



Installation Manual
IM-DOC-021

Using the DiskOnChip[®]
with Linux OS

Written by: Ron Dick, Esther Spanjer & Vadim Khmel'nitsky

NOV-2000

91-SR-005-10-7L REV. 3.2

Limited Warranty

(a) M-Systems warrants that the Licensed Software — **prior to modification and adaptation by Licensee** — will conform to the documentation provided by M-Systems. M-Systems does **not** warrant that the Licensed Software will meet the needs of the Licensee or of any particular customer of Licensee, nor does it make any representations whatsoever about Licensed Software that has been modified or adapted by Licensee.

(b) Subsection (a) above sets forth Licensee's sole and exclusive remedies with regard to the Licensed Software.

M-SYSTEMS MAKES NO OTHER WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THE LICENSED SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. THERE ARE NO OTHER WARRANTIES WITH RESPECT TO THE LICENSED SOFTWARE ARISING FROM ANY COURSE OF DEALING, USAGE, OR TRADE OR OTHERWISE.

IN NO EVENT SHALL M-SYSTEMS BE LIABLE TO LICENSEE FOR LOST PROFITS OR OTHER INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES, WHETHER UNDER THIS AGREEMENT, IN TORT OR OTHERWISE.

(c) Licensee shall not make any promise, representation, warranty or guaranty on behalf of M-Systems with respect to the Licensed Software except as expressly set forth herein.

Note: The Licensed Software is not warranted to operate without failure. Accordingly, in any use of the Licensed Software in life support systems or other applications where failure could cause injury or loss of life, the Licensed Software should only be incorporated in systems designed with appropriate and sufficient redundancy or back-up features.

Contents

1	Introduction	4
1.1	Hardware Requirements for the DiskOnChip.....	4
1.2	Requirements for Installation of the DiskOnChip into Linux.....	5
1.3	Utilities Diskette Content	6
1.4	Linux Driver Files for the DiskOnChip	6
2	Installing the DiskOnChip as an Additional Drive	7
2.1	Updating the Firmware.....	8
2.2	Integrating the TrueFFS Driver into Linux.....	8
2.2.1	Compiling as Part of the Linux Kernel ("Built-in")	8
2.2.2	Initializing the DiskOnChip.....	11
2.2.3	Compiling as a Loadable Kernel Module ("Loadable-module").....	13
2.2.4	Creating and Populating the DiskOnChip Module Directory	13
2.2.5	Building the DiskOnChip Module (doc.o)	14
2.2.6	Installing the DiskOnChip Board	14
3	Creating a Linux Partition on the DiskOnChip	15
3.1	Creating a Native Linux File System on the DiskOnChip.....	17
4	Bootng Linux from the DiskOnChip.....	18
4.1	Creating a Root File System	18
4.1.1	Making the DiskOnChip Accessible	19
4.1.2	INITRD Building Process.....	19
4.1.3	Creating a Personal File System	21
5	Troubleshooting	26
5.1	DiskOnChip does not Boot Linux.....	26
5.2	The Kernel Boots, but will not Proceed.....	26
5.3	Cannot Log In	26
5.4	Kernel Does not Compile Correctly or Patch Utility is Not Available	26
6	Bootng From a HDD When the DiskOnChip Firmware is Active	27
	Additional Information and Tools.....	28
	How to Contact Us.....	29

1 Introduction

M-Systems' DiskOnChip[®] is a family of flash disks. It contains built-in firmware that provides full hard disk emulation and allows the DiskOnChip to operate as a boot device.

When used under Linux, the DiskOnChip is managed by a device driver, based on M-Systems' TrueFFS[®] (True Flash File System) technology. The TrueFFS driver is attached to the standard Linux file system [ext2].

This Installation is intended for system integrators designing with the DiskOnChip 2000 or DiskOnChip Millennium. It describes how the DiskOnChip can be installed as an additional disk or as a boot device under Linux.

It is assumed that the reader is familiar with the Operating System in use.

While the hardware requirements of the DiskOnChip will be briefly discussed, the main part of this installation manual is related to software installation. This will include basic driver installation and boot issues.

1.1 Hardware Requirements for the DiskOnChip

Originally designed for PC environments, the DiskOnChip can also be used in different hardware environments. The minimum requirements are a 12-bit address bus, 8-bit data bus, and three active low control signals (CE#, OE#, WE#). Figure 1 illustrates the DiskOnChip and its pins. For more detailed information of the DiskOnChip hardware environment, refer to the various DiskOnChip Datasheets or to the following Application Notes:

- AP-DiskOnChip-10 "Designing with the DiskOnChip"
- AP-DiskOnChip-30 "Designing with the DiskOnChip Millennium in a RISC Environment"
- AP-DiskOnChip-31 "Designing with the DiskOnChip Millennium in a PC Environment"

