Efika OpenFirmware and Linux howto

I) Introduction

This document is a small guide which introduce the Efika's Firmware and how to use and play with the Liunx operating system (installation, kernel compilation...)

The Efika is a small but complete and powerful computer. It features the bplan's OpenFirmware which allows smart and quick software supports through open standards such like RTAS, CHRP and OF1275.

For example, any RTAS compliant OS will automatically takes advantage of the Efika. Moreover, the OpenFirmware device tree provides informations (interrupts, IO memory...) about on-board devices.

Linux is an free (open-source) kernel. It has been ported on lots of platforms and became very popular. The port of the Efika has been pretty trivial as both Linux and the Efika use same standards.

You will find a FAQ (Frequently Asked Question) at the end of this document.

II) Efika OpenFirmware

In this document, OpenFirmware will be shorten "OF".

II.1) Features.

- OpenFirmware/CHRP compliant
- RTAS Compliant
- built-in x86 virtual machine for graphic card.
- Drivers for the the on-board devices: Ethernet, USB, ATA, serial...
- USB support: hub, keyboard, mass storage (for USB boot)
- TFTP protocol support.

II.2) The device tree.

The OpenFirmware is organised as a tree (such like a file system). For example, the root entry looks like this:

```
ok ls /
openprom
aliases
options
packages
chosen
memory@0
cpus
rtas
failsafe
builtin@F0000000
pci@80000000
```

Any entry may have several childs. As an example, the *builtin* node contains the built-in devices of the MPC5200b:

```
ok ls /builtin/
pic@F0000500
bestcomm@F0001200
sram@F0008000
ata@F0003A00
usb@F0001000
ethernet@F0003000
sound@F0002200
serial@F0002000
pcidma@F0003800
```

Each node also features a bunch of properties (type *.properties* to print them out):
```
ok cd /builtin/ata
ok .properties
name                  "ata"
device_type           "ata"
.description          "MPC52xx ATA controller"
model                 "mpc5200-ata"
compatible            [0xD bytes]
        [000] 6D706335 3230302D 61746100 00

interrupts            [0xC bytes]
        [000] 00000002 00000007 00000003

reg                   F0003A00:80
```

II.3) Quick start

Once you start up your machine, you will enter into the OF console. The help command provides comprehensive instructions to guide you. As an example, *help boot* will tell you how to use the boot command. *help auto-boot?* will describe the *auto-boot?* variable usage. etc...

In order to boot an image (for example an OS installer), use the *boot* command:

```
ok boot mydevice myimage [arguments]
```

Where *mydevice* should be replaced by the device node or alias. *myimage* by the file name on the device, and *[arguments]* the optional arguments which would be passed to the image.

Quick tip: Use the <TAB> key to complete variables, methods and commands name.

Please refer to 'FAQ' and 'Installing Debian etch' section.

II.4) Environment variables.

Some machine settings (suck like network configuration, default boot device, load-base, etc...) can configured through the environment variables.

The *printenv* command lists all of them as well as their values. You can get a comprehensive description on each one using *help myvariable*.
for example:

```
ok help auto-boot?
auto-boot? (-- auto?)
        if true, automatically execute boot-command after power-up
```

A variable value can be changed using the *setenv* command. For example:
```
ok setenv boot-file kernel_efika root=/dev/mpc52xx_ata2 console=ttyPSC0
```

III.5) Booting from a storage device

The OpenFirmware can loads files from mass storage devices such like hard disk, USB devices (stick, memory card reader, CD-ROM drive, floppy drive...). Moreover, the OpenFirmware knows about popular file systems (FAT, ext2/3, ISO9660...). By the way, some of them are case sensitive. In order to boot:

- Plug your device on your machine and issue a reboot or shut it on. The new device will be automatically probed and the appropriate node added into the device tree. For example, a USB stick could appear here */builtin@F0000000/usb@F0001000/scsi@1/disk@0,0* and get an alias *hd0*.

