds des modes NUM-LOCK et CAPS-LOCK du clavier et les modes effectivement activés.

-n Interdit la redirection de l'unité logique n:

L'ordre des paramètres doit être respecté.

II.8 Reset de la carte

Il est possible de relancer la carte en tapant CTL-ALT-DEL. Le logiciel redonne automatiquement le contrôle au P.C.; après quelques secondes, la carte « reboote » et il est à nouveau possible de taper ALT-X. Si cette méthode ne fonctionne pas correctement (notamment avec certains utilitaires tels que QEMM386 de QuarterDeck), utilisez le programme REBOOT.COM côté carte ou le programme RESETMCX.COM côté host.

[1] Uniquement pour les cartes MCX.

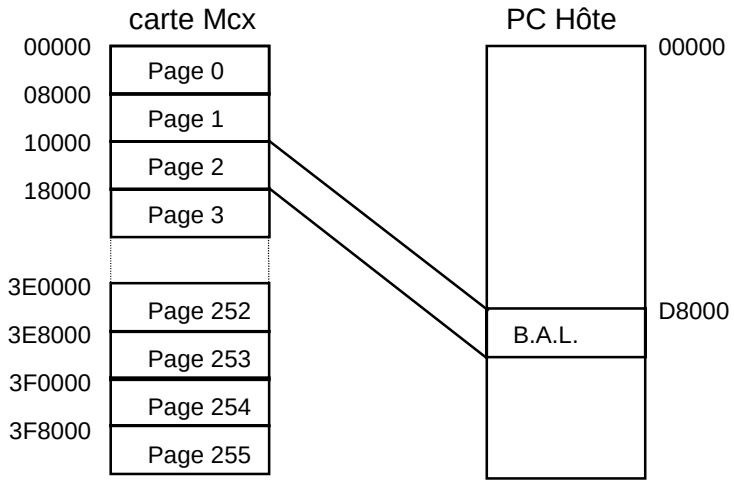
II.9 DEVELOPPEMENT D'APPLICATIONS

MCXDOS utilise un système d'échange d'informations sous interruption entre la carte MCX et le PC hôte, ainsi qu'un espace de 32 K octets, côté PC, lui permettant d'accéder à la mémoire de la carte MCX. L'adresse physique de cette fenêtre, ou « boîte aux lettres », est fixée par le bloc d'inter DIL SW3. La mémoire de la carte est virtuellement divisée en 256 pages logiques de 32Koctets :

La page logique 0 correspond au champ d'adresses	00000h	à	07FFFh
La page logique 1 correspond au champ d'adresses	08000h	à	0FFFFh
La page logique 2 correspond au champ d'adresses	10000h	à	17FFFh
La page logique 3 correspond au champ d'adresses	18000h	à	1FFFFh

...

Toute page logique peut être « mappée » dans la fenêtre du PC, par simple écriture de son numéro sur un port de sortie. Le PC peut par conséquent lire et écrire dans n'importe quelle page logique de la carte (0 à 4Mo).



Visualisation de la page logique 2 dans la boîte aux lettres

Les pages logiques 0 à 13h correspondent à la mémoire système de la carte (640Ko, adresse 0 à 9FFFFh). La page logique 16 est la mémoire d'écran MCX.

La page logique 1Ah (adresse MCX D0000h) contient 32 Koctets de mémoire RAM; C'est la page placée initialement dans la boîte aux lettres. Cette zone mémoire présente l'avantage de ne pas faire partie de la mémoire système, et par conséquent ne peut pas être écrasée par le chargement d'une application. C'est pourquoi elle est utilisée pour les transferts d'informations entre la carte et le PC. Mcxdos n'utilise que les adresses D000:5000h à D000:7FFFh. Les adresses D000:0h à D000:4FFFh sont libres et disponibles pour vos applications.

ATTENTION : Si vous utilisez coté MCX un gestionnaire de mémoire tel que EMM386 ou QEMM, la mémoire 'remappée' par le CPU de la carte MCX dans les UMB n'est pas visible depuis le PC (en réalité, elle est visible, mais à son adresse réelle, au delà du premier méga).

Page	Adresses
00h	0000:0-7FFFh
01h	0800:0-7FFFh
02h	1000:0-7FFFh
03h	1800:0-7FFFh
04h	2000:0-7FFFh
05h	2800:0-7FFFh
06h	3000:0-7FFFh
07h	3800:0-7FFFh
08h	4000:0-7FFFh
09h	4800:0-7FFFh
0Ah	5000:0-7FFFh
0Bh	5800:0-7FFFh
0Ch	6000:0-7FFFh
0Dh	6800:0-7FFFh
0Eh	7000:0-7FFFh
0Fh	7800:0-7FFFh

Page	Adresses
10h	8000:0-7FFFh
11h	8800:0-7FFFh
12h	9000:0-7FFFh
13h	9800:0-7FFFh
14h	A000:0-7FFFh
15h	A800:0-7FFFh
16h	B000:0-7FFFh
17h	B800:0-7FFFh
18h	C000:0-7FFFh
19h	C800:0-7FFFh
1Ah	D000:0-7FFFh
1Bh	D800:0-7FFFh
1Ch	E000:0-7FFFh
1Dh	E800:0-7FFFh
1Eh	F000:0-7FFFh
1Fh	F800:0-7FFFh

Correspondance entre numéros de page et adresses Mcx

Au lancement, la carte MCX positionne la boîte aux lettres sur la page logique 1Ah (adresse Mcx D000:0-7FFFh). Le positionnement ultérieur sur une autre page logique ne peut être effectué que par le PC, simplement par écriture du numéro de page désiré à l'adresse base + 1 du port d'entrées/sorties de la carte. Lors du fonctionnement sous Mcxdos, cette opération doit **toujours** être effectuée au moyen des fonctions 2, 4, 6 ou 7 de l'interruption 8Fh installée par Intermcx.com , **jamais** par écriture directe sur le port. Par exemple, si votre application a besoin de lire le contenu de la page logique 8 (adresse Mcx 4000:0h - 7FFFh de la carte MCX), appelez l'interruption 8Fh avec les paramètres suivants :

AH = 02 Fonction 02
AL = 08 Page logique 8

Toutes les fonctions de l'interruption 8Fh effectuant un changement de page logique renvoient l'ancien numéro de page dans le registre AL. Vous devez sauvegarder ce numéro, afin de repositionner la boîte aux lettres sur l'ancienne page logique dès que votre opération est terminée.

Si votre application, coté PC hôte, nécessite l'accès à une adresse mémoire précise de la carte Mcx, la fonction 6 de l'interruption 8Fh calcule pour vous le numéro de page logique correspondant, positionne cette page dans la boîte aux lettres, et revient avec l'adresse correspondante de la boite aux lettres dans les registres CX et BX. Par exemple, si vous désirez examiner le contenu d'une table située à l'adresse 3500:2480h de la mémoire Mcx, appelez l'interruption 8Fh avec :

AH = 06H Fonction 06
BX = 2480H Offset de la table dans la mémoire Mcx
CX = 3500H Segment de la table dans la mémoire Mcx

Au retour, la boite aux lettres est positionnée sur la page logique 06 (adresse Mcx 3000:0h-7FFFh) et les registres contiennent :

AL = 1Ah	Numéro de page précédent, ici 1Ah
BX = 7480h	Offset de la table dans la boîte aux lettres
CX = D000h	Segment de la boîte aux lettres (selon config)

Les valeurs de CX et BX peuvent être utilisées pour constituer un pointeur 'FAR' sur la table, en prenant soin de rester dans les limites de la boîte aux lettres : la dernière adresse Mcx disponible dans cette page est 3000:7FFFh (= 3500:2FFFh) c'est à dire D000:7FFFh pour le PC. Si vous devez aller au delà, il vous faut passer à la page suivante (à l'aide de la fonction 7 par exemple). N'oubliez pas de repositionner la fenêtre sur l'ancienne page, dont vous avez sauvegardé le numéro, à l'aide de la fonction 2 de l'Int 8Fh.

