Utilisation de MILO (Alpha Miniloader Howto)

David A. Rusling, david.rusling@reo.mts.dec.com

v0.84, 6 Décembre 1996

Ce document expose le fonctionnement du Miniloader, un programme pour les machines Alpha qui sert à initialiser le système et démarrer Linux. Le "Alpha Linux Miniloader" (pour être exact) est également dénommé MILO.

Contents

1 Introduction

Ce document expose le fonctionnement du logiciel MILO (Miniloader) pour Alpha AXP. Ce logiciel de console a pour fonction d'initialiser le système Alpha AXP, de charger et démarrer Linux et, pour finir, il met un PALcode à disposition de Linux.

1.1 Droits d'auteurs (Copyright)

1.2 Nouvelles versions de ce document

La dernière version de ce document est disponible à l'adresse suivante :

`ftp://gatekeeper.dec.com/pub/Digital/Linux-Alpha/Miniloader/docs` et David Mosberger-Tang a eu l'amabilité de l'inclure sous la forme d'une page Web sur l'excellent site Linux pour Alpha :

`http://www.azstarnet.com/axplinux` .

2 Qu'est-ce que MILO ?

Sur machines Intel, le BIOS configure le système et ensuite charge, depuis le secteur d'amorçage d'un disque DOS, une image à exécuter. Cela est en quelque sorte la fonction principale de MILO sur un système Alpha. Il existe toutefois un certain nombre de différences entre BIOS et MILO, notamment le fait que MILO intègre certains pilotes de périphériques Linux inchangés. MILO est un logiciel appelé FirmWare, à l'inverse de LILO qui dépend du logiciel FirmWare BIOS responsable de son chargement et de son exécution en mémoire centrale. Les principales fonctions de MILO sont :

1. PALcode ;
2. Initialisation du gestionnaire de mémoire. (construction des tables de pages et mise en service de l'adressage virtuel) ;
3. Logiciel d'affichage (Code d'émulation BIOS et TGA (21030)) ;
4. Une partie du noyau Linux. Incluant, par exemple, un gestionnaire d'interruption qui fait office de noyau Linux ;
5. Gestionnaires de périphériques en mode bloc (par exemple le pilote de disquettes) ;
6. Un support des systèmes de fichiers (ext2, MS-DOS et ISO9660) ;
7. Un logiciel d'interface utilisateur (MILO) ;
8. Une interface avec le noyau (configure le HWRPB et la cartographie de la mémoire spécifiques à linux) ;
9. Configuration de la NVRAM (mémoire non volatile) pour la gestion de variables d'environnement.

Les paragraphes suivants décrivent ces fonctionnalités plus en détail.

Le PALcode peut être perçu comme une minuscule couche logicielle qui prépare le processeur Alpha en vue de l'utilisation d'un système d'exploitation spécifique. Il fonctionne dans un mode spécial du processeur (PALmode) qui a certaines limitations mais utilise les instructions standard des processeurs Alpha, plus cinq instructions supplémentaires. De cette manière le processeur Alpha peut exécuter une grande diversité de systèmes d'exploitation tels que Windows NT, OpenVMS, Digital Unix et bien sûr Linux. Le PALcode que MILO utilise (et in extenso Linux lui-même) est, comme le reste de MILO un freeware. Il est inspiré d'un exemple de PALcode pour Digital Unix que Digital fournissait avec ses premières cartes d'évaluation. Les différences entre les PALcode sont dues à des différences qui existent dans la cartographie de la mémoire, dans la gestion des interruptions entre les différentes versions du processeur Alpha (par exemple le 21066 possède une cartographie des entrées-sorties différente de l'association du 21064 avec les contrôleurs E/S de la famille 2107x, en effet le 21066 intègre un équivalent de ce contrôleur sur son support).

Pour que MILO fonctionne correctement il lui faut savoir quelle est la quantité de mémoire disponible, à quel endroit Linux peut éventuellement être chargé en mémoire, et il doit, de plus, être capable d'allouer temporairement de la mémoire pour les pilotes de périphériques Linux. Le code contient une cartographie de la mémoire qui comporte des espaces disponibles pour une allocation de pages mémoire permanentes ou temporaires. Lorsqu'il démarre, MILO se décompresse à l'emplacement mémoire adéquat. Lorsqu'il transfère le contrôle au noyau Linux, il réserve un espace pour une instance compressée de lui-même, pour le PALcode (indispensable au fonctionnement du noyau) ainsi que quelques structures de données. Cela laisse `la quasi-totalité` de la mémoire centrale libre pour Linux.

L'opération finale du gestionnaire mémoire est de configurer et d'activer l'adressage virtuel afin que les structures de données attendues par Linux soient à leur place en mémoire virtuelle.

MILO contient du code d'initialisation de l'affichage qui prépare le système graphique à l'utilisation de Linux. Il détectera et utilisera un adaptateur VGA s'il est présent, sinon il essaiera d'utiliser le pilote TGA (21030). S'il y a échec de cette initialisation, MILO considèrera qu'il n'y a aucun périphérique graphique sur le système. L'émulation BIOS incluse dans MILO est en fait celle de Digital qui est capable de supporter la plupart, voire la totalité, des cartes graphiques disponibles.

Les pilotes de périphériques de Linux résident dans le noyau dont ils attendent un certain nombre de services. Certains de ces services sont directement fournis par le code du noyau Linux inclus dans MILO. Par exemple la gestion des interruptions est réalisée par un ensemble de fonctions similaires à celles du vrai noyau Linux.

La fonctionnalité la plus puissante de MILO est de permettre l'inclusion de n'importe quel pilote Linux sans apporter de modifications. Cela lui offre la possibilité d'être compatible avec n'importe quel périphérique compatible avec Linux. MILO contient, en standard, tous les pilotes de périphériques en mode bloc du noyau Linux.

MILO charge le noyau Linux depuis un vrai système de fichiers plutôt que depuis un secteur d'amorçage ou d'autres emplacements étranges. Il supporte les systèmes de fichiers MSDOS, EXT2 et ISO9660. Les fichiers GZIPpés sont également reconnus et recommandés, en particulier lors d'un chargement à partir d'une disquette qui reste un support relativement lent. MILO les reconnaît grâce à leur suffixe *.gz*.

Un gestionnaire de clavier rudimentaire est inclus dans MILO si bien qu'avec un pilote de périphérique vidéo d'une simplicité égale il dispose d'une interface utilisateur simple. Cette interface permet de lister les systèmes de fichiers disponibles par le biais de pilotes de périphériques configurés, de démarrer Linux ou des utilitaires de mises à jour de la mémoire flash, de définir des variables d'environnement agissant sur le démarrage du système. Comme avec LILO vous pouvez transmettre des arguments au noyau.