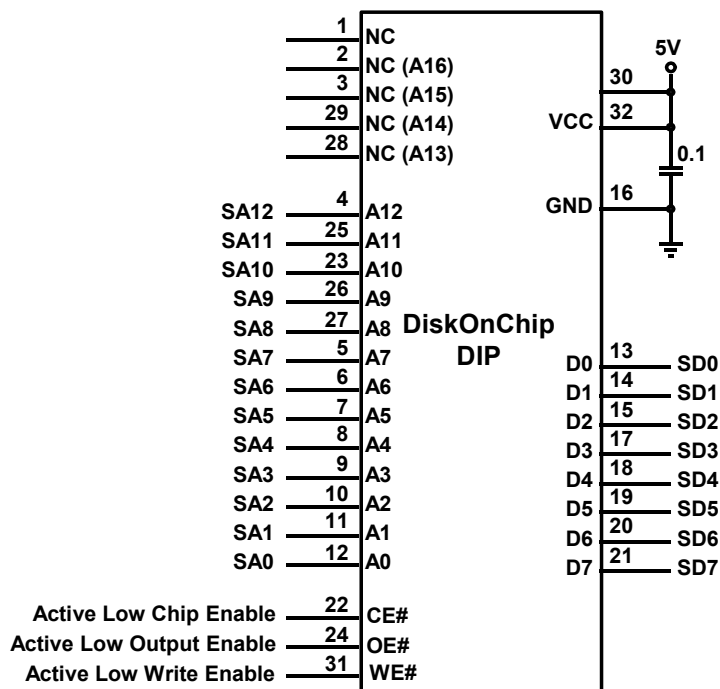


Figure 1 DiskOnChip Pins

1.2 Requirements for Installation of the DiskOnChip into Linux

In order to prepare the DiskOnChip to boot Linux, the following software programs and tools are required:

- Linux must be installed on your HDD. It is possible to check the kernel version by typing the following command:
uname -r.
- The Linux kernel sources package must be installed in Linux. Prior to installing Linux, ensure that it is possible to pass a full compilation of the required sources.
(To obtain Linux kernel sources, refer to kernel-HOWTO at <http://sunsite.unc.edu/LDP/>).
- A DOS boot diskette or a HDD that boots DOS
- M-Systems' DiskOnChip DOS utilities diskette
- M-Systems' TrueFFS driver for Linux.

Note: The DiskOnChip TrueFFS driver for Linux supports kernel versions 2.0x, 2.2x and 2.4x.

Note: The latest DiskOnChip DOS utilities can be downloaded from M-System' website: www.m-sys.com.

1.3 Utilities Diskette Content

The following files are required from the DiskOnChip DOS utilities diskette:

Dformat	DiskOnChip formatting utility
Dupdate	Utility for updating DiskOnChip firmware
Docpmap	Utility to retrieve information about the DiskOnChip
Doc42.exb	DiskOnChip firmware image, “42” is the firmware version etc.
Doc2.fff	Alternative firmware image for the DiskOnChip

Note: The file `doc2.fff` can be found in the `\AP012` directory when using DiskOnChip DOS utilities version 1.21 or higher.

1.4 Linux Driver Files for the DiskOnChip

The compressed file `driver_4.2.1.tgz` contains the following files:

<code>/driver/Create_standalone</code>	Script to create standalone loadable module directory
<code>/driver/Patch_linux</code>	Script to patch Linux kernel, adding built-in driver
<code>/driver/mknod-fl</code>	Script to create the <code>/dev</code> entries
<code>/driver/linux-2_2-patch</code>	Patch for 2.2.X kernel
<code>/driver/linux-2_0_38-patch</code>	Patch for 2.0.38 kernel
<code>/driver/linux-2_4_0_test8-patch</code>	Patch for 2.4.0 kernel
<code>/driver/driver-patch</code>	Patch to create <code>drivers/block/doc</code> directory
<code>/driver/README.kit</code>	Readme file
<code>/lilo/README.lilo</code>	Explanations regarding the lilo directory usage
<code>/lilo/doc-lilo-0.21-18.tgz</code>	TGZ file that contains source code of doc-lilo
<code>/lilo/doc-lilo-0.21-18.i386.rpm</code>	RPM file that contains Binary for the patched LILO based on LILO version 0.21-18

/lilo/doc-lilo-0.21-18.src.rpm	RPM file containing source code of the doc-lilo based on LILO version 0.21-18
/lilo/boot/doc.b	Boot loader, updated to use with DiskOnChip
/lilo/sbin/doc-lilo	Linux loader patched to use with the DiskOnChip
/INITRD/Insmod	Statically linked insmod utility
/INITRD/modutils-2.2.2-pre6.tar.gz	Source code of insmod and other utilities that work with modules
/INITRD/linuxrc	Executable that runs when INITRD is loaded
/INITRD/linuxrc.c	Source code of the linuxrc

2 Installing the DiskOnChip as an Additional Drive

M-Systems' DiskOnChip may be used either as a supplemental (non-boot) disk or as a bootable disk. The DiskOnChip TrueFFS Linux driver may be compiled either as a General Public License (GPL) compliant loadable kernel module ("loadable-module"), or as part of the Linux kernel ("Built-In").

This means that two independent decisions have to be made when working with DiskOnChip:

1. Driver installation method.
 - Compiled as part of the Linux kernel ("built-in")
 - Compiled as a loadable kernel module ("loadable-module" - GPL Compliant)
2. Is the DiskOnChip used as a bootable disk?