REMARQUE : Si vous utilisez un gestionnaire de mémoire étendue tel que QEMM386, 386MAX ou EMM386 côté Mcx, vous **devez** exclure la page 1Ah des UMB, à l'aide de la commande **X=D000-D7FF** dans la ligne d'invocation du gestionnaire de mémoire, dans le fichier CONFIG.SYS.

Mcxdos utilise également, pour son propre fonctionnement, la ligne d'interruption de la carte vers le PC (définie par SW2), et la ligne d'interruption du PC vers la carte (Mcx IRQ9 = Int 72h). Si votre application requiert également un dialogue sous interruption entre la carte et le système, Intermcx.com côté PC et Mcx2host.com côté Mcx proposent un certain nombre de fonctions autorisant le partage des lignes d'interruption avec Mcxdos. Elles permettent notamment d'associer des routines de traitement à des commandes passées par interruption du PC vers la MCX et de la MCX vers le PC. 32 commandes utilisateur (80h à 9Fh) sont disponibles dans les deux sens. Les routines associées aux numéros de commande sont appelées directement par les programmes d'interruption de Mcxdos et de Mcx2host.

II.10 Le programme INTERMCX.COM

Le programme Intermcx.com est une interface logicielle entre la carte et les applications côté host. Il permet de développer des applications sans tenir compte des adresses mémoire et I/O de la carte. Il est indispensable pour le fonctionnement de MCXDOS, mais peut être utilisé dans d'autres contextes. Nous recommandons l'utilisation systématique des fonctions de changement de page de INTERMCX, notamment lorsque ceux-ci se produisent sous interruption et dans les applications multi-tâches. Les applications appellent les fonctions de INTERMCX par l'interruption 8FH. Les paramètres suivants doivent être spécifiés au lancement :

- Adresse de la boîte aux lettres
- Adresse du port I/O
- Numéro d'IRQ utilisée

Exemple : Boîte aux lettres en D000h, Port I/O en 340h, IRQ 15 :

```
INTERMCX  D000h, 340h, 15
```

II.11 Description des fonctions de INTERMCX - INT 8FH

Fonction 00 - retourne la configuration :

Entrée : *ah = 00*
Sortie : *al = 0 si numéro d'IRQ de la carte < 8, 1 si IRQ > 8 (Utilisation du PIC 2).*
bh = Masque d'autorisation de l'IRQ de la carte pour le PIC. Si al = 1, concerne le PIC 2 (adresse A1h).
bl = Niveau d'interruption de la carte.
dx = Adresse du port I/O de la carte.
cx = Segment de la fenêtre mémoire double accès (Boite aux lettres)
si = Offset paramètres des commandes 80h - 9Fh
di = Segment paramètres des commandes 80h - 9Fh

Exemple: Carte à l'adresse I/O 300h, MB en D800h, sur IRQ 11. Valeurs retournées :

al = 01	dx = 300h
bh = 0F7h	cx = D800h
bl = 73h	si = 42h
	di = DFF0h

Fonction 01 - Retourne le numéro de page logique courant de la boîte aux lettres.

Entrée : *ah = 01*
Sortie : *al = Numéro de page*

Exemple: Boîte aux lettres sur l'adresse D000:0000h de la carte, soit la page logique numéro 26 (1Ah). Valeur retournée : al = 1Ah.

Fonction 02 - Changement de page logique. Retourne le numéro de page précédent.

Entrée : *ah = 02*
al = Nouveau numéro de page.
Sortie : *al = Ancien numéro de page.*

L'ancien numéro de page permet de restituer le contexte, à la sortie d'un programme d'interruption par exemple.

Fonction 03 - Retourne le segment MCX correspondant a la page logique visible dans la boîte aux lettres (B.A.L.)

Entrée : *ah = 03*
Sortie : *cx = Segment de l'adresse MCX (0, 800, 1000, 1800 etc..)*

Exemple: La boîte aux lettres pointe sur la page logique n° 26 (1Ah), soit le champ d'adresse MCX D000:0000 - D000:7FFFh. La fonction 03 renvoie la valeur du segment :

cx = D000h.

Fonction 04 - Change le segment de l'adresse MCX visible dans la B.A.L. et retourne le segment correspondant côté host.

Entrée : *ah = 04*
cx = Segment de l'adresse MCX
Sortie : *al = Ancien numéro de page.*
cx = Segment côté host.

La mémoire de la carte est découpée en 256 pages logiques de 32Ko :

Page 0 = Adresse 0000:0000 - 0000:7FFF
Page 1 = Adresse 0800:0000 - 0800:7FFF
Page 2 = Adresse 1000:0000 - 1000:7FFF
Page 3 = Adresse 1800:0000 - 1800:7FFF
etc.

Si le segment demandé ne correspond pas à celui d'une page logique, la fonction 4 détermine le numéro de page dans laquelle il démarre, puis calcule sa position dans la boîte aux lettres et retourne le segment HOST correspondant dans cx.

Exemple: Segment B.A.L. en D000h, segment MCX demandé = 1900h. Le segment multiple de 800h (32Ko) le plus proche est 1800h, le nouveau numéro de page logique est donc 3. Le segment demandé est visible à l'offset 1000h de la B.A.L. (Seg 1900h - 1800h). La valeur retournée dans cx est par conséquent : D100h (<=> D000:1000h).

Fonction 05 - Retourne l'offset dans la boîte aux lettres de l'adresse MCX demandée. Retourne ax = 0 si cette adresse n'est pas comprise dans la page logique courante. Ne change pas le numéro de page logique.

Entrée : *ah = 05*
bx = Offset de l'adresse MCX désirée
cx = Segment de l'adresse MCX
Sortie : *bx = Offset dans la B.A.L. (coté host).*
cx = Segment de la B.A.L.

Exemple : Segment B.A.L en D000h, Adresse MCX demandée = 0040:006Ch, page logique courante = 0. Valeurs retournées :

```
ah =  05          ; L'adresse demandée est dans la page logique
bx =  46Ch        ; Offset dans la B.A.L. = offset dans seg 0000h
cx =  D000h       ; Segment de la B.A.L.
```

Fonction 06 - Comme la fonction 05 mais change le numéro de page logique si l'adresse MCX demandée n'est pas dans le champ de la page courante.

Entrée : *ah = 06*
bx = Offset de l'adresse MCX désirée
cx = Segment de l'adresse MCX
Sortie : *al = Ancien numéro de page.*
bx = Offset dans la B.A.L. (coté host).
cx = Segment de la B.A.L..