- *devalias* will print out the device aliases list. By the way, these aliases are dynamicly assigned on each new start. This could look like:

```
ok devalias
Alias                 Device Path
--------------------------------------------------
ide                   /builtin@F0000000/ata@F0003A00/disk@0,0
hd                    /builtin@F0000000/ata@F0003A00/disk@0,0
scsi                  /builtin@F0000000/usb@F0001000/scsi@0/disk@0,0
hd0                   /builtin@F0000000/usb@F0001000/scsi@0/disk@0,0
scsi0                 /builtin@F0000000/usb@F0001000/scsi@1/disk@0,0
hd1                   /builtin@F0000000/usb@F0001000/scsi@1/disk@0,0
eth                   /builtin/ethernet
```

- Additionally, *show-scsi* gives your some more details:

```
ok show-scsi
MPC5200 ATA controller:
  Disk |          | TOSHIBA MK2018GA | Q2.0 | SCSI-0

USB Storage:
  Disk |          | USB DISK Pro     | PMAP | SCSI-0 removable
```

- *ls* prints the device content out:

```
ok ls hd0
RDB partition 0 <FFS>: <dhx> (0x444F5301)
RDB partition 1 <LNX>: <dhx> (0x4C4E5800)
RDB partition 2 <FFS>: <dhx> (0x444F5301)
RDB partition 3 <EXT>: <dhx> (0x45585403)
RDB partition 4 <unknown>: <dhx> (0x53575000)
RDB partition 5 <unknown>: <dhx> (0x58465300)
RDB partition 6 <EXT>: <dhx> (0x45585403)

ok ls hd0:1
.
..
lost+found
etc
media
var
```

```
usr
bin
boot
dev
[...]

ok ls hd0:1 boot/
.
..
System.map-2.6.8-3-powerpc
config-2.6.8-3-powerpc
vmlinux-2.6.8-3-powerpc
initrd.img
vmlinux
initrd.img-2.6.8-3-powerpc
vmlinuz-2.6.8-3-powerpc
kernel_efika
```

- the *boot* command will load and execute a image. For example, to boot "kernel_efika" in the boot directory from the ATA disk partition 2:

```
ok boot hd0:1 boot/kernel_efika
```

- To add some arguments, simply append them.

```
ok boot hd0:1 boot/kernel_efika root=/dev/sdc2
```

II.6) Network boot.

The Firmware can also load data from a TFTP server. There are several environment variables to configure the network:
- *boot-protocol*: dhcp or rarp
- *server-ip*: default server ip if not specified
- *client-ip*: default client-ip if not specified

By default, *server-ip* and *client-ip* are empty and the boot-protocol is *dhcp*. If you network features well-configured DHCP, BOOTP and TFTP servers, then the Efika will automatically get its IP, the server IP and the image to load. You should then only type *boot* in the OF console. If you wish to load a file using a static IP address change the boot-protocol to *rarp*, and modify *server-ip* and *client-ip* to your flavour.

You can also specify in the command line the IPs and boot arguments like this:

```
ok load or boot eth:[server-ip],[file-to-load],[machine-IP],[gateway-ip]
[file-to-load] [arguments]
```

For example, to boot the file "kernel_efika" with "root=/dev/mpc52xx_ata1" as argument on the server 192.168.0.20 and with 192.168.0.22 as machine IP type:

```
ok boot eth:192.168.0.20,kernel_efika,192.168.0.22 kernel_efika
root=/dev/mpc52xx_ata1
```

Notice here that the gateway ip has been omitted as it is not required.

If one IP is missing, the OF will either use the default one (*server-ip* and/or *client-ip* environment variables), if *boot-protocol* is rarp or get one using DHCP if it is dhcp.

For example:

```
ok boot eth:192.168.0.20,kernel_efika kernel_efika root=/dev/mpc52xx_ata1
```

will to the same than the last command expect that the machine IP will be obtain through DHCP (if *boot-protocol* is set to *dhcp*)

In case of problem, enable the diag mode (see FAQ) to get some more informations. The last command would print this out, with diag mode enabled.

```
ok boot eth:192.168.0.20,kernel_efika kernel_efika root=/dev/mpc52xx_ata1
Config:
IP-addr = 0.0.0.0
server-IP = 192.168.0.20
gateway-IP = 0.0.0.0
filename = kernel_efika
Boot path: /builtin@F0000000/ethernet@F0003000:192.168.0.20,kernel_efika  Boot
args: kernel_efikaDHCP .
Timeout 500ms waiting for BOOTP/DHCP packetOKDHCP Over:
IP-addr = 192.168.0.31
server-IP = 192.168.0.20
gateway-IP = 0.0.0.0
filename = kernel_efika

ARP ar
Our IP addr: 192.168.0.31
Server's IP addr: 192.168.0.20
Gateway's IP addr: 0.0.0.0
TFTP kernel_efika
```

The *Config* section indicates that no IP has been specified for the machine, but later (*DHCP Over*) the correct address has been assigned.