Typically, if it is required to use the DiskOnChip for a root file system, compiling the driver as part of the kernel is uncomplicated and requires less additional configuration. However, due to the GPL restrictions, a kernel with the M-System driver built into it cannot be distributed (all source code for the kernel must be provided and the M-Systems software contains some pre-compiled software).

If it is required to distribute a working solution, compile the driver as a loadable module. When booting from the DiskOnChip, use the kernel INITRD facilities to load the driver module during the boot process INITRD (refer to Chapter 3).

Note: If it is required to work in /usr/src/linux, ensure to back up both the kernel source tree in /usr/src/linux and the modules in /lib/modules/.

The file /usr/src/linux/README contains basic outline of the kernel build process.

Some of the steps below produce large amounts of output. The easiest way to save this information for later read-through is to use the "script" command.

For example:

```
script output.file
command_that_prompts_for_input_and/or_produces_lots_of_output
exit
```

This starts a subshell and copies all input and output to the "output.file". Unfortunately, screen oriented programs like "vi" and "make menuconfig" place trash in the script file (even the terminal escape sequences are caught), but it is still useable. See "man script" for details.

2.1 Updating the Firmware

Before the DiskOnChip can be used as the boot disk or as an additional disk in Linux, it must first be formatted with the alternative firmware image.

To Update Firmware:

1. Insert the DiskOnChip into its socket and boot DOS from the HDD or Floppy.
2. Format the DiskOnChip with the alternative firmware:

```
dformat /win:d000 /s:doc2.fff /y
```

Note: If the error message "No DiskOnChip 2000 (R) was found at D000:0" is displayed, run the DOS command docpmap /i to locate the DiskOnChip address.

2.2 Integrating the TrueFFS Driver into Linux

To Prepare Linux for Driver Integration:

- Unpack the driver using the following command:

```
tar xvfz driver_4.2.1.tgz
```

Note: Unpacking the driver, requires being logged in as the superuser (root).

2.2.1 Compiling as Part of the Linux Kernel ("Built-in")

Compiling as part of the Linux kernel makes use of the "patch_linux" script. This script takes three arguments:

```
/patch_linux kernel_patch driver_patch [kernel_src_dir]
```

where "kernel_patch" is the kernel version dependent name of a patch file (linux-2_2-patch or linux-2_0_38-patch or linux-2_4_0_test 8-patch), and "driver_patch" is the name of the driver patch file (always driver-patch), and "kernel_src_dir" is the top directory of the kernel source tree you want patched. The "kernel_source_dir" defaults to the usual directory, /usr/src/linux. It is also possible to work using a copy of the kernel source tree for example, /usr/src/linux-doc).

For the remainder of this description, it is assumed that:

- a) The tarball in /usr/src/Linux-DOC-4.2.1 is unpacked
- b) A kernel source tree in /usr/src/linux-2.2.5-doc is used
- c) The kernel is version 2.2.x (for example, 2.2.5-15)
- d) LILO is used as the boot loader.

1. Apply the following patches:

```
./patch_linux linux-2_2-patch driver_patch /usr/src/linux-2.2.5-15
```

2. Create the device nodes for the DiskOnChip device (/dev/fl*):

```
./mknod-fl
```

3. Build the new kernel:

- Change into the directory with the patched kernel source tree, and ensure that the "extra-version" is "-doc":

```
cd /usr/src/linux-2.2.5-doc
```

- Edit the Makefile to set the value of EXTRAVERSION on line 4:

```
EXTRAVERSION = -doc
```

4. Turn ON the DiskOnChip drivers in the configuration:

```
make menuconfig
```

5. From the block devices area, turn ON the M-Systems driver.

Note: Take care when exiting. When prompted to press the esc key twice to exit <esc> <esc>, the config changes may be canceled. Proceed slowly, to respond to the dialog options regarding saving changes.

6. From the Block Drivers Menu, the hot key for the DiskOnChip block drivers is "S":

```
B)lock Devices,
S) M-(S)ystems drivers,
Y)es - build the drivers in,
esc,  -- return to main menu
esc,  -- exit
y      -- yes, save the changes)
```

7. Build the dependencies:

```
script mkdep.out
make dep
```

```
exit
```

Examine the script output file "mkdep.out" for any errors.

8. Build the compressed kernel:

```
script mkbz.out
make bzImage
exit
```

Examine the script output file "mkbz.out" for any errors.

9. Build any configured loadable modules (typically, several are present, despite the fact that the DiskOnChip driver is being built-in):

```
script mkmod.out
make modules
exit
```

Examine the script output file "mkmod.out" for any errors.

10. Back up the module tree and then install the loadable modules

If you've set the "EXTRAVERSION" value as above, this places the modules in /lib/modules/2.2.5-doc. script mkinstmod.out

```
make modules_install
exit
```

Examine the script output file "mkinstmod.out" for any errors.