Fonction 07 - Incrémente le numéro de page de la valeur spécifiée dans le registre al. Si la valeur est négative (ex: FDh = -3), le numéro de page est décrémenté.

Entrée : *ah = 07*
al = Valeur de décalage.
Sortie : *al = Ancien numéro de page.*

Fonction 08 - Lecture de la mémoire FIFO de la carte. Si elle est vide, retourne ax = 0.

Entrée : *ah = 08*
Sortie : *al = Valeur lue ou :*
ax = 0 si FIFO vide.

Fonction 09 - Envoi d'une interruption à la carte.

Entrée : *ah = 09*
Sortie : *Pas de valeur retournée*

Fonction 10 - Envoi d'une commande à la carte. Cette fonction n'attend pas que la carte ait exécuté la commande.

Entrée : *ah = 0Ah*
al = Commande (80h ... 9Fh)
bx = Paramètre 1
cx = Paramètre 2
dx = Paramètre 3
si = Paramètre 4
di = Paramètre 5
Sortie : *ax = 0 si la commande n'a pas pu être envoyée (commande en cours)*

Fonction 11 - Envoi d'une commande à la carte avec paramètres en retour. Cette fonction attend que la commande ait été exécutée par la carte.

Entrée : *ah = 0Bh*
al = Commande (80h ... 9Fh)
bx = Paramètre 1
cx = Paramètre 2
dx = Paramètre 3
si = Paramètre 4
di = Paramètre 5

Sortie : *ax = 0 si la commande n'a pas pu être envoyée (commande en cours)*
bx = Paramètre 1
cx = Paramètre 2
dx = Paramètre 3
si = Paramètre 4
di = Paramètre 5

Fonction 12 - Lecture de l'adresse de la routine de traitement de la commande spécifiée (commandes 80h à 9Fh). Voir fonction 13.

Entrée : *ah = 0Ch*
al = Commande (80h ... 9Fh)

Sortie : *ax = 0 si commande < 80h ou commande > 9Fh*
bx = Offset de la routine de traitement
cx = Segment de la routine de traitement

Fonction 13 - Définition de l'adresse de la routine de traitement de la commande spécifiée (commandes 80h à 9Fh). La routine de traitement peut être appelée par la fonction INTERMCX numéro 14. MCXDOS utilise les fonctions 12, 13 et 14 pour la gestion des interruptions « client ».

Entrée : *ah = 0Dh*
al = Commande (80h ... 9Fh)
bx = Offset de la routine de traitement
cx = Segment de la routine de traitement

Sortie : *ax = 0 si commande < 80h ou commande > 9Fh*

Fonction 14 - Suppression d'une routine de traitement de commande.

Entrée : *ah = 0Eh*
al = Commande (80h ... 9Fh)

Sortie : *ax = 0 si commande < 80h ou commande > 9Fh*

Fonction 15 - Appel de la routine de traitement de la commande spécifiée. Celle ci doit se terminer par un RET FAR. Les registres AX, BX, CX, DX, SI, DI, BP et FLAGS sont préservés systématiquement. Les autres registres doivent être préservés s'ils sont utilisés par la routine de traitement.

Entrée : *ah = 0Fh*
al = Commande (80h ... 9Fh)
Sortie : *Pas de valeur retournée.*

Fonction 16 - Convertit une adresse de la boîte aux lettres en adresse MCX relative au segment MCX passé dans cx...

Entrée : *ah = 10*
bx = Offset dans la boîte aux lettres.
cx = Segment MCX
Sortie : *bx = Offset dans le segment MCX (cx).*
cx = Segment MCX inchangé.
ax = 0 si l'adresse ne fait pas partie du segment demandé

Exemple: Segment B.A.L. en 4800h (PL = 9), Offset dans MB = 3640h. On veut l'offset équivalent côté carte, dans le segment 4100h. Valeurs retournées :

```
ah = 10h          ; L'adresse est dans le segment demandé
bx = A640h        ; Offset relatif à 4100h
cx = 4100h        ; Segment MCX demandé
```

En effet 4800:3640h est équivalent à 4100:A640 (= 4B64:0000)

Fonction 17 - RESET de la carte

Entrée : *ah = 11h*
Sortie : *Pas de valeur retournée.*

Fonction 18 - RESERVEE

Fonction 19 - Activation de la routine d'interruption interne pour les applications chargées par AUTOMCX

Entrée : *ah = 13h*
Sortie : *Pas de valeur retournée.*

Fonction 20 - Désactivation de la routine d'interruption interne (applications chargées par AUTOMCX)

Entrée : *ah = 14h*
Sortie : *Pas de valeur retournée.*

II.12 Le programme MCX2HOST.COM

MCX2HOST.COM représente la partie MCX du logiciel MCXDOS. Il permet le partage de l'écran et du clavier du P.C. hôte. Pour le développement d'applications, il offre des fonctions similaires à celles de INTERMCX, appelées cette fois par l'interruption 8Eh.

II.13 Les fonctions de MCX2HOST - INT 8EH

Fonction 00 - retourne la configuration :

Entrée : *ah = 00*
Sortie : *ax = Registre de status*
bx = Segment de commande
cx = Segment de la PL commandes
dx = Adresse base port I/O mcx
si = Offset paramètres des commandes 80h - 9Fh
di = Segment paramètres des commandes 80h - 9Fh

Fonction 05 - Autorisation / interdiction du rafraîchissement écran :

Entrée : *ah = 05*
al = 00 : Autorisation
01 : Interdiction
Sortie : *Pas de valeur retournée*

Fonction 09 - Envoi d'une interruption au P.C.

Entrée : *ah = 09*
Sortie : *Pas de valeur retournée*

Fonction 10 - Envoi d'une commande au P.C. Cette fonction n'attend pas que le P.C. ait exécuté la commande.

Entrée : *ah = 0Ah*
al = Commande (80h ... 9Fh)
bx = Paramètre 1
cx = Paramètre 2
dx = Paramètre 3
si = Paramètre 4
di = Paramètre 5
Sortie : *ax = 0 si la commande n'a pas pu être envoyée (commande en cours)*

Fonction 11 - Envoi d'une commande au P.C. avec paramètres en retour. Cette fonction attend que la commande ait été exécutée par le P.C.

Entrée : *ah = 0Bh*
al = Commande (80h ... 9Fh)
bx = Paramètre 1
cx = Paramètre 2
dx = Paramètre 3
si = Paramètre 4
di = Paramètre 5

Sortie : *ax = 0 si la commande n'a pas pu être envoyée (commande en cours)*
bx = Paramètre 1
cx = Paramètre 2
dx = Paramètre 3
si = Paramètre 4
di = Paramètre 5

Fonction 12 - Lecture de l'adresse de la routine de traitement de la commande spécifiée (commandes 80h à 9Fh). Voir fonction 13.

Entrée : *ah = 0Ch*
al = Commande (80h ... 9Fh)

Sortie : *ax = 0 si commande < 80h ou commande > 9Fh*
bx = Offset de la routine de traitement
cx = Segment de la routine de traitement

Fonction 13 - Definition de l'adresse de la routine de traitement de la commande spécifiée (commandes 80h à 9Fh). La routine de traitement peut être appelée par la fonction INT 8EH numéro 14. Mcx2host utilise les fonctions 12, 13 et 14 pour la gestion des interruptions « client ».