MILO doit renseigner le noyau Linux sur la nature du matériel sous-jacent (type de carte mère, quantité de mémoire RAM totale et quantité libre). Il effectue cela en utilisant les informations contenues dans le HWRPB. Celles-ci sont disposées aux emplacements appropriés en mémoire virtuelle juste avant que le contrôle du système ne soit transféré au noyau Linux.

3 Images précompilées de MILO

Si vous envisagez d'utiliser Linux sur un système Alpha standard, un ensemble d'images précompilées "standard" est à votre disposition. Celles-ci ainsi que leurs sources et bien d'autres choses intéressantes sont disponibles à l'adresse suivante :

`ftp://gatekeeper.dec.com/pub/Digital/Linux-Alpha/Miniloader` .

Le sous-répertoire images contient un répertoire par type de système standard (ex : AlphaPC64) adoptant les conventions de nomenclatures suivantes :

1. `MILO` - Image de MILO, celle-ci peut être démarrée de diverses manières ;
2. `fmu.gz` - Utilitaire de gestion de la mémoire flash ;

3. `MILO.dd` - Image d'une disquette bootable de MILO, pouvant être reproduite grâce à rawrite.exe sous DOS ou `dd` sous Linux.

Le sous-répertoire `test-images` contient un ensemble d'images expérimentales sous la même forme que les précédentes. Bien qu'expérimentales, ces images tendent à contenir les dernières fonctionnalités.

4 Comment compiler MILO ?

La compilation de MILO s'effectue de manière indépendante du noyau. Étant donné qu'il requiert des parties du noyau pour fonctionner, vous devrez, en premier lieu, configurer un noyau qui corresponde au système auquel MILO est destiné. Cela correspond à attribuer le même numéro de version à MILO que celui du noyau utilisé pour le construire. Ainsi MILO-2.0.25.tar.gz sera compilé à l'aide de linux-2.0.25.tar.gz. MILO `peut` être compilé correctement avec une version plus récente du noyau, mais avec celle-ci ce ne sera pas le cas. Étant donné que les librairies dynamiques sont complètement fonctionnelles, il existe deux versions des sources de MILO. Pour effectuer la compilation de MILO dans sa version ELF vous devez premièrement extraire les sources standard puis appliquer un patch à ces dernières, correspondant au numéro de version du patch ELF. Je considérerai, dans la suite de ce document, que les sources et les fichiers objets du noyau sont situés dans le répertoire `/usr/src/linux`, et que le noyau a été correctement compilé à l'aide de la commande `make boot`.

Pour compiler MILO, allez dans le répertoire contenant les sources de MILO et faites appel à la commande `make` de la manière suivante :

```
$ make KSRC=/usr/src/linux config
```

De même que pour la compilation du noyau, le système vous posera un certain nombre de questions.

```
Echo output to the serial port (MINI_SERIAL_ECHO) [y]
```

Il est utile d'utiliser le port série comme redirection de la fonction du noyau printk ; celle-ci est effectuée vers le port `/dev/ttyS0`. Si vous pouvez (et souhaitez) le faire, entrez 'y', sinon 'n'. Toutes les versions précompilées de MILO utilisent le port COM1 comme écho.

```
Use Digital's BIOS emulation code (not free) \
                   (MINI_DIGITAL_BIOS_EMU) [y]
```

Ce code est inclus en tant que bibliothèque de fonctions dont la distribution est gratuite si elle est utilisée sur une machine à base de processeur Alpha. Les sources n'en sont pas disponibles. Si vous répondez `'n'`, l'émulation BIOS équivalente freeware sera compilée. Sachez que vous ne pouvez pas encore choisir le système de Digital utilisant le système ELF (la bibliothèque n'est pas encore prête). Vous devrez donc répondre 'n' à cette question.

```
Build PALcode from sources (Warning this is dangerous) \
                   (MINI_BUILD_PALCODE_FROM_SOURCES) [n]
```

Vous ne devrez utiliser cette option que si vous avez changé les sources du PALcode ; dans tous les autres cas, utilisez la version standard précompilée du PALcode fourni avec MILO.

Tout est désormais prêt, vous pouvez lancer la compilation :

```
$ make KSRC=/usr/src/linux
```

Lorsque la compilation s'est achevée avec succès, l'image de MILO est écrite dans le fichier `milo`. Il y a un grand nombre de fichiers appelés `milo.*`, ceux-ci devront être ignorés.

5 Comment charger MILO ?

La manière la plus courante et la plus simple pour charger MILO est de le faire à partir de la console ARC. Cependant il est possible de réaliser cette opération de diverses manières :

- une disquette bootable dite failsafe ;
- firmware ARC pour Windows NT ;
- Windows NT AlphaBlOS ;
- Console SRM de Digital ;
- un Debug Monitor existant sur les cartes d'évaluations de Digital,
- flash/ROM.

5.1 Chargement de MILO depuis la console ARC pour Windows NT

La plupart, sinon la totalité, des systèmes à base d'Alpha AXP intègrent le firmware ARC pour Windows NT et cela est la méthode recommandée pour démarrer MILO et de surcroît Linux. Une fois que vous disposez de ce firmware et de la version adéquate de MILO, la méthode est complètement générique.

Le firmware ARC pour Windows NT offre un environnement dans lequel les programmes peuvent demander à celui-ci d'effectuer des opérations. Le programme OSLoader de Windows NT réalise exactement cela. Linload.exe est comparable mais beaucoup plus simple, il fait juste ce qui est nécessaire au chargement et à l'exécution de MILO. Il charge le fichier image adéquat en mémoire à l'adresse 0x00000000 puis il exécute les deux instructions swap-PAL puis PALcall à cette adresse. MILO, comme Linux, utilise un PALcode différent de celui utilisé par Windows NT, cela expliquant pourquoi l'instruction swap est nécessaire. MILO se reloge lui-même à l'adresse 0x200000 puis poursuit la réinitialisation du PALcode à cette nouvelle adresse.

Avant d'ajouter des options de démarrage pour Linux, vous devrez copier linload.exe et MILO à un endroit que la console ARC pourra lire. Dans l'exemple suivant on suppose que le démarrage s'effectue à partir d'une disquette au format DOS.

1. Choisissez `"Supplementary menu..."`
2. Au `"Supplementary menu"` choisissez `"Set up the system..."`
3. Au `"Setup menu"` choisissez `"Manage boot selection menu..."`
4. Dans `"Boot selections menu"` choisissez `"Add a boot selection"`
5. Choisissez `"Floppy Disk 0"`
6. Entrez `"linload.exe"` dans la rubrique OSLOADER
7. Répondez "yes" à la question suivante (qui stipule que linload.exe est au même endroit que le système d'exploitation ; pour la console ARC, MILO est vu comme un système d'exploitation à part entière)
8. Entrez '\' ensuite (stipulant que la racine du système est la racine de notre disquette)
9. Entrez le nom de ce choix de démarrage (`Linux` par exemple !)
10. Répondez 'No' à la question 'Initialize debugger at boot time ?'
11. Vous vous retrouvez maintenant dans la section `"Boot selections menu"` : choisissez `"Change a boot selection option"` et sélectionnez le nom que vous avez choisi ci-dessus dans le but de l'éditer.

