Guillemot
CORPORATION

Hercules eCAFÉ SDK

Hercules Netbook Division

Revision 1.1 Rennes – 2011-08-11

Document history:

Revision	Date	Changes / Comments
Revision 1.0	2011-07-26	First revision
Revision 1.1	2012-09-18	Adding dedicated V4C details

TABLE OF CONTENTS

I. About this document

This document explains how to build and install the Hercules eCAFÉ Linux Operating System on the Hercules eCAFÉ device, including board switch settings for device booting and all kinds of boot modes, the steps to create a bootable SD card, U-Boot, as well as the boot commands for each boot mode.

1. Audience

This document is intended for software, hardware, and system engineers who are planning to use the product, and for anyone who wants to understand more about the product.

II. Overview

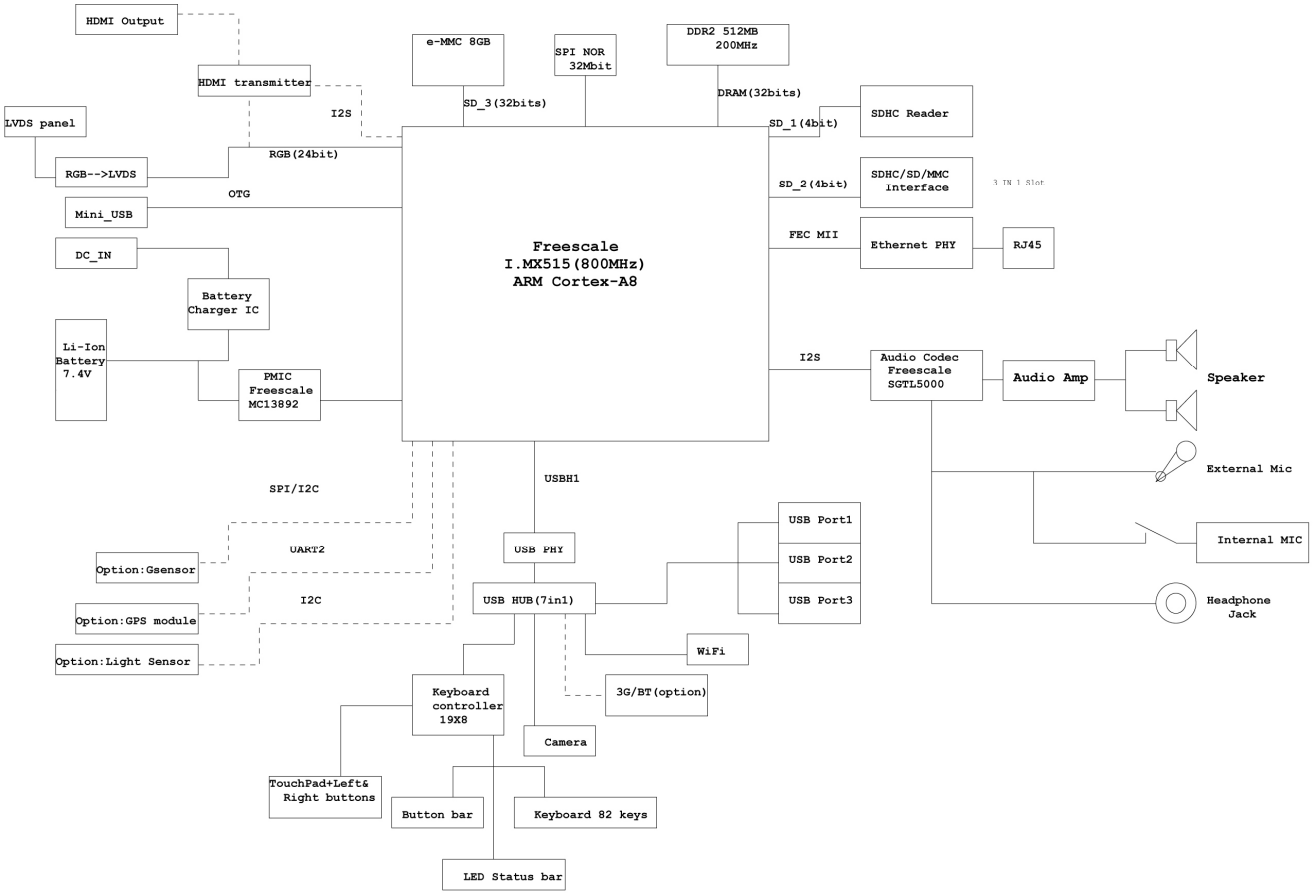
Thank you for your purchase of the Hercules eCAFÉ.