Entrée : *ah = 0Dh*
al = Commande (80h ... 9Fh)
bx = Offset de la routine de traitement
cx = Segment de la routine de traitement

Sortie : *ax = 0 si commande < 80h ou commande > 9Fh*

Fonction 14 - Suppression d'une routine de traitement de commande.

Entrée : *ah = 0Eh*
al = Commande (80h ... 9Fh)

Sortie : *ax = 0 si commande < 80h ou commande > 9Fh*

Fonction 15 - Appel de la routine de traitement de la commande spécifiée. Celle ci doit se terminer par un RET FAR. Les registres AX, BX, CX, DX, SI, DI, BP et FLAGS sont préservés systématiquement. Les autres registres doivent être préservés s'ils sont utilisés par la routine de traitement.

Entrée : *ah = 0Fh*
al = Commande (80h ... 9Fh)
Sortie : *Pas de valeur retournée.*

Fonction 16 - Effectue l'écriture de 0000h à l'adresse 0000:7FFE, permettant la terminaison du programme de chargement AUTOMCX.COM lancé côté PC. Cette fonction doit être appelée au lancement de l'application chargée par cette méthode.

Entrée : *ah = 10h*
Sortie : *Pas de valeur retournée.*

Fonction 17 - Provoque une réinitialisation totale de la carte ('Reset hard').

Entrée : *ah = 11h*

Fonction 18 - Réservée ACKSYS

Fonction 19 - Réservée ACKSYS

Fonction 20 - Réservée ACKSYS

Fonction 21 - Synchronise l'horloge RTC de la carte sur l'horloge du PC. Cette opération est réalisée systématiquement au lancement de MCXDOS, mais des décalages peuvent se produire si les réinitialisations sont peu fréquentes. Lorsque cette fonction est appelée, la carte MCX reprend l'heure et la date du PC hôte.