- 11 Create a new INITRD image with the required modules, and then install it in the /boot directory. (This is required when working from SCSI disks. In addition, this may be required, when using modules during the early stages of the boot).

```
mkINITRD INITRD-2.2.5-doc.img 2.2.5-doc
cp INITRD-2.2.5-doc.img /boot
```

INITRDINITRD

12. Edit /etc/lilo.conf to add a description of the new kernel and its resources. There must be a section in /etc/lilo.conf as it is distributed:

```
image=/boot/vmlinuz-2.2.5-15
    label=linux
    root=/dev/sda1
    INITRD=/boot/INITRD-2.2.5-15.img
    read-only
```

Add a similar section at the end of the file:

```
image=/boot/vmlinuz-2.2.5-doc
    label=doc
    root=/dev/sda1
    INITRD=/boot/INITRD-2.2.5-doc.img
    read-only
```

See "man 5 lilo.conf" for more details.

Install the new kernel (ensure to back up the old kernel first)

13. Install the new kernel as follows:

```
script mkinst.out
make install
exit
```

This copies the new kernel into/boot and runs lilo to install the new boot map and INITRD. The following is displayed near the end of the output:
"Adding doc".

2.2.2 Initializing the DiskOnChip

After installing the DiskOnChip as an additional drive, and assuming the DiskOnChip board is physically installed – it is necessary to initialize the DiskOnChip device and then boot the new kernel from the HDD. Once completed, the DiskOnChip is recognized as an additional drive.

To Initializing the DiskOnChip device:

1. Boot with a DOS floppy containing the DiskOnChip utilities, and ensure that the DiskOnChip is not set up as the boot device (since it is empty):
`dformat /win:d000 /S:doc2.fff`
2. Verifying that the DiskOnChip is operational as an additional drive:
Run Lilo (Linux Loader) to create the map for the kernel and ensure that "doc" is listed (if not, return to the beginning of Section 2.2.1):
`# lilo`
3. Load the updated kernel with the TrueFFS as follows:
 - Reboot the computer and load Linux.
 - When the LILO prompt is displayed, press <Ctrl> or <Alt> or <Tab>. The screen displays the following message:
Lilo boot:

4. Load the recompiled kernel:

```
Lilo boot:doc
```

5. Look in /var/log/messages for messages from the new kernel and the DiskOnChip driver that are displayed as follows:

```
Linux version 2.2.5-doc.....  
Flash disk driver for DiskOnChip2000  
Copyright (C) 1998,2000 M-Systems Flash Disk Pioneers Ltd.  
Copyright (C) 2000 Lineo  
DiskOnChip device(s) found: 1  
Fat Filter Enabled  
fl_geninit: registered device at major: 100  
partition: 0: start_sect: 0, nr_sects: f98 Fl_blk_size[]: 7cckb  
partition: 1: start_sect: 0, nr_sects: 0 Fl_blk_size[]: 0kb  
Partition check:  
fla: fla1
```

6. Verify that the DiskOnChip is operational:

```
cd /mnt  
mkdir doc  
mount -t vfat /dev/msys/fla1 /mnt/doc  
put stuff into the DiskOnChip (/mnt/doc/*)
```

7. When the DiskOnChip is operational, add an entry to /etc/fstab to be mounted it at boot time.

Remember: Due to GPL restrictions the built-in Linux driver provides a non-redistributable kernel. To redistribute this kernel the GPL requires the entire source on request, which is unavailable for the DiskOnChip driver. The solution, to a redistributable kernel that operates with a DiskOnChip, is to set up the driver as a loadable module.

2.2.3 Compiling as a Loadable Kernel Module ("Loadable-module")

Compiling as a loadable kernel module makes use of the "create_standalone" script:

```
./create_standalone driver_patch
```

where "driver-patch" is the name of the driver patch file (always driver-patch). This creates a directory "doc-module" under the current directory. This approach makes no changes to an existing source tree (except that it adds the directory "doc-module"), so it is reasonable to apply this to an existing kernel source tree.

For the rest of this description it is assumed that:

- a) The tarball in /usr/src/Linux-DOC-4.2.1 is unpacked
- b) A kernel source tree in /usr/src/linux-2.2.5-doc is used
- c) The kernel is version 2.2.x (for example, 2.2.5-15)
- d) LILO is used as the boot loader.

2.2.4 Creating and Populating the DiskOnChip Module Directory

To Create and Populate the DiskOnChip Module Directory:

1. Change into the directory where the tarball was unpacked:

```
cd /usr/src/Linux_DOC_4.2.1/driver
```

2. Apply the following patches:

```
./create_standalone driver_patch
```

3. Create the device nodes for the DiskOnChip device (/dev/msys/fl*):
(Skip this step if applied in Section 2).

```
/mknod -f1
```

2.2.5 Building the DiskOnChip Module (doc.o)

If using a new or newly modified kernel source tree (i.e., the kernel sources were just installed or modified other than in Section 2.2.), it is required to build the kernel and then build and install the second (non-DiskOnChip) modules, (refer to Section 2.2.1, steps 7-9), before building the DiskOnChip module. Once the kernel and other modules are ready, build the DiskOnChip module as follows:

1. Change into the newly created DiskOnChip-module directory in the kernel source tree:

```
cd /usr/src/Linux_DOC_4.2.1/driver/doc-module
```

2. Build the module.

```
make TOPDIR=/usr/src/linux
```

3. Install the module in the appropriate place, and let the module system know it is there:

```
cp doc.o /lib/modules/2.2.5-15/block
```

```
depmod
```

2.2.6 Installing the DiskOnChip Board

If you started with Section 2.1, and the DiskOnChip is already installed, skip steps 1 through 3.