12. Avec les flèches, sélectionnez "OSLOADFILENAME" puis saisissez le nom de l'image MILO que vous souhaitez utiliser. Par exemple `noname.arc` ou `milo` suivi de Entrée.
13. Retournez à la section "Boot Selections menu" à l'aide de la touche Esc
14. Tapez la touche Esc de nouveau et choisissez "Supplementary menu, and save changes"
15. Retournez au "Boot menu" et vous pouvez alors essayer de démarrer MILO.

Après avoir réalisé cela, vous devriez avoir un 'boot selection' de la forme :

```
LOADIDENTIFIER=Linux
SYSTEMPARTITION=multi(0)disk(0)fdisk(0)
OSLOADER=multi(0)disk(0)fdisk(0)\linload.exe
OSLOADPARTITION=multi(0)disk(0)fdisk(0)
OSLOADFILENAME=\noname.arc
OSLOADOPTIONS=
```

Vous pouvez désormais démarrer MILO (puis Linux). Vous pouvez aussi charger linload.exe et MILO depuis un système de fichiers que Windows NT comprend. Par exemple NTFS ou DOS sur un disque dur.

Le contenu de la variable `OSLOADOPTIONS` est passé à MILO qui l'interprète comme une commande. Donc, pour démarrer Linux sans attente, il faudra fournir une valeur du type :

```
boot sda2:vmlinux.gz root=/dev/sda2
```

Reportez-vous à la section 6 (L'interface utilisateur de MILO) pour de plus amples renseignements sur les commandes disponibles.

Une autre méthode de démarrage de MILO via la console ARC (bien que tortueuse) est d'appeler MILO `fwupdate.exe` puis de choisir l'option 'Upgrade Firmware'.

5.2 Démarrage de MILO depuis l'AlphaBIOS Windows NT

Avec l'apparition des machines de la série XLT, Digital a changé la console ARC pour ses systèmes Windows NT et l'a remplacée par l'AlphaBIOS. Cette nouvelle console offre l'avantage d'une plus grande convivialité. Ce changement d'interface implique un changement de procédure de configuration pour ceux qui souhaitent démarrer Linux pour Alpha dans ce contexte.

La première chose à faire est d'installer la dernière version de l'AlphaBIOS sur votre système. Celle-ci est disponible à l'adresse suivante :

`http://www.windows.digital.com/support/sysoft.htp` .

Téléchargez le fichier ZIP, décompactez-le et installez-le comme suit :

1. Copiez le fichier sur une disquette DOS ;
2. Allumez l'ordinateur et insérez la disquette. Lors de l'affichage des premières informations à l'écran, enfoncez la touche F2 pour entrer dans le setup ;
3. Choisissez "Upgrade AlphaBIOS" ;
4. Suivez les instructions.

Une fois que l'AlphaBIOS est mis à jour, vous pouvez démarrer votre machine comme suit :

1. Créez une disquette DOS contenant les fichiers linload.exe et milo ;
2. Allumez le système et entrez dans le setup ;
3. Choisissez `"Utilities->OS Selection Setup"` ;
4. Appuyez sur INSERT pour ajouter une nouvelle entrée ;
5. Pour `"Boot Name"` entrez un nom de votre choix (ici Linux) puis pressez Tab pour changer de champ ;
6. Avec les flèches, choisissez 'A:' pour la variable `"Boot File is"`, passez au champ suivant ;
7. Entrez "linload.exe". Deux fois TAB ;
8. Entrez "\" pour la variable `"OS Path load file"` ;
9. Pressez ENTREE pour valider.

À ce moment l'AlphaBIOS devrait afficher une boîte de dialogue angoissante indiquant : "Warning: Operating System Selection not valid!". Ne tenez pas compte de cette erreur (cela ne pose de problème qu'à NT), pressez Entrée pour valider.

1. Pressez F10 puis Entrée pour valider ces changements ;
2. Pressez Esc jusqu'à arriver à l'écran d'accueil ;
3. Choisissez, à l'aide des flèches, l'entrée que vous venez de saisir, pressez Entrée pour lancer MILO.

Si la première partition de votre disque dur est un système de fichiers DOS de petite taille destiné au démarrage (ainsi que la procédure d'installation le recommande), lorsque Linux sera installé, vous devrez y copier linload.exe et MILO. Au démarrage suivant, vous devrez configurer votre firmware de telle sorte qu'il aille chercher ces programmes à l'emplacement voulu. Pour ce faire je vous recommande d'utiliser la démarche suivante :

1. Entrez dans le setup (F2 à l'écran de démarrage) ;
2. Choisissez `"Utilities->OS Selection setup"` ;
3. Sélectionnez l'entrée correspondant à Linux, puis pressez F6 pour la modifier ;
4. Placez le curseur sur l'entrée correspondant à la partie périphérique de la ligne `"Boot File"` (`device` pour les versions en Anglais). Avec les flèches, choisissez la partition sur laquelle résident linload.exe et MILO. Appuyez sur Entrée pour valider.
5. Si vous souhaitez que votre système démarre automatiquement après le chargement de MILO positionnez-vous (à l'aide de la touche TAB) sur la variable `"OS Options"` puis spécifiez ici quelle est la ligne de commande à fournir à MILO, par exemple : "boot sda2:vmlinux.gz". Pressez Entrée pour valider ;
6. Utilisez la touche F10 pour sauvegarder les modifications.

Cela fait que l'utilisation de Linux sur une plate-forme utilisant AlphaBIOS devient quasiment identique à celles utilisant la console ARC.

5.3 Démarrage de MILO depuis le Debug Monitor des cartes d'évaluation

Les cartes d'évaluation (et souvent les cartes conçues à partir de leur exemple) proposent un logiciel appelé "debug monitor". Reportez-vous à la documentation de votre système avant d'envisager cette possibilité. Les systèmes suivants *proposent* cette fonctionnalité :

- AlphaPC64 (Section 5.7.2 (AlphaPC64 (Cabriolet)))
- EB64+ (Section 5.7.4 (EB 64+))
- EB66+ (Section 5.7.3 (EB 66+))
- EB164 (Section 5.7.6 (EB 164))
- PC164 (Section 5.7.7 (PC164))

Sachez avant toute chose que, sur certaines anciennes versions, ce logiciel n'inclut pas de gestionnaire écran / clavier. Vous devrez donc vous préparer à connecter un terminal série à votre système. Son interface est très simple et une commande d'aide (help) documente une grande quantité de commandes. Les plus intéressantes de ces commandes incluent les mots `boot` et `load`.