Entrée : *ah = 15h*
Sortie : *Pas de valeur retournée.*

II.14 Exemple de programmation de la carte MCX-XX ou MCX-*Lite*/S

Programme côté Host

Ce programme envoie en permanence une chaine de caractères sur le canal 1 de la carte MCX. Les caractères sont envoyés par écriture directe dans le buffer d'émission de la carte MCX; on obtient l'adresse de ce buffer par la commande 81h, traitée coté MCX par le programme MCXSAMP.EXE.
Dès que MCXSAMP détecte une différence entre les pointeurs d'entrée et de sortie de son buffer d'émission, il envoie le contenu de ce buffer sur le canal 1.
Lorsque MCXSAMP reçoit sur le canal 1 un Carriage Return (car 13), il envoie une commande 80h au PC. Cette commande est traitée par la routine Str_Received, appelée par le programme d'interruption de MCXDOS grâce aux fonctions 12, 13 et 14 de INTERMCX.

```
#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>

#define END_MB    0x8000                    /* Offset fin de la boîte aux lettres (MB)      */
#define Intermcx(reg)    int86(0x8F, &reg, &reg )    /* Int 8FH Mcx service                */


/* PROTOTYPES DES FONCTIONS */
void far _loadds str_received();           /* Traitement réception chaine                  */
void PagePlus(char);                       /* Décalage page logique de MB                  */


/* GLOBAL */

union REGS reg, irg;                       /* Pour le passage des params avec int 8FH      */

void (far *fp)();                          /* Pointeur sur fonction                        */
int far *Cmd_Ptr= 0;                       /* Pointeur sur parametres des commandes        */
char far *Txpt;                            /* Pointeur sur buffer tx dans segment MB       */
char far *Rxpt;                            /* Pointeur sur buffer rx dans segment MB       */
unsigned int Txbuff;                       /* Adresse début buffer tx dans seg MB          */
unsigned int Rxbuff;                       /* Adresse début buffer rx dans seg MB          */
unsigned int Txend;                        /* Adresse fin buffer tx dans segment MB        */
unsigned int Rxend;                        /* Adresse fin buffer rx dans segment MB        */
unsigned int Rxstrad;                      /* Adresse structure rx coté MCX                */
unsigned int Txstrad;                      /* Adresse structure tx coté MCX                */
unsigned int Rxstrseg;                     /* Segment structure rx coté MCX                */
unsigned int Txstrseg;                     /* Segment structure tx coté MCX                */
int rx_flag;                               /* Flag chaine reçue                            */

char *Tx_Str = "ACKSYS 39-11-62-81\r";
char Rx_Str[256];

struct Buffer {
      unsigned char *ptin;                 /* Pointeur entrée car.                         */
unsigned char *ptout;                      /* Pointeur sortie car.                         */
      unsigned char buffer[256];           /* Buffer 256 caracteres.                       */
      } far *rx, far *tx;
```

```
reg.h.ah = 0;                          /* Int 8Fh, Fonction 00 :          */
Intermcx(reg);                         /* Lecture config                  */
FP_SEG(Cmd_Ptr) = reg.x.di;            /* Segment Cdes (MB+7F0h)          */
FP_OFF(Cmd_Ptr) = reg.x.si;            /* Offset param pour Host int      */

fp= (void far *) str_received;
reg.x.cx = FP_SEG(fp);                 /* traitement de la commande       */
reg.x.bx = FP_OFF(fp);                 /* 80h                             */
reg.x.ax = 0x0D80;                     /* Int 8Fh, Fonction 13 :          */
Intermcx(reg);                         /* Set user subroutine nb 80h      */

reg.x.ax = 0x0B80;                     /* Envoi commande 80h => Mcx       */
Intermcx(reg);                         /* Envoi Int 8Fh                   */
Rxstrad= reg.x.bx;                     /* Adresse struct rx coté MCX      */
Rxstrseg= reg.x.cx;                    /* Segment struct rx coté MCX      */
reg.h.ah = 6;                          /* Conversion en adresse HOST      */
Intermcx(reg);                         /* Envoi Int 8Fh                   */
FP_SEG(rx)= reg.x.cx;                  /* Adresse structure tx mcx        */
FP_OFF(rx)= reg.x.bx;                  /* Segment structure tx mcx        */

reg.h.ah = 1;                          /* Fonction 1 'Get current page'   */
Intermcx(reg);                         /* Envoi Int 8Fh                   */
rxpage = reg.h.al;                     /* Sauvegarde page rx              */

reg.x.ax = 0x0B81;                     /* Envoi commande 81h => Mcx       */
Intermcx(reg);                         /* Envoi Int 8Fh                   */
Txstrad= reg.x.bx;                     /* Adresse struct tx coté MCX      */
Txstrseg= reg.x.cx;                    /* Segment struct tx coté MCX      */
reg.h.ah = 6;                          /* Passage sur la page de tx       */
Intermcx(reg);                         /* Envoi Int 8Fh                   */
FP_SEG(tx)= reg.x.cx;                  /* Adresse structure tx mcx        */
FP_OFF(tx)= reg.x.bx;                  /* Segment structure tx mcx        */

reg.h.ah = 1;                          /* Fonction 1 'Get current page'   */
Intermcx(reg);                         /* Envoi Int 8Fh                   */
txpage = reg.h.al;                     /* Sauvegarde page tx              */

Txend= (int)(&tx->buffer[256]) & 0x7FFF;   /* Adr. fin buffer TX          */
Rxend= (int)(&rx->buffer[256]) & 0x7FFF;   /* Adr. fin buffer RX          */

puts("TEST CARTE MCX AVEC MCXDOS - rev 1.20 - ACKSYS 1992\n\r");
printf("Chaine envoyée : %s\n\r",Tx_Str);
puts("[Echap] pour stopper\n\r");

while (!(kbhit() && (getch()==27))){
   if (tx->ptin == tx->ptout)          /* EMISSION DE CHAINES             */
      {
      pr = 0;                          /* Flag changement de page         */
      reg.x.cx= Txstrseg;              /* Segment structure tx sur MCX    */
      reg.x.bx= (int)(tx->ptin);       /* Pointeur txptin sur MCX         */
      reg.h.ah= 06;                    /* Conversion en adresse Host      */
      Intermcx(reg);                   /* (relative a MB)                 */

      FP_SEG(Txpt) = reg.x.cx;         /* Segment boîte aux lettres       */
       FP_OFF(Txpt) = reg.x.bx;        /* Adresse pointée par txptin      */
```

```
    for (n = 0; n < strlen(Tx_Str); n++)
        {
        if (FP_OFF(Txpt) > END_MB)          /* Dépassement limite de la fenêtre de 32K ?*/
            {                               /* Si oui :                                 */
            PagePlus(1);                    /* Page suivante                            */
            FP_OFF(Txpt) = 0;               /* Et offset 0                              */
            }
        *Txpt++ = Tx_Str[n];                /* Ecriture 1 caractère                     */
        if (FP_OFF(Txpt) == Txend)          /* Test fin de buffer                       */
            {                               /* Si oui :                                 */
            reg.h.ah = 02;                  /* Selection de page                        */
            reg.x.al = txpage;              /* Page tx                                  */
            Intermcx(reg);                  /* Pointe sur page tx                       */
            Txpt = tx->buffer;              /* Pointeur sur debut buffer                */
            }
        }
     reg.h.ah = 16;                         /* Conversion en offset MCX                 */
     reg.x.bx = FP_OFF(Txpt);               /* Nouveau pointeur txin                    */
     reg.x.cx = Rxstrseg;                   /* Mcx segment                              */
     Intermcx(reg);                         /* conversion                               */
     reg.h.ah = 02;                         /* Selection de page                        */
     reg.x.al = txpage;                     /* Page tx                                  */
     Intermcx(reg);                         /* Pointe sur page tx                       */
     (int)(tx->ptin)= reg.x.bx;             /* Ecriture dans struct tx                  */
     printf("TX N°%5ld\r",++ntx);           /* Affichage message                        */
     }

  if (rx_flag)                              /*    RECEPTION DE CHAINES                  */
     {
     rx_len = rx_flag;
     rx_flag = 0;
     reg.h.ah = 02;                         /* Selection de page                        */
     reg.x.al = rxpage;                     /* Page rx                                  */
     Intermcx(reg);                         /* Pointe sur page rx                       */
     reg.x.cx= Rxstrseg;                    /* Segment structure rx sur MCX             */
     reg.x.bx= (int)(rx->ptout);            /* Pointeur rxptout sur MCX                 */
     reg.h.ah= 06;                          /* Conversion en adresse Host               */
     Intermcx(reg);                         /* (relative a MB)                          */
     FP_SEG(Rxpt) = reg.x.cx;               /* Segment boîte aux lettres                */
     FP_OFF(Rxpt) = reg.x.bx;               /* Adresse pointée par txptin               */

     for (n = 0; n < rx_len; n++){
        if (FP_OFF(Rxpt) > END_MB)          /* Dépassement limite fenêtre ?             */
            {                               /* Si oui :                                 */
            PagePlus(1);                    /* Page suivante                            */
            FP_OFF(Txpt) = 0;               /* Et offset 0                              */
            }
        Rx_Str[n] = *Rxpt++;                /* Copie 1 caractère                        */
        if (FP_OFF(Rxpt) == Rxend)          /* Test fin de buffer                       */
            {                               /* Si Oui :                                 */
            reg.h.ah = 02;                  /* Selection de page                        */
            reg.x.al = rxpage;              /* Page rx                                  */
            Intermcx(reg);                  /* Pointe sur page rx                       */
            Rxpt = rx->buffer;              /* Pointeur sur debut buffer                */
            }
        }
```

```
        reg.h.ah = 16;                                          /* Conversion en offset MCX      */
        reg.x.bx = FP_OFF(Rxpt);                                /* Nouveau pointeur rxin         */
        reg.x.cx = Rxstrseg;                                    /* Mcx segment                   */
        Intermcx(reg);                                          /* conversion                    */
        reg.h.ah = 02;                                          /* Selection de page             */
        reg.x.al = rxpage;                                      /* Page rx                       */
        Intermcx(reg);                                          /* Pointe sur page rx            */
        (int)(rx->ptout)= reg.x.bx;                             /* Ecriture dans struct rx       */

        Rx_Str[rx_len-1] = 0;                                   /* Marque de fin de chaine       */
        printf("TX N°%5ld - RX N°%5ld : %s\r",ntx,++nrx,Rx_Str);
        reg.h.ah = 02;                                          /* Selection de page             */
        reg.x.al = txpage;                                      /* Page tx                       */
        Intermcx(reg);                                          /* Pointe sur page tx            */
        }
     }
   reg.x.ax = 0x0B82;                                           /* Envoi commande 81h => Mcx     */
   Intermcx(reg);                                               /* Termine mcxsamp (coté mcx)    */
   reg.x.ax = 0x0E80;                                           /* Suppression de la routine de  */
   Intermcx(reg);                                               /* traitement commande 80h       */
   }
```

DECALAGE PAGE LOGIQUE

```
void PagePlus(char n)
   {
   reg.h.ah = 07;
   reg.h.al = n;
   Intermcx(reg);
   }
```

TRAITEMENT COMMANDE 80h - Réception de chaine

La commande 80h est envoyée par la MCX sur réception d'un carriage return (13).
str_received est appelée par la routine d'interruption de MCXDOS.

PARAMETRE : Longueur de la chaine reçue.

```
void far _loadds str_received()
   {
   _enable();                                   /* Interrupt enabled    */
   rx_flag = Cmd_Ptr[0];                        /* Set flag             */
   }
```