1. Power OFF your machine.

```
shutdown -r now
```

2. Install the DiskOnChip board.

3. Boot with a DOS floppy containing the DiskOnChip utilities, and ensure that the DiskOnChip is not set up as the boot device:

```
dformat /win:d000 /S:doc2.fff
```

4. Boot with the kernel that matches the newly built module. (If the module was built in the distributed kernel src tree /usr/src/linux, a normal boot is required.)

5. Load the module

```
modprobe doc.o
```

6. Verify that the DiskOnChip is operational.

```
cd /mnt
```

```
mkdir doc
```

```
mount -t vfat /dev/msys/fla1 /mnt/doc
```

7. The DiskOnChip is now ready for operation. Place all required files and directories into the DiskOnChip (/mnt/doc/*)
 - If it is required to load the DiskOnChip module during the boot/init process, add lines to /etc/rc.d/rc.sysinit after checking the module dependencies and root file system and before the local file systems are checked and mounted to load the doc.o module.
 - For RedHat 6.0, add the lines prior to the "# Add RAID devices" line. Also, for RedHat 6.0, use the "action" command to match the rest of the boot/init actions:

```
# Add the DiskOnChip device
```

```
action "Loading Disk-On-Chip module" modprobe doc
```

Once the DiskOnChip module is being loaded and is operational, add an entry to /etc/fstab to mount the DiskOnChip at boot time.

3 Creating a Linux Partition on the DiskOnChip

To create a Linux partition on the DiskOnChip, all the DOS partitions on the DiskOnChip need to be removed and a Linux partition with the native Linux File system [ext2] needs to be created.

Note: To create more than one primary partition, ensure that the boot flag of the main partition boot is active.

To Create a Linux Partition:

1. Run the fdisk utility.

```
# fdisk /dev/msys/fla
```

2. Display the contents of the partition table:

```
Command(m for help):p
```

3. Delete all existing partitions, enter each partition number for deletion:

```
Command(m for help):d
```

4. Create a new Linux native partition:

```
Command(m for help):n
```

```
Command action    e extended
```

```
p primary partition (1-4)p
```

```
Partition number (1-4): 1
```

```
First cylinder (1-XXX): 1
```

```
Last cylinder or +size or +sizeM or +sizeK ([1] XXX):XXX
```

5. Change the type of the partition to Linux native:

```
Command (m for help): t
```

```
Partition number (1-4): 1
```

```
Hex code (type L to list codes): 83
```

6. Make the partition bootable:

```
Command (m for help): a
```

```
Partition number (1-4): 1
```

7. Recheck the partition table:

```
Command (m for help): p
```

```
Disk /dev/msys/fla: 16 heads, 9 sectors, 1002
```

```
Cylinder units = cylinders of 144 * 512 bytes
```

```
Device boot begin Start End Blocks Id System
```

```
/dev/msys/fla1 *      1      1  1002 72139+ 83 Linux native
```

8. Save the new partition table (disregard any fdisk warnings):

```
Command (m for help): w
```

9. Reboot the machine to let the new partition table load into memory and then reload Linux with the new compiled kernel. If there is more than one partition, this step is vital.

3.1 Creating a Native Linux File System on the DiskOnChip

To Create a Linux File System:

1. To initialize the file system on the newly created Linux partition on the DiskOnChip type the following command:

```
# mke2fs /dev/msys/fla1
```

Note: When a small capacity DiskOnChip is used (4MB or smaller), more space for inodes needs to be allocated. Type the following command: `# mke2fs -i 2048 /dev/msys/fla1`

2. Mount the file system to a directory:

```
# mkdir /diskonchip
```

```
# mount /dev/msys/fla1 /diskonchip
```

The DiskOnChip can now be used as an additional disk in your system.

Note: If more than one partition was created, repeat the last step for each partition (dev/msys/fla1, /dev/msys/fla2, etc.).

4 Booting Linux from the DiskOnChip

Being able to boot Linux from the DiskOnChip makes it possible to use the DiskOnChip as the only disk in the system. This enables holding the OS itself in addition to all other applications and files.

Note: A Linux partition must be created on the DiskOnChip (refer to Chapter 3) before continuing.