The Hercules eCAFÉ Slim HD and eCAFÉ EX HD are low-power, high-performance computers based on the Freescale i.MX515 ARM Cortex-A8™ System-on-Chip.

The Hercules eCAFÉ Linux Operating System is divided into 3 main parts:

- The boot loader (U-Boot)
- The kernel (Linux kernel 2.6.35)
- The root file system

III. Hardware architecture



1. Components of the Hercules eCAFÉ

- Freescale i.MX515 processor, T03, 800 MHz, with ARMR Cortex / A8 core
- 512 MB DDR2 RAM
- 10.1" backlit LCD display (max. native resolution 1024*600)
- 300 K pixels USB UVC built-in webcam
- Built-in sound card (SGTL5000)
- Built-in Wi-Fi 802.11 b/g/n card (up to 150 Mbps) (RT3070)
- 8 GB iNand Flash memory
- Internal Flash memory SDHC expansion slot, up to 32GB
- External Flash memory SDHC expansion slot, up to 32GB
- Rechargeable Li-Ion polymer battery: 17 W/h (7.4 V / 2400 mAh) (eCAFÉ Slim HD) / 50 W/h (7.4 V / 6800 mAh) (eCAFÉ EX HD)
- 3 USB 2.0 ports (compatible with USB 1.1 devices)
- 1 mini-USB port
- 10/100 Mbps Ethernet port
- 1 HDMI output SIL9024 (eCAFÉ EX HD only)
- 1 headphone connector
- 1 microphone connector
- 82 key Chiclet keyboard, with flat and separate keys
- 2-button touchpad
- Multimedia touch controls (eCAFÉ EX HD only)