EXEMPLE DE PROGRAMMATION DE LA CARTE MCX AVEC MCXDOS
Programme côté MCX

Ce programme réalise l'installation des routines de traitement des commandes 80h, 81h, et 82h envoyées par le programme HOSTSAMP.EXE depuis le PC. Il effectue également l'initialisation des contrôleurs de communication, puis boucle sur l'émission de caractères sur la voie 1.
La réception est faite sous interruption; sur réception du caractère 13 (0Dh = Carriage return), envoi d'une commande 80h vers le PC (traitée dans HOSTSAMP).

```
#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include "mcx_c.h"

#define sccit       11                                  /* IRQ3 = Interrupt 11 (0Bh)                */
#define IrqOn       0xF7                                /* Autorisation IRQ3 (PIC)                  */
#define IrqOff      0x08                                /* Interdiction IRQ3 (PIC)                  */
#define McxService(reg) int86(0x8E, &reg, &reg )        /* Int 8EH Mcx service                      */


/* PROTOTYPES DES FONCTIONS */
void interrupt far Scc_Rx_Int(void);                    /* SSP IT Reception caracteres              */
void far _loadds Cmd_RxStruct(void);                    /* Traitement commande 80h                  */
void far _loadds Cmd_TxStruct(void);                    /* traitement commande 81h                  */
void far _loadds Cmd_Terminate(void);                   /* traitement commande 82h                  */


union REGS reg, irg;                                    /* Pour le passage des parametres           */
                                                        /* avec mcxlib et INT 8EH.                  */

void (interrupt far *Oldsccit)();
void (far *fp)();
int far *Cmd_Ptr= 0;                                    /* Pointeur sur paramètres des commandes    */
                                                        /* envoyées par le host                     */
int rx_flag= 0;                                         /* Flag reception de chaine. Positionné par la */
                                                        /* routine d'IT reception de caractère lors de */
                                                        /* la reception de carriage return (0dh), et égal */
                                                        /* au nombre de caractères reçus (début du  */
                                                        /* buffer jusqu'a cr)                       */
int Run_Flg= -1;                                        /* Ce flag, remis à zéro par la commande 82h */
                                                        /* du PC, permet d'interrompre l'execution  */
                                                        /* de MCXSAMP depuis le PC                  */

struct Buffer {
      unsigned char *ptin;                              /* Pointeur entrée car                      */
      unsigned char *ptout;                             /* Pointeur sortie car                      */
      unsigned char buffer[256];                        /* Buffer 256 car                           */
      } rx, tx;
```

```
void main()
   {
   int i;
   long nb_trame= 0;
   char v, vmax;

   puts("PROGRAMME DE TEST DE LA CARTE MCX AVEC MCXDOS\n\r");
   puts("Dès que le buffer de transmission contient un caractère à émettre,");
   puts("celui-ci est envoyé sur le canal 1. Envoi d'une commande 80h au PC");
   puts("chaque fois qu'un carriage return est reçu sur le canal 1\n\r");

   rx.ptin = rx.buffer;                                  /* Initialisation des pointeurs    */
   rx.ptout = rx.buffer;                                 /* d'entrée et de sortie des       */
   tx.ptin = tx.buffer;                                  /* buffers de réception et d'      */
   tx.ptout = tx.buffer;                                 /* émission.                       */

   printf("Adresse buffer rx:=%Fp  tx=%Fp \n\n\r",rx.buffer,tx.buffer);

   fp = (void (far *)( )) Cmd_RxStruct;                  /* Pointe sur la routine de        */
   reg.x.cx = FP_SEG(fp);                                /* traitement commande 80h         */
   reg.x.bx = FP_OFF(fp);
   reg.x.ax = 0x0D80;                                     /* Int 8Eh, Fonction 13 :          */
   McxService(reg);                                      /* Set user subroutine 80h         */

   fp = (void (far *)( )) Cmd_TxStruct;                  /* Pointe sur la routine de        */
   reg.x.cx = FP_SEG(fp);                                /* traitement commande 81h         */
   reg.x.bx = FP_OFF(fp);
   reg.x.ax = 0x0D81;                                    /* Int 8Eh, Fonction 13 :          */
   McxService(reg);                                      /* Set user subroutine 81h         */

   fp = (void (far *)( )) Cmd_Terminate;                 /* Pointe sur la routine de        */
   reg.x.cx = FP_SEG(fp);                                /* traitement commande 82h         */
   reg.x.bx = FP_OFF(fp);
   reg.x.ax = 0x0D82;                                    /* Int 8Eh, Fonction 13 :          */
   McxService(reg);                                      /* Set user subroutine 82h         */

   reg.h.ah = 0;                                         /* Int 8Eh, Fonction 00 :          */
   McxService(reg);                                      /* Lecture config                  */
   FP_SEG(Cmd_Ptr) = reg.x.di;                           /* Seg. Commandes (MB+7F0h)        */
   FP_OFF(Cmd_Ptr) = reg.x.si;                           /* Offset param pour Int           */

   for (i=0; i < 256; i++)                               /* Effacement buffers RX & TX      */
      {
      rx.buffer[i] = 255;
      tx.buffer[i] = 255;
      }

   mcxlib(F_CHECK_CHANNEL, &reg, &reg);                  /* Lecture nombre de voies         */
   vmax= reg.h.al;                                       /* vmax = nombre de voies          */

   Oldsccit = _dos_getvect(sccit);                       /* Sauve ancien vecteur            */
   _dos_setvect(sccit, Scc_Rx_Int);                      /* Vecteur d'IT SCC                */
```

```
for (v=1; v <= vmax; v++)                    /* Reset SCC et init des      */
   {                                         /* lignes                     */
   reg.x.ax=0x1F07;                          /* 1200 Bd, 8b, np, 1 stop    */
   reg.h.ch = v;                             /* Numero de canal            */
   reg.h.cl= (v==1) ? 0x80 : 0;              /* Voie 1 = Int RX & Cond.    */
   mcxlib(F_SCC_INIT_PARMS, &reg, &reg);
   }

_asm cli                                     /* Interdit interrutions      */
   mcxlib(F_MASK_READ, &reg, &reg);          /* Lecture masques PIC        */
   reg.h.al &= IrqOn;                        /* Valide IRQ3 (MCX-BP 1)     */
   mcxlib(F_MASK_WRITE, &reg, &reg);         /* Ecriture nouveau mask      */
_asm    sti                                  /* Interruptions autorisées   */

while (Run_Flg)
   {
   if (rx_flag)
      {                                      /* Int 8Eh, Fonction 10h :    */
      reg.x.ax = 0x0A80;                     /* Envoi cde 80h => host      */
      reg.x.bx = rx_flag;                    /* Param = nombre caractère   */
      McxService(reg);                       /* Envoi Int 8Eh              */
      rx_flag = 0;                           /* Raz flag reception         */
      printf("Nombre de trames reçues = %d\r",++nb_trame);
      }

   if (!(tx.ptin == tx.ptout))               /* Si buffer Tx non vide :    */
      {
      reg.x.cx = 0x1FF;                      /* Canal 1,    cl = -1        */
      while (reg.h.cl)
         {
         reg.h.bl = *tx.ptout;               /* caractère en bx            */
         mcxlib(F_SCC_CHAR_WRITE, &reg, &reg);/* Envoi caractère           */
         }
      if (++tx.ptout == &tx.buffer[256])     /* Si fin de buffer ..        */
           tx.ptout = tx.buffer;             /* .. pointeur = debut        */
      }
   if (_kbhit() && (getch()== 27)) Run_Flg = 0;/* Stoppe si Echap          */
   }

_asm cli                                     /* Interdit interruptions     */
   mcxlib(F_MASK_READ, &reg, &reg);          /* Lecture masque PIC         */
   reg.h.al |= IrqOff;                       /* Disable IRQ3 (MCX-BP 1)    */
   mcxlib(F_MASK_WRITE, &reg, &reg);         /* Ecriture nouveau masque    */
_asm sti                                     /* Interruptions autorisées   */

_dos_setvect(sccit, Oldsccit);               /* Restaure ancien vecteur    */

reg.x.ax = 0x0E80;                           /* Int 8Eh, Fonction 14 :     */
McxService(reg);                             /* supprime subroutine 80h    */

reg.x.ax = 0x0E81;                           /* Int 8Eh, Fonction 14 :     */
McxService(reg);                             /* supprime subroutine 81h    */

reg.x.ax = 0x0E82;                           /* Int 8Eh, Fonction 14 :     */
McxService(reg);                             /* supprime subroutine 82h    */
}
```