To make a block device bootable on Linux, the kernel and the LILO program must be copied to the block device. Additionally, a root file system needs to be created. Creating a root file system on Linux is required to activate the following programs:

Program	Description
Init	Initialize all processes
Swapon	Activate swapping
Mount	Mount the root and proc file systems
Sh	Shell

For further details refer to <http://sunsite.unc.edu/LDP/> or type:

```
#zcat /usr/doc/HOWTO/Bootdisk-HOWTO.gz | more
```

Warning: Notice that the DiskOnChip firmware (i.e. `doc42.exb`) collides with LILO. This means that it is not possible to load Linux from the HDD after the original firmware is restored (refer to Chapter 6. This does not mean that the HDD is non-functional, only that the alternate firmware (`doc2.fff`) needs to be reloaded (refer to Section 2.1).

If it is required to boot Linux from both the HDD and the DiskOnChip, it is necessary to use both `doc-lilo` and `doc.b` (provided with the TrueFFS driver). For further details, refer to Section 6.

4.1 Creating a Root File System

The DiskOnChip can be configured to present a BIOS extension that enables booting from the DiskOnChip. The DiskOnChip can be either the first or the last kernel in the boot chain. The kernel to be booted from the DiskOnChip can access the DiskOnChip by either method. However, only the loadable module method is redistributable and it is used in this case.

If a driver compiled with the kernel is used, skip all steps related to INITRD (initial RAM disk). Setting up a bootable DiskOnChip board requires 6 steps, although it may be required to repeat the last 2 steps several times until all required files are loaded onto the DiskOnChip.

Note : (For RedHat distribution): When developing the bootable image on the same machine containing the installed DiskOnChip, it is required to boot from the DiskOnChip and to boot from the normal hard drive. The easiest way to do both, is to prepare a floppy boot disk using the "mkbootdisk" command. For example:

```
mkbootdisk 2.2.5-15.
```

4.1.1 Making the DiskOnChip Accessible

This chapter is only relevant when it is required to boot and load Linux from the DiskOnChip without using the kernel version of the driver. If it is required to boot from the DiskOnChip with a driver that is not part of the kernel, first load the driver prior to accessing system root files placed on the DiskOnChip. Using the initial RAM disk does this. When using INITRD, the system boots as follows:

1. The boot loader loads the kernel and the initial RAM disk.
2. The kernel converts INITRD into a "normal" RAM disk and frees the memory used by INITRD.
3. INITRD is mounted read/write as root.
4. /linuxrc is executed (this can be any valid executable, including shell scripts; it is run with uid 0 and can do basically everything init can do). In our case linuxrc loads DiskOnChip driver.
5. When linuxrc terminates, the "real" root file system is mounted.
6. If a directory /INITRD exists, the INITRD is moved to this directory otherwise, INITRD is unmounted.
7. The usual boot sequence (e.g. invocation of /sbin/init) is performed on the root file system.

For more detailed information regarding INITRD refer to:
<http://www.linuxhq.com/kernel/v2.2/doc/initrd.txt.html>

Note : The kernel must be compiled with RAM disk support and with support for INITRD enabled.

Prepare an INITRD image that contains the doc.o module.

To Enable INITRD:

1. Run make menuconfig and select the required options.
2. Run make zImage (or bzImage)

4.1.2 INITRD Building Process

The following describes the RAM disk method:

1. Use a RAM disk device /dev/ram (block, major 1, minor 0).
2. Create an empty file system of the appropriate size, for example,

```
# mke2fs -m0 /dev/ram 3500
```

(If space is critical, use the Minix FS instead of Ext2).

3. Mount the file system on an required directory, e.g.

```
# mount -t ext2 /dev/ram /mnt/ram
```

4. Create dev directory:

```
# mkdir /mnt/ram/dev
```

```
# cp -a /dev/tty? /mnt/ram/dev
```

```
# cp -a /dev/con* /mnt/ram/dev
# cp -a /dev/msys/fla? /mnt/ram/dev
```

5. Create /proc and /sbin directories:

```
# mkdir /mnt/ram/sbin
# mkdir /mnt/ram/proc
```

6. Copy insmod and doc.o to /mnt/ram/sbin directory and linuxrc to /mnt/ram directory. Note that /linuxrc's permissions must include "x" (execute).

7. Unmount the RAM disk

```
# umount /dev/ram
```

8. Copy the image to a file and then compress it.

```
# dd if=/dev/ram bs=1k count=3500 of=/boot/INITRD
# cd /boot
# gzip -f -9 INITRD
```

It then creates initrg.gz file .

9. Copy initrd.gz to the DiskOnChip /boot directory

Note: For the RedHat distribution only: (When not redistributing the kernel, this step can be avoided by using a kernel with DiskOnChip support built in). For RedHat 6.0:

```
mkinitrd -v --preload doc initrd-2.2.5-doc.img 2.2.5-15
```

This creates an INITRD image called initrd-2.2.5-doc.img. Move it into the DiskOnChip boot/ directory.

10. Create a file system on the DiskOnChip and populate it. For example, the DiskOnChip can be mounted to /mnt/doc directory .

4.1.3 Creating a Personal File System

To Create a Personal file System:

1. Create the following directories:

```
# cd /mnt/doc  
# mkdir bin dev etc lib mnt proc sbin tmp usr var
```

2. Create devices in the /dev directory. This can be done manually or by copying the /dev directory from the HDD. To save space, remove non-required devices. For example, if a SCSI drive is not available, then remove all the sd* devices.

```
# cp -dpR /dev /mnt/doc/dev
```

This command copies many unnecessary inodes to the DiskOnChip. Removing them causes no problem, providing the ones listed in the sample file system are present.