Le debug monitor peut charger une image à partir du réseau (netboot) ou d'une disquette (flboot). Dans tous les cas, l'image doit être chargée à l'adresse 0x200000 (utilisez la commande `bootadr 200000`).

Si l'image se trouve sur une disquette (notez que le seul format de disquette reconnu est DOS) vous devrez utiliser la commande suivante :

```
AlphaPC64> flboot <MILO-image-name>
```

5.4 Chargement de MILO depuis une disquette de démarrage failsafe

D'après les informations *dont je dispose*, seul l'AXPpci33 propose la reconnaissance de secteur d'amorçage de type failsafe floppy (Section 5.7.1 (AXPpci33 (Noname))).

Si vous ne disposez pas d'une image MILO standard précompilée, vous devrez confectionner une disquette au format SRM. Une fois MILO compilé, vous devrez exécuter les instructions suivantes sous Digital Unix :

```
fddisk -fmt /dev/rfd0a
cat mboot bootm > /dev/rfd0a
disklabel -rw rfd0a lrx231 mboot bootm
```

Ou bien les commandes suivantes sous Linux :

```
cat mboot bootm > /dev/fd0
```

Si vous disposez d'une image MILO précompilée vous pourrez construire la disquette de la manière suivante :

```
dd if=MILO.dd of=/dev/fd0
```

5.5 Démarrage de MILO à partir de la mémoire Flash

Il existe certains systèmes qui permettent d'intégrer MILO directement dans la PROM Flash, permettant ainsi le démarrage direct de Linux (sans avoir à utiliser de console du type ARC) :

- AlphaPC64 (Section 5.7.2 (AlphaPC64 (Cabriolet)))
- EB64+ (Section 5.7.4 (EB 64+))
- EB66+ (Section 5.7.3 (EB 66+))
- EB164 (Section 5.7.6 (EB 164))
- PC164 (Section 5.7.7 (PC164))

5.6 Démarrage de MILO par le biais de la console SRM

La console SRM (abréviation de System Reference Manual) ne reconnaît aucun système de fichiers ni même aucune partition disque. Elle s'attend tout simplement à trouver le logiciel d'amorçage à une position physique démarrant à un emplacement donné (il s'agit d'un offset ou position relative). L'information décrivant ce logiciel d'amorçage (sa taille et sa position relative) est décrite dans le premier bloc de 512 octets du disque. Pour charger MILO depuis la SRM vous devez générer cette structure de données en bonne et due forme sur un support que la console peut atteindre. Cela explique l'existence des fichiers `mboot` et `bootm`.

Pour charger MILO depuis un périphérique de démarrage, compilez `mboot` et `bootm` puis écrivez-les sur disque à l'aide de la commande suivante :

```
$ cat mboot bootm > /dev/fd0
```

ou bien téléchargez une image appropriée de MILO à partir d'un site Web, puis utilisez soit `RAWRITE.EXE` soit `dd` pour l'inscrire sur disque.

Cela fait, vous pouvez envisager de démarrer MILO depuis la console SRM, puis d'utiliser une de ses nombreuses commandes pour démarrer. Par exemple, pour démarrer depuis une disquette, vous devrez effectuer l'opération suivante :

```
>>>boot dva0
(boot dva0.0.0.0.1 -flags 0)
block 0 of dva0.0.0.0.1 is a valid boot block
reading 621 blocks from dva0.0.0.0.1
bootstrap code read in
base = 112000, image-start = 0, image-bytes 4da00
initializing HWRPB at 2000
initializing page table at 104000
 initializing machine state
setting affinity to the primary CPU
jumping to bootstrap code
MILO Stub: V1.1
Unzipping MILO into position
Allocating memory for unzip
####...
```

Les systèmes suivants sont compatibles avec la console SRM :

- Noname (Section 5.7.1 (AXPpci33 (Noname)))
- AlphaPC64 (Section 5.7.2 (AlphaPC64 (Cabriolet)))
- EB164 (Section 5.7.6 (EB 164))
- PC164 (Section 5.7.7 (PC164))

5.7 Informations spécifiques à certains systèmes

5.7.1 AXPpci33 (Noname)

La carte Noname est capable de charger MILO depuis une console ARC ou SRM ou depuis une disquette failsafe. Un utilitaire de gestion de la mémoire PROM flash, exécutable depuis MILO permet de copier ce dernier en mémoire flash. En revanche, nous tenons à vous avertir que cette manipulation est très périlleuse car la Noname ne comportant que 256 Ko de mémoire flash, elle ne peut contenir qu'une image en PROM. Si l'image que vous copiez en flash est corrompue, votre système ne démarrera plus.

La méthode de démarrage des cartes Noname est contrôlée par les jumpers J29 et J28. Ils sont disposés comme suit :

```
Numero de broche   4
     J29       2 x x x 6
               1 x x x 5

     J28       2 x x x 6
               1 x x x 5
                   3
```

Les deux options de configuration qui nous intéressent sont sur J28, dont les plots 1-3, qui démarrent la console depuis la flash et J29, dont les plots 1-3 permettent de démarrer la console depuis une disquette. La seconde option est celle dont vous avez besoin pour démarrer MILO la première fois. Une fois que les jumpers auront été configurés pour l'utilisation d'une disquette de démarrage, insérez la disquette contenant MILO en version bootable dans le lecteur puis relancez l'ordinateur. En l'espace de quelques secondes (après l'extinction de la lumière du lecteur), vous devrez constater que l'écran passe du noir au blanc et y lire les informations relatives à l'exécution de MILO. Si les aspects techniques vous intéressent, sachez que la carte Noname charge le contenu de la disquette à l'adresse 0x104000 et les images provenant de la mémoire flash en 0x100000. Pour cette raison, MILO intègre son PALcode à l'adresse 0x200000. Lors de son démarrage, il se reloge lui-même à l'adresse correcte.

5.7.2 AlphaPC64 (Cabriolet)

L'AlphaPC64 est doté, en standard, du Firmware Windows NT (Section 5.1 (Chargement de MILO depuis la console ARC pour Windows NT)), de la console SRM (Section 5.6 (Démarrage de MILO par le biais de la console SRM)) et du Debug Monitor (Section 5.3 (Démarrage de MILO depuis le debug monitor des cartes d'évaluation)). Ces images sont en flash et il reste de la place dans cette mémoire pour ajouter l'image de MILO de manière à pouvoir démarrer MILO directement depuis la PROM. Un utilitaire de gestion de la mémoire flash est disponible sous MILO, ainsi il est possible d'intégrer MILO à la mémoire flash lorsque celui-ci s'exécute (Section 7 (Exécution du gestionnaire de mémoire flash)). Ce procédé accepte l'utilisation de variables d'environnement MILO.