Fonction Scc_Rx_Int()

Programme d'interruption SCC en réception. Les caractères reçus sont rangés dans le buffer de réception. Lorsque le caractère 13 (cr) est détecté, le nombre de caractères présents dans le buffer est affecté à rx_flag pour envoi d'une IT au PC (depuis main).

```
void interrupt far Scc_Rx_Int()
   {
   _enable();                                          /* Interrupt enabled          */
   irg.h.ch= 0x01;                                     /* Lecture car canal 1        */
   mcxlib(F_SCC_CHAR_READ, &irg, &irg);                /* Lecture caractère          */

   while (!irg.h.cl)                                   /* Test car. present          */
      {
      *rx.ptin++ = irg.h.bl;                           /* Caractère => buffer        */
      if (irg.h.bl == 13)                              /* Carriage Return ?          */
         rx_flag= rx.ptin>rx.ptout ? rx.ptin-rx.ptout : 256 - (rx.ptout-rx.ptin);
      if (rx.ptin == &rx.buffer[256]) rx.ptin = rx.buffer;
      irg.h.ch= 0x01;
      mcxlib(F_SCC_CHAR_READ, &irg, &irg);             /* Autre caractères ?         */
      }
   irg.h.ch= 0x01;                                     /* SCC Canal 1                */
   mcxlib(F_SEND_SCC_EOI, &irg, &irg);                 /* Acquittement int.          */
   irg.h.al= 0;                                        /* PIC #1                     */
   mcxlib(F_INT_CNTRL_EOI, &irg, &irg);                /* Acquittement PIC           */
   }
```

Fonction Cmd_RxStruct()

Appelée par la routine de gestion des interruptions envoyées par le HOST pour la commande 80h : Lecture de l'adresse de la structure rx.

ENTREE : Pas de paramètres
SORTIE : Param1 = Segment de la structure rx
Param2 = Offset de la structure rx

```
void far _loadds Cmd_RxStruct()
   {
   Cmd_Ptr[0] = (int) &rx;
   Cmd_Ptr[1] = (_segment)&rx;
   }
```

Fonction Cmd_TxStruct()

Appelée par la routine de gestion des interruptions envoyées par le HOST pour la commande 81h : Lecture de l'adresse de la structure rx.

ENTREE : Pas de paramètres
SORTIE : Param1 = Segment de la structure tx
Param2 = Offset de la structure tx

```
void far _loadds Cmd_TxStruct()
   {
   Cmd_Ptr[0] = (int) &tx;
   Cmd_Ptr[1] = (_segment)&tx;
   }
```

Fonction Cmd_Terminate()

Appelée par la routine de gestion des interruptions envoyées par le HOST pour la commande 82h : arrêt du programme.

ENTREE : Pas de paramètres
SORTIE : Pas de paramètre

```
void far _loadds Cmd_Terminate()
   {
   Run_Flg= 0;
   }
```

III. LES UTILITAIRES

III.1 BOOTFILE.COM

Le programme Bootfile.com est utilisé pour créer sur le disque dur un fichier image de la disquette de chargement de la carte. Ce fichier, nommé MCXBOOT par le programme, pourra remplacer la disquette de chargement de la carte, permettant de réduire de façon appréciable le temps de chargement (voir chapitre 3). Le fichier de « boot » MCXBOOT peut être renommé par la suite à l'aide de la commande RENAME.

Pour créer le fichier MCXBOOT, introduisez la disquette de chargement dans un des lecteurs du système puis tapez la commande BOOTFILE A: si vous utilisez le lecteur A: ou BOOTFILE B: si vous utilisez le lecteur B:. Le programme détermine automatiquement le type de disquette utilisé.

Après création du fichier de chargement, celui-ci peut être modifié depuis la carte, après chargement de MS-DOS. Il est possible de régénérer la disquette de chargement à partir du fichier MCXBOOT à l'aide de l'option /r

III.2 REBOOT.COM

Ce programme est utilisé pour réinitialiser la carte. Il a la même fonction qu'un séquence Control-Alt-Suppr, cette dernière n'étant pas toujours utilisable depuis la carte MCX (notamment lors de l'utilisation côté carte d'un gestionnaire de mémoire étendu tel que EMM386 de Microsoft ou QEMM de QuarterDeck.

III.3 RESETMCX.COM

Resetmcx.com permet également de réinitialiser la carte, mais cette fois depuis le système hôte. MCXDOS doit fonctionner normalement au moment du lancement de ce programme.

III.4 EMULMCC.COM

Ce programme permet de passer la carte en mode émulation MCC sans modifier la position des cavaliers ST2, ST3 et ST4. Il sera utilisé notamment pour les mises à jour du BIOS de la carte.

III.5 DOSSETUP.EXE

Il s'agit du programme de « setup » du BIOS de la carte. Il permet de modifier certains paramètres de fonctionnement de la carte. Pour l'utilisation de MCXDOS, aucun lecteur de disquette et aucun disque dur ne doivent être déclarés dans le setup. Le type d'écran doit être monochrome. Il est possible d'indiquer à la carte de « booter » en priorité sur le disque dur.

III.6 AUTOMCX.COM

Automcx.com permet de charger un système d'exploitation dans la carte. Il diffère de MCXDOS par les points suivants :

- Pas de partage écran/clavier.
- Le système d'exploitation chargé sur la carte n'est pas nécessairement MS-DOS.
- Le partage de disque n'est valide que pendant la phase de chargement de l'OS.
- Automcx n'est pas un programme résident en mémoire. Il se termine lorsque la carte a chargé son OS.
- Automcx n'utilise pas le signal d'interruption provenant de la carte. Celui-ci est entièrement disponible pour les applications de l'utilisateur.
- Automcx n'utilise pas INTERMCX.COM côté P.C, ni MCX2HOST.COM côté carte. Il revient à l'utilisateur le soin de gérer l'échange d'informations entre ses applications côté carte et côté P.C.

Lancement de AUTOMCX :

Automcx a besoin de connaître l'adresse du port d'entrée/sortie de la carte, l'adresse de la boîte aux lettres et éventuellement la source de chargement de l'OS :

AUTOMCX D000h,280h /2 File

Avec:
- D000h = Adresse de la B.A.L.
- 280h = Adresse du port I/O de la carte.
- /2 = Autorise l'accès à la partition 2 du système hôte. Si aucun nom de fichier de chargement n'est spécifié et s'il n'y a pas de disquette dans le lecteur A:, la carte essaie de « booter » sur cette partition (toujours vue par la carte comme le disque C:).
- File = Nom d'un fichier de chargement créé avec le programme BOOTFILE.COM. Ce fichier remplace le lecteur de disquette A: pour la carte.

Si aucun fichier de chargement n'est spécifié, la carte essaie de « booter » depuis le lecteur de disquette A:, puis sur la partition du disque dur si elle est spécifiée (/n).

Dans le cas du chargement de MS-DOS depuis une partition DOS étendue du P.C. hôte, les fichiers « système » doivent être chargés sur la partition à l'aide de MCXSYS.BAT (voir chapitre 4).