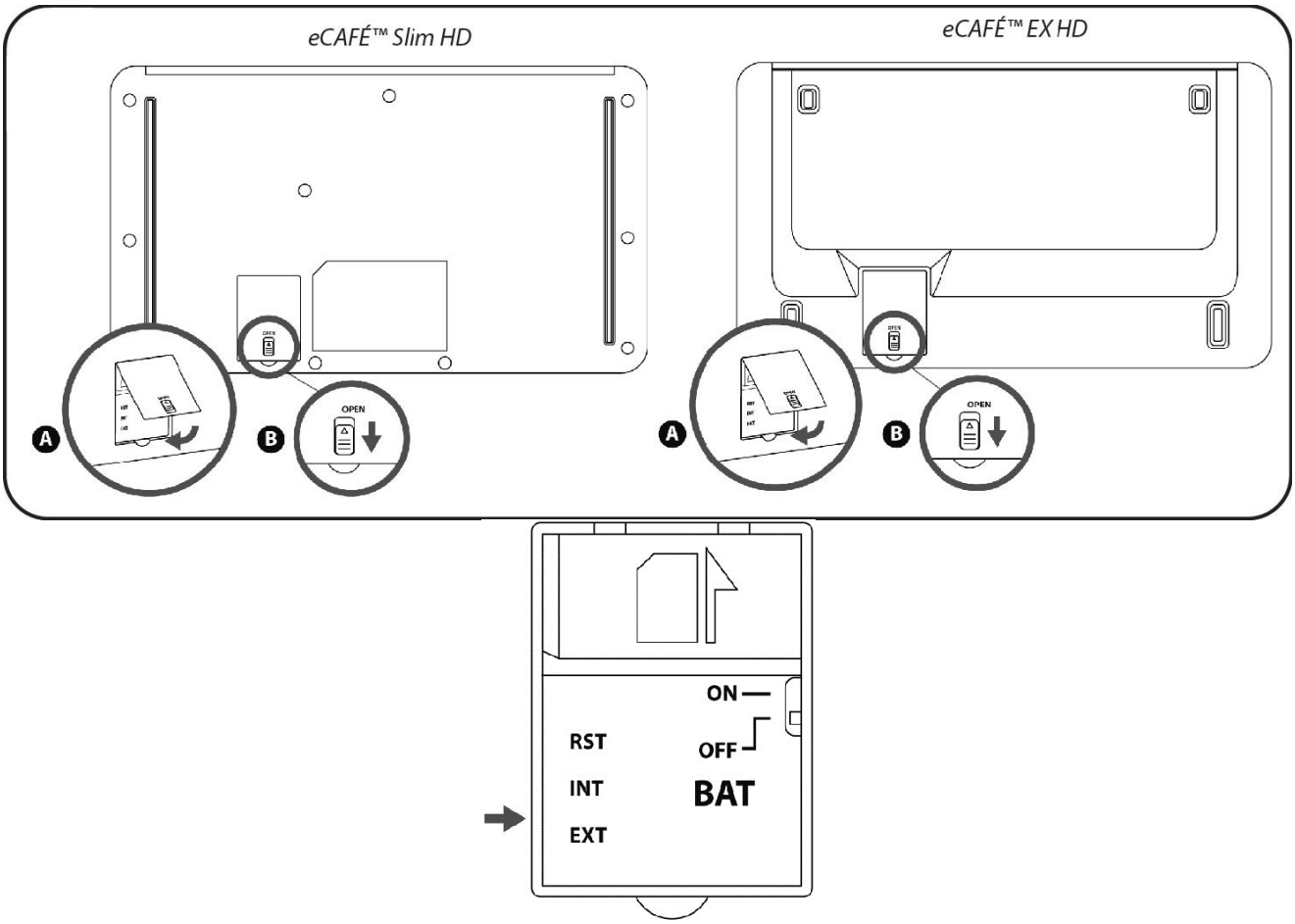
IV. Boot loader (U-Boot)

The U-Boot utility is a multi-platform, open-source, universal boot loader with comprehensive support for loading and managing boot images, such as the Linux kernel. It supports the following features:
• Network download: TFTP, BOOTP, DHCP, NFS
• Serial download: s-record, binary (via Kermit)
• Flash management: copy, erase, protect, cramfs, jffs2
• Flash types: MMC/SD, SPI-NOR
• Memory utilities: copy, dump. crc, check, mtest
• Boot from disk: raw block, ext2, fat, reiserfs
• Interactive shell: choice of simple or "busybox" shell with many scripting features