Il est possible de choisir parmi les options de démarrage (ARC, SRM, MILO) en utilisant une combinaison de jumpers et de définir des options de démarrage qui seront sauvegardées dans la NVRAM de l'horloge TOY ("CMOS").

Il s'agit du jumper J2 ; les bits 6 et 7 ont la fonction suivante :

- SP Bit 6 doit toujours être ouvert (pas de jumper) ; dans le cas contraire le mini-debugger sera exécuté.
- SP Bit 7 fermé : Exécuter l'image définie dans la NVRAM
- SP Bit 7 ouvert : Exécuter la première image.

Donc, si le SP Bit 7 est ouvert, le Debug Monitor sera exécuté car il est `toujours` positionné en première place dans la PROM. Et si le SP Bit 7 est fermé, l'image exécutée sera celle définie dans l'horloge système (TOY). L'ARC, le Debug Monitor et MILO acceptent cette option ; il faut, cependant, être très prudent lors de son utilisation. Par exemple, vous ne pouvez pas définir d'option qui vous permettra de démarrer MILO au démarrage suivant : lors de l'utilisation de la console ARC, cette dernière vous permet de passer en mode Debug Monitor ou ARC lors du démarrage, mais elle ne permet pas de passer en mode MILO.

Pour inclure MILO dans la mémoire flash via le Debug Monitor, vous aurez besoin d'une image adéquate (dite flashable). La commande de compilation est : make MILO.rom, mais vous pouvez aussi construire une image rom à l'aide de l'outil makerom du Debug Monitor.

```
> makerom -v -i7 -1200000 MILO -o mini.flash
```

(tapez makerom pour comprendre ce que signifient les paramètres, '7' représente un identificateur d'image flash utilisé par la SROM et -l200000 indique l'adresse de chargement de cette image).

Pour charger cette image en mémoire, utilisez une des commandes flload, netload, ... à l'adresse 0x200000, puis insérez l'image en mémoire flash de la manière suivante :

```
AlphaPC64> flash 200000 8
```

200000 est l'adresse de chargement et 8 est le numéro du segment de mémoire à utiliser. Il y a 16 segments de 64 Ko (soit 512 Ko) dans la SROM. (Le Debug Monitor est au segment 0 et l'ARC au segment 4).

Définissez l'image que la SROM va exécuter au démarrage en donnant une valeur à la variable TOY bootopt :

```
AlphaPC64> bootopt 131
```

(131 indique la 3ème image, 129 la 1ère image, 130 la 2ème, etc...)

Éteignez la machine, mettez le jumper 7 en place et redémarrez la machine. Vous devrez alors pouvoir observer MILO s'exécuter. Félicitations ! Vous pouvez remonter la machine (enfin !). Si ce n'est pas le cas, conservez le tournevis pour plus tard, et enlevez à nouveau le jumper 7 pour redémarrer en mode Debug Monitor, reprenez les étapes les unes à la suite des autres, méthodiquement, en prenant garde de ne rien oublier.

5.7.3 EB 66+

La carte EB66+, comme toutes les cartes d'évaluation de Digital, contient le Debug Monitor, et de ce fait celui-ci va nous permettre de charger MILO. Souvent (mais pas toujours), les cartes dérivées des cartes d'évaluation possèdent ce logiciel. Habituellement, ces cartes contiennent la console ARC. Un utilitaire de gestion de la mémoire flash est disponible sous MILO, ainsi il est possible d'intégrer MILO à la mémoire flash lorsque celui-ci s'exécute (Section 7 (Exécution du gestionnaire de mémoire flash)). Ce système accepte les variables d'environnement MILO.

Ces cartes disposent de plusieurs images en mémoire flash contrôlées par jumper. Les deux bancs de jumpers sont J18 et J16 et se situent au centre bas de la carte (considérant que le processeur Alpha se situe en haut de la carte). Vous pouvez choisir l'option de démarrage par ces jumpers (et MILO lorsqu'il est chargé) ainsi qu'une variable d'environnement sauvegardée dans la mémoire non volatile (NVRAM TOY).

jumper 7-8 Fermé signifie qu'il faut utiliser l'image désignée par la variable bootopt, lorsqu'il est ouvert le Debug Monitor est exécuté.

Pour le reste de la configuration, reportez vous au paragraphe précédent traitant de l'AlphaPC64 (Section 5.7.2 (AlphaPC64 (Cabriolet))).

5.7.4 EB 64+ / Aspen Alpine

Cette carte est très similaire à l'AlphaPC64 excepté qu'elle ne contient pas de mémoire flash utilisable par MILO. Cette carte possède deux ROMS, l'une contenant l'ARC, l'autre contenant le Debug Monitor.

L'Aspen Alpine, quant à elle, ne contient qu'une ROM où est gravée la console ARC.

5.7.5 Universal Desktop Box (Multia)

C'est une station ultra compacte à base d'AXP 21066 qui intègre un sous-système graphique TGA (21030). De plus il n'y a de place que pour une carte graphique PCI demi-hauteur. Elle utilise la console ARC (Windows NT) et il est donc recommandé de l'utiliser pour le démarrage de MILO (Section 5.1 (Chargement de MILO depuis la console ARC pour Windows NT)).

5.7.6 EB 164

La carte EB164, comme toutes les cartes d'évaluation de Digital, contient le Debug Monitor, et de ce fait celui-ci va nous permettre de charger MILO. Souvent (mais pas toujours) les cartes dérivées des cartes d'évaluation possèdent ce logiciel. Habituellement, ces cartes contiennent la console ARC. Un utilitaire de gestion de la mémoire flash est disponible sous MILO, ainsi il est possible d'intégrer MILO à la mémoire flash lorsque celui-ci s'exécute (Section 7 (Exécution du gestionnaire de mémoire flash)). Ce système accepte les variables d'environnement MILO. La console SRM est, de plus, disponible (Section 5.6 (Démarrage de MILO par le biais de la console SRM)).

Ces cartes disposent de plusieurs images en mémoire flash contrôlées par jumper. Le banc de deux jumpers s'appelle J1 et se situe en bas à gauche de la carte (considérant que le processeur Alpha se situe en haut de la carte). Vous pouvez choisir l'option de démarrage par ces jumpers (et MILO lorsqu'il est chargé) ainsi qu'une variable d'environnement sauvegardée dans la mémoire non volatile (NVRAM TOY).

jumper SP-11 de J1 fermé signifie qu'il faut utiliser l'image désignée par la variable bootopt ; lorsqu'il est ouvert le Debug Monitor est exécuté.