Terminaison de AUTOMCX

Lorsque la carte a terminé le chargement de son système d'exploitation et de l'application, elle doit libérer Automcx en écrivant 00 00 à l'adresse D000h:7FFEh. Il reste néanmoins possible de terminer Automcx par CTRL-C.

III.6.1 Chargement d'applications développées sous MCXDOS

Les applications développées sous MCXDOS utilisent généralement les services de MCX2HOST et de INTERMCX, notamment pour le passage de commandes ou d'informations entre la carte et le PC (INT 8F fonction 10,11,12,13...). Afin de faciliter le chargement de ces applications sous Automcx, nous avons inclus dans Intermcx, à partir de la version 1.22, une routine d'interruption destinée au traitement des commandes utilisateur. En effet, sous MCXDOS, lorsque l'application côté carte envoie une commande utilisateur au PC (commande 80h-9Fh, int 8Eh fonction 10), celle-ci est traitée par la routine d'interruption de MCXDOS, qui effectue, après analyse, l'appel à la routine utilisateur (définie par la fonction 13 de l'int 8Fh). Lorsque l'application 'carte' est chargée avec Automcx, l'application 'PC' a la possibilité d'activer la routine d'interruption de INTERMCX, de façon à conserver les mêmes mécanismes de passage de commandes.

Intermcx peut être lancé indifféremment avant ou après le chargement dans la carte de Mcx2host et de l'application par Automcx. L'activation de la routine d'interruption de Intermcx doit être effectuée par l'application côté PC, en utilisant la fonction 19 (13h) de l'int 8Fh. La fonction 20 (14h) permet de désactiver la routine d'interruption à la clôture de l'application.

Il est possible d'activer la routine d'interruption directement au lancement de Intermcx à l'aide du commutateur /i. Dans ce cas Intermcx doit être lancé impérativement **après** le chargement de l'application dans la carte par Automcx. Exemple :

```
Intermcx D000h,280h,10 /i
```

III.6.2 Utilisation simultanée de plusieurs cartes MCX

Bien qu'il ne soit pas possible d'utiliser plusieurs cartes simultanément sous MCXDOS, cela est tout à fait réalisable avec Automcx. Si les services d'Intermcx sont nécessaires pour les deux cartes, celui-ci devra être chargé deux fois, en spécifiant les adresses et IRQ de chacune des cartes, et en définissant pour la seconde carte le numéro d'interruption logicielle utilisé pour les appels de fonction. Par exemple, pour utiliser deux cartes configurées respectivement aux adresses D000h, port 280h, IRQ 10 et adresse D800h, port 300h, IRQ 12 on utilisera l'interruption logicielle 90h pour la deuxième carte :

```
Intermcx D000h,280h,10          (première carte)
Intermcx D800h,300h,12,90h      (seconde carte)
```

III.6.3 Accès au disques du PC

Les applications chargées par Automcx n'ont pas accès comme sous MCXDOS à toutes les unités logiques du PC. L'accès est limité au lecteur A: (ou fichier 'bootfile') et à la partition principale (Automcx /1) ou étendue (Automcx /2) du disque du PC, toujours vue en tant que disque C:. Dans le cas de l'utilisation de MCX2HOST, le disque C: doit être exclu de la liste des disques redirigés pour conserver sa visibilité. La ligne de commande dans le fichier AUTOEXEC.BAT de la carte doit être par conséquent :

```
MCX2HOST -C
```

III.7 MCXSPY.COM

Ce programme permet de visualiser la page écran de la carte MCX. Pour stopper l’affichage de l’écran MCX appuyez sur une touche quelconque. Lorsqu'il est appelé avec le paramètre '{' , MCXSPY permet également de rediriger la frappe clavier vers la carte MCX. Dans ce cas, pour terminer, appuyer sur 'Ctrl-}'.

Sous MCXDOS, Mcxspy utilise les services de INTERMCX et ne nécessite pas de paramètres supplémentaires. Dans le cas contraire, Mcxspy a besoin de connaître l'adresse du port d'entrée/sortie de la carte MCX et l'adresse de la boîte aux lettres.

Exemple: Carte en C800h, adresse I/O 300h, redirection clavier :

```
MCXSPY C800h,300h {
```

Les paramètres par défaut sont D000h et 280h.

IV. EXTENSION BIOS PCMCIA

A partir de la révision 1.7P, l'extension BIOS PCMCIA pour cartes MCX-XX permet de définir dynamiquement le type de disque PCMCIA utilisé. Il n'est plus nécessaire de reprogrammer la FLASH EPROM.

Les disques PCMCIA fournis par ACKSYS sont préparés, formatés et testés pour la carte MCX et peuvent être utilisés directement. Si le disque provient d'une autre source, il peut être nécessaire de le partitionner à l'aide d'utilitaires Phoenix® , CardSoft® ou équivalent.

Pour configurer la carte MCX, chargez le DOS à l'aide du fichier de boot (Bootfile), puis lancez le programme DOSETUP.EXE. Indiquez les paramètres de votre disque. Par exemple, pour un disque MAXTOR 105 Mo :

```
810 cylindres, 15 têtes, 17 secteurs par piste
```

Quittez DOSETUP en utilisant la touche de fonction F10, pour valider les modifications, puis répondez non (N) à la question "do you want to reboot ?".

Lancez le programme HDTYPE.EXE et sélectionnez le disque et le connecteur ('slot') que vous utilisez. Le connecteur inférieur est le 'slot' 1, le connecteur supérieur est le 'slot' 2. Pour chaque item, l'ancienne valeur est affichée, mais il n'y a pas de valeur par défaut : vous devez entrer un numéro.

REMARQUE IMPORTANTE : Les valeurs du type de disque et du numéro de slot définis par HDTYPE.EXE sont altérées par l'exécution de DOSETUP.EXE. Par conséquent, il est absolument impératif de relancer systématiquement HDTYPE après chaque exécution de DOSETUP, même s'il n'y a pas eu de modification de la configurartion.

Vous pouvez maintenant relancer votre machine ('*Reset hard*' ou coupure de tension) et lancer de nouveau MCXDOS, toujours en utilisant un fichier de chargement. Il peut être nécessaire, à ce point, de formater le disque PCMCIA à l'aide du Format.com de MS-DOS, option ***/s***. Transférez ensuite les drivers et programmes composants votre application, sans oublier MCX2HOST.COM si vous avez l'intention de développer sur la carte.

Relancez de nouveau votre PC, puis lancez MCXDOS, sans option de boot, ainsi il tentera de booter depuis le lecteur de disquette du PC, puis depuis le disque local PCMCIA. Vous pouvez contrôler le déroulement de l'opération à l'aide de MCXSPY.COM

V. NOTES

VI. FICHE ERREUR

Nous avons besoin de vos commentaires et suggestions pour améliorer la qualité et la facilité d'utilisation de nos documentations.

Nous vous serions très reconnaissants de remplir cette fiche d'appréciation, et de nous la retourner. Nous vous en remercions par avance.

Société	Tel
Utilisateur	Fonction
Adresse	
Code Postal	Ville

Indiquez clairement la version de la carte MCX, la version des logiciels et de la documentation dont vous parlez :

Carte MCX	
Carte MCX-*Lite*/S	
Carte MCX-*Lite*/U	
Carte MCX-*Lite*/485	
Révision FLASH EPROM	
Révision Documentation MCXDOS	
Révision Logiciel MCXDOS	