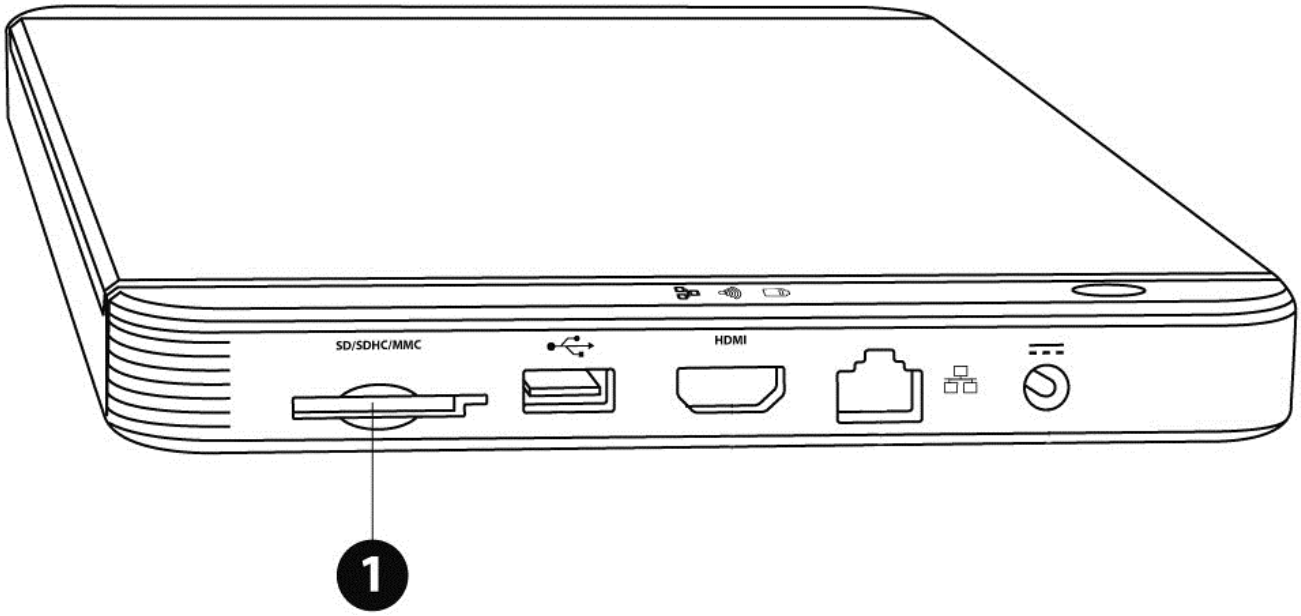
For more information on U-Boot, please refer to http://www.denx.de/wiki/U-Boot/WebHome.

By default, the boot loader is stored on the SPI-NOR flash, but it is possible to store it on an external SDHC card for testing purposes.

If you want to start the boot loader from your SD card, you need to set the Boot switch to EXT (instead of INT).



Please note that only the external SD card reader can be used as an alternative boot loader device.



2. Boot loader in SPI-NOR

The SPI-NOR flash scheme is configured statically by the software.
The SPI-NOR flash size is only 4 MB; it only has space for the boot loader and (optionally) the kernel.
Due to access time, the kernel is not stored in the SPI-NOR by default.

U-Boot	0x00000000
	0x00100000

Defaults U-Boot parameters for SPI-NOR versions are:

netdev=eth0
ethprime=FEC0
uboot_addr=0xa0000000
uboot=U-Boot.bin
kernel=uImage
bootargs_base=setenv bootargs console=ttymxc0,115200
bootargs_nfs=setenv bootargs ${bootargs} root=/dev/nfs
ip=dhcp nfsroot=${serverip}:${nfsroot},v3,tcp
bootcmd_net=run bootargs_base bootargs_nfs
tftpboot ${loadaddr} ${kernel}; bootm
load_uboot=tftpboot ${loadaddr} ${uboot}
bootargs_mmc=setenv bootargs ${bootargs} splash root=/dev/mmcblk0p1 rootwait rootfstype=ext4 U-Boot-Version
bootcmd_mmc=run bootargs_base bootargs_mmc
mmc read 2 ${loadaddr} 0x800 0x1800;bootm
bootcmd=run bootcmd_mmc

WARNING: We do not recommend updating the internal SPI-NOR. However, if you choose to do so, you can do this at your own risk.

3. Boot loader on external SD card

The SD flash scheme is different from NAND and NOR flash, which are deployed in the BSP software. SD flash must keep the first sector (512 bytes) as MBR (Master Boot Record), in order to use MMC/SD as the rootfs. At boot-up, MBR is executed to look up the partition table to determine which partition to use for booting. Boot loader should be at the end of MBR. Kernel Image and rootfs can be put at any address after boot loader. MBR can be generated through the fdisk command when creating partitions on SD cards in the Linux Host server as follows:

MBR	0x0000000
U-Boot	0x0000400 (512 x 2)
(kernel) uImage	0x0100000 (1M)
Root file System (ext4)	0x0400000 (4M = 512 x 8192)

$>sudo fdisk /dev/mmcblk1

u
d
d
n
p
1
8192
Enter
w

Then you can format the partition for the root file system:

$>sudo mkfs.ext4 /dev/mmcblk0p1

To copy the U-Boot to an SD card, you need a host computer:

$>sudo dd if=U-Boot_SD.bin of=/dev/mmcblk1 bs=512 skip=2 seek=2

/dev/mmcblk1 should be replaced according to your host: use "dmesg" after inserting the SD card to find out the location of the SD card on your host. Unmount it before issuing the dd command.