II.7) Web resources:

- Openfirmware.org:
http://www.openfirmware.org/

- CHRP Specification:
ftp://ftp.software.ibm.com/rs6000/technology/spec/chrp/

- OpenFirmware specification:
http://playground.sun.com/1275/home.html

III) Linux Kernel

III.1) Introduction.

At www.efika.de/download/, you can find the needed resources to build/install a Linux operating system. The support in the official kernel is planed into 2.6.20. Unfortunately, some drivers are not (yet) part on the official kernel. However, those are available various developers' places, as patches and into the sources tar ball provided.

For latest linux kernel development, check theses various resources;

- Official linux kernel: http://www.kernel.org
- PowerPC specific Linux development
  - http://patchwork.ozlabs.org/linuxppc/
  - https://ozlabs.org/mailman/listinfo/linuxppc-dev
- Linux PowerPC development gits:
  - http://git.kernel.org/git/?p=linux/kernel/git/paulus/powerpc.git
  - http://gitbits.246tnt.com/gitweb.cgi?p=linux-2.6-mpc52xx.git
  - http://git.secretlab.ca/

III.2) Installing Debian etch

!WARNING!

Any OS installation may erase data on your hard disk. Make sure you do not have any important data on the target disk!

Debian is a free operating system (OS) which uses the Linux kernel (the core of an operating system), but most of the basic OS tools come from the GNU project; hence the name GNU/Linux.

This paragraph is not aim to be a full installation guide for Debian. Please refer to http://www.debian.org/doc/ for a complete documentation.

If you wish to load the linux kernel from your hard disk, it is required to format the */boot* using a file system the OF can read. We recommand to use ext2 file system.

Let's start, you need:
- A mobile storage device such like USB stick or another machine on your network.
- An internet access through proxy or gateway
- A hard disk correctly mounted on your Efika

Optional but recommended: a graphic card and an USB keyboard.

First, retrieve a so called "Debian Installer" (shorten "di"). A efika's di is available at http://www.efika.de/download/. Once you have got the file, boot it on your Efika.

Then, follow the installer instructions. The current debian installer try to loads kernel and modules for Efika. However, the Efika is not yet supported on official debian mirrors. As a result, the di issues some warnings but all of them can be safely ignored.

1. The di is unable to locate correct kernel modules. Ignore and answer "Yes"
2. The di will complain that no default partition type is known for this architecture. Simple ignore and enter yes. The di will create a DOS/MBR style partition table if you let the installer do the partitions automatically.
3. The di can not find any appropriate kernel. This is harmless as an appropriate kernel can be found on the www.efika.de/download/. Moreover, the debian kernel should support the Efika in a close future. Simply enter yes. To install a kernel "by hand" refer to the "Precompiled kernel" section.

Only this three points differs from a normal installation, the rest is the same as any others di. See Debian documentation at http://www.debian.org/doc/.

At the end of the installation, the installer will tell you which partition is the Linux root "/". A typical install will use */dev/sda1* as root. This argument has to be passed to the kernel when booting.

```
>boot kernel_efika root=/dev/sda1.
```

Quick tip: For normal desktop usage on embbeded oriented platform, you may want to install a light desktop manager (like Black Box, Window Maker).

Look at the official Debian web site for up to date release: http://www.debian.org/

III.3) Precompiled kernel.

A precompiled kernel and modules are available at www.efika.de/download/. This kernel should be suitable for most of the distribution. Simply boot this kernel and then copy the modules into right directory (this could depend on your distribution).

In order to install a such kernel:

- First boot the kernel with the correct root argument. Refer to section "Booting from a storage device" and "Network boot"
- Once, you have booted, copy the kernel into */boot*.
- Install the modules. On Linux Debian the kernel modules reside in */lib/modules/*. login as root. Change to the "/" and uncompress the archive.

[as root and /media contains the modules tarball]

```
> cd /
> tar -xzvf /media/modules_efika.tgz
```

It is recommended to use ext2/3 file system for the /boot directory.
You can find in the FAQ section, hits and tips to copy and install a precompiled kernel.

III.4) Kernel sources.