3. Copy and configure the files in the /etc directory:

```
# cp -dr /etc/rc.d /mnt/doc/etc (for RedHat)
```

OR

```
# cp -dr /sbin/init.d /mnt/doc/sbin (for SuSE)  
# cp -d /etc/inittab /mnt/doc/etc
```

4. Copy the password file and ensure that each user has its default shell installed:

```
# cp /etc/passwd /mnt/doc/etc  
# cp /etc/shadow /mnt/doc/etc (it is possible that you do not  
have this)  
# cp /etc/group /mnt/doc/etc
```

5. Create the file /etc/fstab that contains the list of files to be mounted:

```
# vi /mnt/doc/etc/fstab
```

6. Press <INS> to start editing and insert the following lines:

```
/dev/msys/fla1 / ext2 defaults 1 1  
/proc /proc proc defaults 0 0
```

7. Press <ESC>, ':', 'w' and 'q' to save the file.

Note: At this point it is possible to add more devices. For more information, refer to the man pages.

There are several programs that need to be copied to have a functional environment.

For example : /sbin/init .

Other programs (listed below) are not as essential, although it would be rather difficult to work without them. All other programs not listed below are considered optional. Copy these programs to the directories /bin or /sbin as follows:

```
# cp /bin/{program_name} /mnt/doc/bin/{program_name}
```

/bin directory:

```
cat      echo      mount
chmod    hostname    mv
chown    kill      ps
Cp       Ln        rm
cut      login     rmdir
Dd       Ls        sh
Df       mkdir     su
dircolors      mke2fs  sync
Du       mknod     umount
e2fsck   more      uname
```

/sbin directory:

```
halt      Shutdown
init      Swapoff
ldconfig      Swapon
mingetty      Telinit
mkswap      Update
reboot
rdev
runlevel
```

Note: The file name of mingetty varies with the distribution, i.e., RedHat and SuSE use mingetty, Slackware uses agetty. To find out the name of this file in your distribution, perform a grep on "getty":

```
# grep getty /etc/inittab
```

8. The /lib directory contains all the shared libraries and loaders. Only the appropriate libraries need to be copied to the /lib directory. To check which libraries are needed, type the following command for each file in these two directories:

```
# ldd /bin/{filename}
# ldd /sbin/{filename}
```

For example:

```
# ldd /sbin/mke2fs
libext2fs.so.2  /lib/libext2fs.so.2
libcom_err.so.2  /lib/libcom_err.so.2
libuuid.so.1    /lib/libuuid.so.1
libc.so.5       /lib/libc.so.5
```

This shows which libraries are required for the program mke2fs. In this example, it is necessary to copy the following four libraries:

```
# cp /lib/ext2fs.so.2  /mnt/doc/lib
# cp /lib/libcom_err.so.2  /mnt/doc/lib
# cp /lib/libuuid.so.1  /mnt/doc/lib
# cp /lib/libc.so.5  /mnt/doc/lib
```

If a long list of files needs to be copied, run the following command:

```
# ldd /bin/* > lib_list
# more lib_list
```

9. Copy the library loaders as follows:

```
# cp lib/ld.so /mnt/doc/lib          (a.out loader)
# cp /lib/ld_linux.so /mnt/doc/lib    (elf loader)
```

Use objcopy to reduce the size of the libraries. For example:

```
# objcopy -strip-debug /mnt/doc/lib/lib.so.5
```

10. For the RedHat and Caldera distributions it is also necessary to copy the configuration file for the pam library (responsible for making authentic users)

```
#cp /etc/pam.d/other /mnt/doc/etc/pam.d/other
```

11. Update /mnt/doc/etc/lilo.conf file and run LILO. If the kernel (docimg) is placed in /diskonchip/boot directory, doc-lilo is placed in /diskonchip/sbin directory and doc.b is placed in /diskonchip/boot directory. In this case, lilo.conf is as follows:

```
#
# sample LILO config file for Linux system with DiskOnChip 2000
# Copyright (C) 2000 M-Systems Flash Disk Pioneers Ltd.
```

```
#

# Start LILO global Section
boot=/dev/msys/fla
map=/boot/map
install=/boot/doc.b
disk=/dev/msys/fla
bios=0x80
read-only
prompt
timeout=50
vga = normal

# End LILO global section

#
# A Linux native filesystem on DiskOnChip 2000' first partition
#
image = /boot/docimg
root = /dev/msys/fla1
label = docimg
initrd = /boot/initrd.gz
```

Note: INITRD line is optional . Only needed if driver is used as loadable module.

Note: Doc-lilo is a special version of LILO that supports the DiskOnChip and doc.b is a special version of boot.b.