Pour le reste de la configuration, reportez-vous au paragraphe précédent traitant de l'AlphaPC64 (Section 5.7.2 (AlphaPC64 (Cabriolet))).

5.7.7 PC164

La carte PC164, comme toutes les cartes d'évaluation de Digital, contient le Debug Monitor, et de ce fait celui-ci va nous permettre de charger MILO. Souvent (mais pas toujours) les cartes dérivées des cartes d'évaluation possèdent ce logiciel. Habituellement, ces cartes contiennent la console ARC. Un utilitaire de gestion de la mémoire flash est disponible sous MILO, ainsi il est possible d'intégrer MILO à la mémoire flash lorsque celui-ci s'exécute (Section 7 (Exécution du gestionnaire de mémoire flash)). Ce système accepte les variables d'environnement MILO. La console SRM est, de plus, disponible (Section 5.6 (Démarrage de MILO par le biais de la console SRM)).

Ces cartes disposent de plusieurs images en mémoire flash contrôlées par jumpers. Le banc principal de jumpers s'appelle J30, il contient les jumpers de configuration. Le jumper CF6 fermé signifie que le système démarrera le Debug Monitor, il est par défaut ouvert.

Pour le reste de la configuration, reportez-vous au paragraphe précédent traitant de l'AlphaPC64(Section 5.7.2 (AlphaPC64 (Cabriolet))).

5.7.8 XL266

Le XL266 est un des systèmes connus sous le nom d'Avanti. Il possède une carte fille sur laquelle résident le processeur Alpha et le cache qui se connecte à la carte mère. Cette carte remplace une carte fille Pentium équivalente.

Certains de ces systèmes sont vendus avec la console SRM, mais certains autres ne sont livrés qu'avec la console ARC (Section 5.1 (Chargement de MILO depuis la console ARC pour Windows NT)).

Voici une liste compatible avec cette série :

- AlphaStation 400 (Avanti),
- AlphaStation 250,
- AlphaStation 200 (Mustang),
- XL. Il en existe deux modèles, XL266 et XL233 qui ne diffèrent que par la vitesse du processeur et la taille de la mémoire cache.

Note : Le système que j'utilise pour développer et tester MILO est un XL266 ; de ce fait, c'est le seul sur lequel je peux garantir un fonctionnement correct. Cela dit les autres systèmes sont, techniquement, équivalents. Ils possèdent les mêmes chipsets et les mêmes mécanismes d'interruptions.

5.7.9 Platform2000

Il s'agit d'un système à base de processeur 21066 à 233 Mhz.

6 L'interface utilisateur de MILO

Après avoir correctement installé MILO (Abrév. de MIniLOader) vous devrez obtenir l'invite de commande de MILO. Il y a une interface de commandes très simple que vous pouvez utiliser dans le but de démarrer une image Linux particulière. Utilisez la commande help pour obtenir une description sommaire des commandes.

6.1 La commande help

Probablement la commande la plus utile de MILO.

```
MILO> help
MILO command summary:
ls [-t fs] [dev:[dir]]
                        - List files in directory on device
boot [-t fs] [dev:file] [boot string]
                        - Boot Linux from the specified device and file
run [-t fs] dev:file
                        - Run the standalone program dev:file
show                    - Display all known devices and file systems
set VAR VALUE           - Set the variable VAR to the specified VALUE
unset VAR               - Delete the specified variable
reset                   - Delete all variables
print                   - Display current variable settings
help [var]              - Print this help text
```

```
Devices are specified as: fd0, hda1, hda2, sda1...
Use the '-t filesystem-name' option if you want to use
anything but the default filesystem ('ext2').
Use the 'show' command to show known devices and filesystems.
Type 'help var' for a list of variables.
```

Note : la commande `bootopt` n'apparaît que pour les systèmes AlphaPC64 (et équivalents).

Périphériques. Jusqu'à ce que vous utilisiez une commande qui nécessite l'utilisation d'un des périphériques, aucune initialisation n'est réalisée sur ces derniers. La première commande `show`, `ls`, `boot` ou `run` provoquera l'initialisation des périphériques. Les pilotes de périphériques portent exactement les mêmes noms que ceux de Linux. Donc le premier disque IDE sera appelé hda et la première partition sera hda1. Utilisez la commande `show` pour obtenir une liste des périphériques disponibles.

Systèmes de fichiers. MILO est compatible avec trois différents types de systèmes de fichiers : MSDOS, EXT2 et ISO9660. À partir du moment oû un périphérique est disponible pour lui, MILO est à même de charger et d'exécuter n'importe quelle image disponible. Le système de fichiers par défaut est `EXT2`. Toutes les commandes utilisant les systèmes de fichiers permettent d'en préciser le type par le biais de l'option -t filesystem. Donc si vous souhaitez obtenir une liste du contenu d'un CDROM il suffit d'entrer la commande :

```
MILO> ls -t iso9660 scd0:
```

Variables. MILO contient quelques variables qui aident au processus de démarrage. Si vous démarrez à partir de la console ARC, MILO utilisera les variables d'environnement définies par ce firmware. Pour certains systèmes (tels que l'AlphaPC64), MILO dispose de son propre jeu de variables qui ne changent pas d'un démarrage à l'autre. Ces variables sont :

```
MILO> help var
Variables that MILO cares about:
  MEMORY_SIZE          - System memory size in megabytes
  BOOT_DEV             - Specifies the default boot device
  BOOT_FILE            - Specifies the default boot file
  BOOT_STRING          - Specifies the boot string to pass to the kernel
  SCSIn_HOSTID         - Specifies the host id of the n-th SCSI controller.
  PCI_LATENCY          - Specifies the PCI master device latency
  AUTOBOOT             - If set, MILO attempts to boot on powerup
                         and enters command loop only on failure.
  AUTOBOOT_TIMEOUT     - Seconds to wait before auto-booting on powerup.
```

ATTENTION à l'utilisation de la variable AUTOBOOT sans définir de valeur délai (AUTOBOOT_TIMEOUT) car MILO démarrera après avoir attendu 0 seconde. Ce n'est peut-être pas ce que vous souhaitez.

PCI_LATENCY représente le nombre de cycles PCI qu'un périphérique fonctionnant en bus master se réserve lorsqu'il acquiert le contrôle du bus. La valeur par défaut est de 32 et la maximum de 255. Définir une valeur élevée revient à dire que chaque fois qu'un périphérique prend le contrôle du bus PCI, il transfèrera plus de données. Cependant cela signifie aussi qu'un périphérique devra attendre plus longtemps pour obtenir le contrôle du bus.