Default U-Boot parameters for SD card version are:

netdev=eth0
ethprime=FEC0
uboot_addr=0xa0000000
uboot=U-Boot.bin
kernel=uImage
bootargs_base=setenv bootargs console=ttymxc0,115200
bootargs_nfs=setenv bootargs ${bootargs} root=/dev/nfs
ip=dhcp nfsroot=${serverip}:${nfsroot},v3,tcp
bootcmd_net=run bootargs_base bootargs_nfs
tftpboot ${loadaddr} ${kernel}; bootm
load_uboot=tftpboot ${loadaddr} ${uboot}
bootargs_mmc=setenv bootargs ${bootargs} splash root=/dev/mmcblk1p1 rootwait rootfstype=ext4 U-Boot-Version
bootcmd_mmc=run bootargs_base bootargs_mmc
mmc read 1 ${loadaddr} 0x800 0x1800;bootm
bootcmd=run bootcmd_mmc

4. Kernel boot parameters

Kernel parameters	Description	Typical value	When used
console	Where to output kernel log by printk	console=ttymxc0	All cases
root	Indicates where the root file system is	root=/dev/mmcblk1p1	root=/dev/mmcblk0p1 when rootfs is on first internal 8G eMMC partition root=/dev/mmcblk1p1 when rootfs is on external SD partition
rootfstype	Indicates file system type of the root file system	rootfstype=ext4	Used together with "root=/dev/mmcblk0p1"
rootwait	Waits (indefinitely) for root device to show up	rootwait	Used when mounting SD rootfs
mmc read	Reads from eMMC or SD card	2 ${loadaddr} 0x800 0x1800	Read uImage on eMMC at offset 1M (512x0x800) Use mmc read 1 if you want to read from external SD

5. How to build U-Boot in a standalone environment

There are 2 ways to compile U-Boot:

- Compile on the device
- Cross-compile on a host computer

Due to the device's RAM size and storage limitations, we recommend using the cross-compile solution:

a) Download the source file **u-Boot-ecafe-2011.08.04.tar.gz** from our ftp server on your host Linux device.

b) Decompress the U-Boot-2009.08.tar.bz2 base line source code into a directory (your home directory). There should be a packages directory under your home directory if this is done correctly.

$ > tar xzvf u-Boot-ecafe-2011.08.04.tar.gz

c) Download and install Toolchain on your host computer:
   a. Download and Install freescale BSP1011 for i.MX515 (available at www.freescale.com)
   b. Download and Install gcc-4.4.4-glibc-2.11.1-multilib-1.0-1.i386.rm

d) Export environment variable for cross-compilation:

$> export ARCH=arm
$> export CROSS_COMPILE=/opt/freescale/usr/local/gcc-4.4.4-glibc-2.11.1-mulitlib-1.0/arm-fsl-Linux-gnueeabi/bin//arm-fsl-Linux-gnueabi-

e) Build U-Boot:
$> make mx51_na04_config (To load eCAFÉ config file)
$> make

6. How to modify U-Boot

Edit the file: /include/configs/mx51_na04.h with your preferred file editor (such as gedit).

a. Create U-Boot version for an SD card:

Uncomment the line **#define CONFIG_CMD_BOOT_ON_SD**

b. Change the default bootable partition:

Modify the line **root=/dev/mmcblk1p1** according your bootable partition.
mmcblk0 corresponds to the internal 8 GB eMMC storage.
mmcblk1 corresponds to the external SD card device present on the right of the device.
mmcblk2 corresponds to the internal SD card device present on the underside of device.

c. Update U-Boot internal SPI-NOR:

WARNING: We do not recommend this, unless you know exactly what you are doing.