12. After lilo.conf is updated run doc-lilo:

```
doc-lilo -v -r /mnt/doc
```

13. Update the firmware in the DiskOnChip to make it bootable. First use a DOS boot disk with the DiskOnChip utilities and then the DiskOnChip linux utils.

```
dupdate /win:d000 /S:doc42.exb
```

14. Boot the DiskOnChip.

The file lists in the various OS/kernel subdirectories are only starting points, booting may produce a number of errors about missing commands or config files etc. Note these messages for use in step 15.

15. Customize the boot image.

If the run level obtained is not appropriate, reboot from the hard disk and analyze the log files on the DiskOnChip (in /mnt/doc/var/log) to determine the required changes. Make changes to the files list, and repeat steps 14 and 15.

5 Troubleshooting

If the DiskOnChip boots Linux without problems but it is required to add more programs to the Linux root file system, then mount the HDD and copy the required files.

5.1 DiskOnChip does not Boot Linux

There are several errors that can arise during boot-time:

- If the DiskOnChip does not boot, follow all instructions as described in the Chapters 3 and 5 in this manual.
- In addition, remember to update the original firmware of the DiskOnChip (i.e. doc42 . exb) with the alternate firmware (doc2 . fff) to boot Linux from the HDD.

5.2 The Kernel Boots, but will not Proceed

- VFS: Unable to mount -.

Most likely, you forgot to do:

```
# rdev /diskonchip/boot/doc2000 /dev/msys/fla1
```

- If the DiskOnChip boots and the kernel is loading, but it gets stuck after:

VFS: Mounted root (ext 2 filesystem) read only.

Most likely, the `init` program or some of its configuration files were not copied.

5.3 Cannot Log In

If you cannot login when booting Linux from the DiskOnChip, ensure that:

- The default shell is installed.
- The pam libraries were placed as explained in Section 4.1.3 step 10 (only for RedHat and Caldera).

5.4 Kernel Does not Compile Correctly or Patch Utility is Not Available

If the kernel does not compile correctly, and the TrueFFS driver does not cause the problem, refer to <http://sunsite.unc.edu/LDP/> or type:

```
#zcat /usr/doc/HOWTO/Kernel-HOWTO.gz | more
```

6 Booting From a HDD When the DiskOnChip Firmware is Active

Since LILO and the DiskOnChip firmware (i.e. `doc42.exb`) share the same area in RAM, the system hangs during boot time when using an unpatched LILO. Updating LILO solves this problem.

To Update the Existing Boot Loader:

- Copy `doc-lilo` to `/sbin` directory

```
#cp doc-lilo /sbin/lilo
```
- Copy `doc.b` to `/boot` directory

```
#cp doc.b /boot/boot.b
```

Run LILO command as usual. This command uses the patched LILO supplied by the TrueFFS driver, the patched `boot.b`, and the default `/etc/lilo.conf`. Ensure that this command is run in the Linux that was booted from the HDD.

Additional Information and Tools

Additional information about the DiskOnChip, including Application Notes, can be found at <http://www.m-sys.com>.

Additional tools and documents are listed in the following table:

Document/Tool	Description
AP-DiskOnChip-010	Designing with the DiskOnChip 2000
AP-DiskOnChip-015	Obtaining DiskOnChip 2000 Information
AP-DiskOnChip-030	Designing with the DiskOnChip Millennium in a RISC Environment
AP-DiskOnChip-031	Designing with the DiskOnChip Millennium in a PC Environment
DiskOnChip 2000 Data Sheet	DiskOnChip 2000 Data Sheet
DiskOnChip Millennium Data Sheet	DiskOnChip Millennium Data Sheet
DiskOnChip DIMM2000 Data Sheet	DiskOnChip DIMM2000 Data Sheet
DiskOnChip Utilities	DiskOnChip Utilities User Manual
DiskOnChip DIP EVB	DiskOnChip Evaluation Board
DiskOnChip2000-PIK	DiskOnChip DIP Programmer and Integrators Kit
DiskOnChip-GANG	DiskOnChip GANG Programmer

How to Contact Us

Internet: <http://www.m-sys.com>

E-mail: info@m-sys.com

USA Office:

M-Systems Inc.
8371 Central Ave, Suite A
Newark CA 94560
Phone: 1-510-494-2090
Fax: 1-510-494-5545

Japan Office:

M-Systems Japan Inc.
Arakyu Bldg., 5F
2-19-2 Nishi-Gotanda Shinagawa-ku
Tokyo 141-0031
Phone: 81-3-5437-5739
Fax: 81-3-5437-5759

Taiwan Office:

Room B, 13th floor, No. 133
Min Sheng East Road
Taipei, Taiwan
R.O.C.
Phone: 886-2-87706226
Fax: 886-2-87706295

U.K. Office:

M-Systems UK Ltd.
PO Box 20
Chalgrove SPDO
OX44 7YP
Phone: 44-1865-891-123
Fax: 44-1865-891-391

Israel Office:

M-Systems Ltd.
Atidim Industrial Park P.O.B. 58036
Tel Aviv 61580
Phone: 972-3-647-7776
Fax: 972-3-647-6668

M-Systems assumes no responsibility for the use of the material described in this document. Information contained herein supersedes previously published specifications on this device from M-Systems. M-Systems reserves the right to change this document without notice.