6.2 Démarrage de Linux

La commande `boot` charge et démarre un noyau Linux à partir d'un périphérique. Vous aurez besoin d'avoir un disque contenant ce noyau (SCSI, IDE, disquette dans un format reconnu par MILO). Ces images peuvent être compressées à l'aide de gzip ; les premières versions de MILO reconnaissaient ces fichiers à l'aide de leur extension '.gz', tandis que les plus récentes font appel à la signature du fichier.

Il est important de retenir que la version de MILO ne nécessite pas de correspondance avec celle du noyau Linux que vous souhaitez charger. Vous démarrez Linux avec la commande suivante :

```
MILO> boot [-t file-system] device-name:file-name \
          [[boot-option] [boot-option]...]
```

`device-name` représente le nom du périphérique que vous souhaitez utiliser, `file-name` le nom de fichier de l'image à charger en mémoire. Tous les arguments spécifiés par la suite sont directement transmis au programme exécuté (ici le noyau Linux).

Si vous installez Linux Red Hat pour Alpha vous devrez spécifier un périphérique racine (root device) et autres options :

```
MILO> boot fd0:vmlinux.gz root=/dev/fd0 load_ramdisk=1
```

MILO contient automatiquement les périphériques en mode bloc configurés dans vmlinux. J'ai testé le pilote de disquette, de disque IDE ainsi qu'un certain nombre de pilotes SCSI (par exemple le NCR810), et ceux-ci fonctionnent bien. Il est de même important d'attribuer un ID SCSI raisonnable à votre contrôleur SCSI. Par défaut MILO l'initialisera à la valeur maximale (7) qui, en principe, devrait fonctionner correctement. Néanmoins, si vous le souhaitez, vous pouvez spécifier l'ID SCSI de votre N-ième contrôleur SCSI en définissant la variable `SCSIn_HOSTID` ; par exemple pour préciser que l'ID SCSI du contrôleur numéro 0 est 7 utilisez la commande suivante :

```
setenv SCSI0_HOSTID 7
```

6.3 Redémarrage de Linux

Vous pouvez vouloir redémarrer Linux à l'aide de la commande `shutdown -r now`. Dans ce cas, le noyau Linux repasse le contrôle à MILO (par une instruction HALT CallPAL entrypoint). MILO laisse une version de lui-même compressée en mémoire pour cette seule raison et détecte que le système est redémarré par le biais du HWRPB (Hardware Restart Parameter Block). Dans ce cas vous pouvez edémarrer à l'aide de la même commande qui vous a servi au démarrage précédent. Il y a un délai de 30 secondes qui vous permet d'interrompre ce processus et de démarrer n'importe quel autre noyau de n'importe quelle autre manière.

6.4 La commande "bootopt"

Pour les systèmes utilisant la mémoire flash comme les AlphaPC64, EB164 et EB66+, il existe différentes options de démarrage. Celles-ci sont modifiables à l'aide de la commande `bootopt`. Cette commande a un unique argument : il s'agit d'un nombre décimal qui représente le type d'image à exécuter au prochain redémarrage de la machine (Allumage / Extinction ou Reset).

La valeur **0** active le Debug Monitor, **1** active la console ARC.

Ce chiffre correspond en fait à la N-ième image présente dans la PROM. Afin d'indiquer que l'action à effectuer est le démarrage, il faut ajouter 128 à ce chiffre pour obtenir la valeur à transmettre à `bootopt`. Vous aurez donc la commande suivante (sachant que MILO est la 3ème image en PROM) :

```
MILO> bootopt 131
```

`Note :` Soyez très vigilant avec cette commande, une bonne règle est de ne jamais utiliser la valeur 0 (Debug Monitor), mais utilisez plutôt le système des jumpers pour obtenir ce résultat.

7 Exécution du gestionnaire de mémoire flash

La commande `run` est utilisée pour exécuter le gestionnaire de mémoire flash. Avant de démarrer, vous devrez disposer d'un périphérique accessible par MILO contenant le programme updateflash. Celui-ci (comme vmlinux) peut être compressé avec gzip. Vous devez exécuter ce programme à l'aide de la commande suivante :

```
MILO> run fd0:fmu.gz
```

Une fois chargé et initialisé, le gestionnaire de mémoire flash vous donnera quelques informations concernant le périphérique flash, et vous proposera une invite de commandes. À nouveau la commande `help` est salvatrice.

```
Linux MILO Flash Management Utility V1.0
Flash device is an Intel 28f008SA
16 segments, each of 0x10000 (65536) bytes
Scanning Flash blocks for usage
Block 12 contains the environment variables
FMU>
```

`Notez` que sur les systèmes qui permettent la sauvegarde de variables d'environnement et qui disposent de plus d'un bloc de mémoire flash (par exemple l'AlphaPC64), le gestionnaire de mémoire flash cherchera un bloc disponible pour la sauvegarde des variables d'environnement de MILO. Si un tel bloc existe, le gestionnaire de mémoire flash vous indiquera son emplacement. Dans le cas contraire, vous devrez utiliser la commande `environment` pour définir un bloc et l'initialiser. Dans l'exemple ci-dessus le bloc numéro 12 de la mémoire flash renferme les variables d'environnement de MILO.

7.1 La commande "help"

```
FMU> help
FMU command summary:

list            - List the contents of flash
program         - program an image into flash
quit            - Quit
environment     - Set which block should contain the environment variables
bootopt num     - Select firmware type to use on next power up
help            - Print this help text
FMU>
```

Note : les commandes `environment` et `bootopt` ne sont disponibles que sur les systèmes EB66+, EB164, PC164 et AlphaPC64 (ainsi que leur clones).

7.2 La commande "list"

La commande "list" affiche les informations sur l'utilisation de la mémoire flash. Pour les systèmes disposant de plus d'un bloc, l'usage de chaque bloc est affiché.

```
FMU> list
Flash blocks:  0:DBM  1:DBM  2:DBM  3:WNT  4:WNT  5:WNT  6:WNT  7:WNT  8:MILO
        9:MILO 10:MILO 11:MILO 12:MILO 13:U 14:U 15:WNT
```

```
Listing flash Images
  Flash image starting at block 0:
    Firmware Id: 0 (Alpha Evaluation Board Debug Monitor)
    Image size is 191248 bytes (3 blocks)
    Executing at 0x300000
  Flash image starting at block 3:
    Firmware Id: 1 (Windows NT ARC)
    Image size is 277664 bytes (5 blocks)
    Executing at 0x300000
  Flash image starting at block 8:
    Firmware Id: 7 (MILO/Linux)
    Image size is 217896 bytes (4 blocks)
    Executing at 0x200000
FMU>
```

7.3 La commande "program"

Le gestionnaire de mémoire flash contient une copie compressée d'une image flash de MILO. Cette commande vous permet de sauvegarder cette image dans la PROM. La commande vous permet de revenir en arrière, mais avant de l'utiliser vous devriez utiliser la commande list pour définir où il faut mettre MILO. Si MILO est déjà dans la mémoire flash, le gestionnaire de mémoire flash vous demandera si vous souhaitez remplacer la version actuellement en PROM.

```
FMU> program
Image is:
    Firmware Id: 7 (MILO/Linux)
    Image size is 217896 bytes (4 blocks)
    Executing at 0x200000
Found existing image at block 8
Overwrite existing image? (N/y)? y
Do you really want to do this (y/N)? y
Deleting blocks ready to program: 8 9 10 11
Programming image into flash
Scanning Flash blocks for usage
FMU>
```

!!! IMPORTANT !!! Attendez que cela soit fini avant d'éteindre votre ordinateur.