You need an SD card with a minimum size of 2 GB.

- Comment the line **#define CONFIG_CMD_BOOT_ON_SD**
- Build the U-Boot version you want to store in the SPI-NOR
  - **$> make clean & make**
- Copy this version to an SD card at offset 1M
  - **$> dd if=U-Boot_spi_nor of=/dev/mmcblk1 bs=1M seek=1**
- Uncomment the line **#define CONFIG_CMD_UPDATE_SPI**
- Build the U-Boot version that will be used to update the SPI-NOR (this version is using predefined environment parameters that read the U-Boot version present at offset 1M) and copy it to the SPI-NOR
  - **$> make clean & make**
- Copy this U-Boot version to the SD card
  - **$>sudo dd if=U-Boot_SD.bin of=/dev/mmcblk1 bs=512 skip=2 seek=2**
- Insert your SD card in the external SD card reader (card reader on right of the device)
- Set the Boot switch to the EXT position (to boot from your SD card)
- Power on the device and wait until the OS start (should take approximately 3 minutes)
- Power off the device
- Set the Boot switch to the INT position (to boot from your SPI-NOR)
- Power on the device

V. Linux Kernel U-Boot Image (uImage)

The kernel is loaded by U-Boot and initializes all the others interfaces (such as USB, SD, eMMC, FEC….).

By default, the Linux Kernel Image is stored on the internal eMMC (/dev/mmcblk0), but for testing purposes, you can store in on an external SD card.

In order to load the kernel from the SD card, you need to use the U-Boot version dedicated to the SD card, or adjust U-Boot parameters to load the kernel from the corresponding SD card instead of the eMMC.

1. Kernel stored in eMMC

Default eMMC partitioning (/dev/mmcblk0)

MBR	0x0000000
(kernel) uImage	0x0100000 (1M)
Root file System (ext4)	0x0400000 (4M = 512 x 8192 blocks)
	End of eMMC

2. Kernel stored on SD card

MBR	0x0000000
U-Boot (SD version)	0x0000400 (512 x 2)
(kernel) uImage	0x0100000 (1M)
Root file System (ext4)	0x0400000 (4M = 512 x 8192)
	End of SD Card

3. How to build uImage in a standalone environment

a) Download the source file linux-2.6.35.4-ecafe-xx_src.tar.gz from our ftp server.

b) Decompress the linux-2.6.35.4-ecafe-xx_src.tar.gz source code into a directory (your home directory). There should be a packages directory under your home directory if this is done correctly.

$> tar xzvf linux-2.6.35.4-ecafe-xx_src.tar.gz

c) Download and install Toolchain:
   a. Install freescale BSP1011 for IMX51 (available at www.freescale.com)
   b. Or download gcc-4.4.4-glibc-2.11.1-multilib-1.0-1.i386.rm toolchain

d) Export environment variable for cross-compilation:
   $> export ARCH=arm
   $> export CROSS_COMPILE=/opt/freescale/usr/local/gcc-4.4.4-glibc-2.11.1-mulitlib-1.0/arm-fsl-Linux-gnueabi/bin/arm-fsl-Linux-gnueabi-

e) Build uImage (V2 -> V4)
   $> make ecafe_defconfig (to load eCAFÉ kernel config file)
   $> make uImage

f) Build uImage (V4C)
   $> make ecafe_V4C_defconfig (to load eCAFÉ kernel config file)
   $> make uImage

To copy the uImage to an SD card, you need a host computer:

$> sudo dd if= src/ linux-2.6.35.4/arch/arm/boot/uImage of=/dev/mmcblk1 bs=1M seek=1

/dev/mmcblk1 should be replaced according to your host: use "dmesg" after inserting the SD card to find out the location of the SD card on your host. Unmount it before using the dd command.

4. How to build kernel modules

Loadable kernel modules are object files which contains code to extend the running kernel. Some of them are embedded in the Linux kernel image, while others are loaded externally (like WiFi drivers).

You need to copy the modules to a directory which you will replicate on the target:

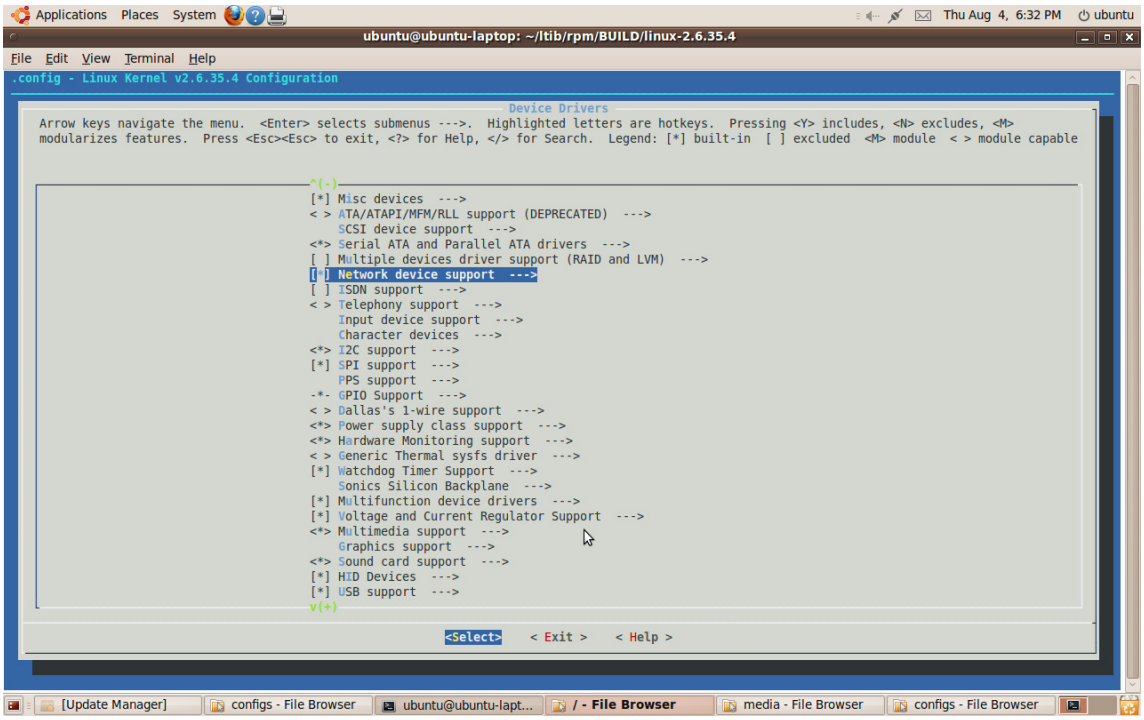
$> make INSTALL_MOD_PATH=<local_dir_module_path_for _my_target> modules_install

Then copy these files to your root file system.

5. How to add a new module to the kernel

Run the command:

$> make menuconfig



In the menu, check the additional modules you want to build. Note that some of them can be built in the kernel or externally.

Build uImage and kernel modules as explained previously, and update your device.

VI. Root File System

The file system can be placed on eMMC, SD card, or NFS.

eCAFÉ Slim HD / eCAFÉ EX HD are equipped with the Hercules eCAFÉ Linux Operating System, based on the Ubuntu operating system, a part GNU-Linux distribution.

The root file system is stored on the first EXT4 partition available on the eMMC (/dev/mmcblk01p1).
It can also be stored on the external eMMC.

IMPORTANT: Bear in mind that when partitioning your mass storage device, you must keep the first 4 MB of the mass storage device free for uImage or U-Boot+uImage (external SD device).

If you want to update the file system on the eMMC, you must boot from the SD card.
Then use an external USB hard drive, and copy the new file system from the USB hard drive to the eMMC.