!!! IMPORTANT !!! Je n'insisterai jamais assez sur le fait que vous devez être extrêmement prudent lorsque vous effectuez ce genre de manipulation. Une très bonne règle à respecter est de ne jamais effacer le Debug Monitor.

7.4 La commande "environment"

Cette commande détermine quel est le bloc qui recevra les variables d'environnement.

7.5 La commande "bootopt"

Il s'agit de la même commande que celle de MILO (Section 6.4 (La commande "bootopt")).

7.6 La commande ”quit”

Cette commande n’a que peu de sens, car la seule manière possible pour se retrouver sous MILO est de redémarrer le système.

8 Restrictions

Malheureusement la perfection n’étant pas de ce monde, il existe des restrictions auxquelles il faut se plier.

MILO n’est pas conçu pour charger d’autres systèmes d’exploitation que Linux, cependant il peut charger et exécuter des images dont l’exécution est possible au même emplacement mémoire que Linux (c’est à dire : 0xFFFFFC0000310000). C’est ce qui permet au gestionnaire de mémoire flash de fonctionner.

Les sources du PALcode contenues dans `miniboot/palcode/`*toto* sont correctes, mais ce PALcode est problématique lorsqu’il est compilé avec la dernière version de `gas`. Problème qui n’existe pas avec l’ancien exécutable gas fourni avec les cartes d’évaluation. J’essaie actuellement de trouver quelqu’un capable de résoudre ce problème. Pour l’instant, j’ai fourni un PALcode précompilé pour les cartes supportant MILO et David Mosberger-Tang a une version de gas corrigée sur son site ftp.

9 Dépannage

Voici un ensemble de problèmes courants que certaines personnes ont rencontrés ; vous trouverez, ensuite, leur solution.

Lecture de disques DOS depuis le Debug Monitor.

Quelques versions du Debug Monitor (pré-version 2.0) ont un problème avec le gestionnaire DOS généré à partir de Linux. Ce problème se manifeste de la manière suivante : le Debug Monitor semble lire correctement les premiers secteurs de la disquette, puis affiche, en boucle, le message "Bad Sectors". Cela vient apparemment du fait qu’il existe une différence entre le système de fichiers DOS tel que l’attend le Debug Monitor et l’implémentation de DOSFS sous Linux. Afin de résoudre ce problème, utilisez un PC sous DOS pour confectionner la disquette (et non pas Linux). Par exemple, si le chargement du fichier `MILO.cab` ne fonctionne pas, effectuez les opérations suivantes sur une machine DOS :

```
copy a:MILO.cab c:
copy c:MILO.cab a:
del c:MILO.cab
```

Et réessayez de démarrer depuis cette disquette. Cela devrait résoudre le problème.

Milo affiche une série de `0>`et n’accepte pas de commandes. Ceci arrive quand MILO a été compilé pour utiliser COM1 comme console secondaire. Dans ce cas, MILO reproduit l’affichage sur COM1 et accepte des saisies depuis ce dernier. C’est excellent pour déboguer, mais pas tant que ça si vous avez un périphérique autre qu’un terminal série connecté. Si cela arrive, déconnectez le périphérique attaché au port COM1 ou mettez-le hors tension jusqu’à ce que le noyau Linux ait démarré. À ce stade tout devrait fonctionner correctement.

MILO se plaint de ce que l’image du noyau possède un mauvais ’magic number’.

Les anciennes versions de MILO ne sont pas compatibles avec les fichiers objets de type ELF. Cela pourrait être la source de vos ennuis. Si c’était le cas, passez à la version la plus récente que vous pourrez trouver. Toutes les versions à partir de la 2.0.20 sont compatibles avec les objets ELF. D’autre part il se pourrait que l’image de MILO soit corrompue, ou qu’il faille ajouter l’extension ’.gz’ au nom du fichier noyau.

MILO affiche "...turning on virtual addressing and jumping to the Linux Kernel" puis rien ne se passe.

Une des causes possibles est que l'image du noyau est configurée de manière incorrecte ou qu'elle a été produite pour un autre type de machine Alpha. Une autre possibilité est que la carte vidéo est une TGA (ZLXp) et que le noyau est configuré pour utiliser le VGA (ou l'inverse). Il est utile de compiler le noyau pour qu'il affiche sa console sur COM1 (echo console) et donc de pouvoir, dans ce cas, connecter un terminal série ou alors d'essayer le noyau qui était livré avec votre distribution Linux.

MILO ne reconnaît pas les périphériques SCSI.

L'image standard de MILO comporte autant de pilotes de périphériques qu'il en existe de stables pour Alpha (au moment de la rédaction de ce manuel il contient les pilotes pour : NCR810, QLOGIC ISP, Buslogic et Adaptec 294x/394x). Si votre carte n'est pas supportée c'est peut-être que le pilote n'est pas encore suffisamment stable sur plate-forme Alpha. Vous pouvez obtenir la liste des périphériques SCSI supportés par une image MILO avec la commande `show`.

10 Remerciements

Je souhaiterais remercier :

- Eric Rasmussen et Eilleen Samberg, les auteurs du PALcode ;
- Jim Paradis pour le pilote clavier, l'interface originelle de MILO et son travail sur AlphaBIOS ;
- Jay Estabrook pour son aide et ses débogages ;
- David Mosberger-Tang pour la version freeware de l'émulateur BIOS et son appui et ses encouragements ;
- Le dernier, et pas des moindres, Linus Torvalds pour le code du timer et celui du noyau.

Beaucoup de choses restent à faire sous MILO ; s'il y a quelque chose que vous souhaitez faire ou ajouter vous-même, faites-le moi savoir :

david.rusling@reo.mts.dec.com `mailto:david.rusling@reo.mts.dec.com` , afin de ne pas développer deux fois la même chose.

Pour finir, un grand merci à Digital pour le développement d'un si merveilleux processeur (et pour le salaire qu'ils me versent afin de le faire).