A source tarball as well as kernel patches and configuration are available at www.efika.de/download/. This kernel is based on a 2.6.19-rc6 modified for Efika. Most of the changes are already in the 2.6.20 development branch. However, some drivers such like the PIO/DMA ATA driver and the ethernet driver are not part of the official tree but will be added later. In the mean time, you can use the versions provided.

In order to compile your own kernel, simply unpack the tarball and type *make*. You are welcome to modify the kernel configuration and/or the sources to your tastes.

The patches against 2.6.19-rc6 are available in a separated archive:

- 01-Fix-compilation-issue-when-PPC_MPC52xx-and-PPC_MERGE-are-selected.txt
  When this two options are selected, some others defines are not done and this create a compilation error.

- 02-Add-USB-OHCI-glue-for-OpenFirmware-devices.txt
  A new OHCI HC glue for OpenFirmware. This is not Efika specific as any platform with a correct OpenFirmware tree should benefit of this new driver.

- 03-Add-MPC5200-serial-driver.txt
  The MPC5200 serial driver. It works with any MPC5200 platform.

- 04-Add-MPC5200-CPU-PIO-driver-using-libata.txt
  A libata CPU/PIO driver for MPC5200.

- 05-Add-MPC5200-specific-header.txt: MPC5200 header.
  This is new in *include/asm-powerpc/* but a similar one was for a long time in *include/asm-ppc/*.

- 06-Add-MPC5200-interrupt-controller-driver.txt:
  The MPC5200 interrupt controller driver.

- 07-Add-MPC5200-ethernet-driver.txt:
  Ethernet driver for MPC5200/Efika. This one is Efika specific and can not be directly be used on others platforms.

- 08-Add-MPC5200-SDMA-PIO-driver.txt:
  A block driver for the MPC5200 ATA. See « The PIO/SDMA ATA driver » section.

- 09-Added-RTAS-support-for-32bit-PowerPC.txt:
  A small patch will add RTAS support in */proc/* for 32bit PowerPC platform.

- 10-Add-Efika-platform.txt:
  Add an entry in *arch/powerpc/platforms/* for the Efika. It performs some low-level initialisation and registers platform specific call backs (set/get clock, pci access....).

- 11-Filter-out-efika.txt:
  The Efika is also a chrp platform. As a result, the chrp linux code may probe it and try to handle it. This patch avoid this.

- 12-Backport-of_platform.txt:
  Add « of_platform » to the kernel 2.6.19-rc6. Indeed, the Efika patches has been done for inclusion in 2.6.20. As a result, some new features have been backported to the last public kernel release.

- 13-Add-EXPORT_SYMBOL-drm_sman_set_manager-to-allow-correct-module-link.txt

A function was not correctly exported for module usage. This not Efika specific.

- 14-Added-mpc52xx_find_ipb_freq.txt
- 15-Removed-mpc52xx_find_ipb_freq.txt
- 16-Updated-to-latest-git-revision.txt
- 17-Added-mpc52xx_common.c.txt
  Add mpc52xx_common.c to the tree, and now use mpc52xx_find_ipb_freq() to get the IPB frequency in device driver.

- 18-Call-pci_read_irq_line-for-each-PCI-device-to-fix-the-IRQ.txt
  Call pci_read_irq_line() in pcibios_fixup_bus for each devices. The PCI IRQ value in the configuration space now matches the Linux value.

Despite what is written in the patches file head, most of the code has been written by Linux PowerPC developers and all the fame should go to them:

- libata driver (4) has been written by Sylvain Munaut and is based of the original driver from Benjamin Herrenschmidt
- The serial driver (3) by Sylvain Munaut and has been recently patch for of_platform by Grant Likely
- of_platform has been written by Benjamin Herrenschmidt.

To get detailed copyright and author look at the kernel sources (*.c, *.h) and the various development mailing lists/repositories/gits.

Beside the tar ball provided on our web site, you can patch your self an official kernel.
Assuming, the official kernel sources (www.kernel.org/) has been unpack into your current directory, and that you have copied the patches tar ball (*linux-2.6.19-rc6_patches.tar.gz*) and the config file (*linux-2.6.19-rc6_config*) from our web site (www.efika.de/download/) related files there. Do the following:

```
> tar -xzvf linux-2.6.19-rc6_patches.tar.gz
> cat efika_patches/* | patch -p1
> cp linux-2.6.19-rc6_config .config
```

You are now welcome to type *make* and take a long pause. A kernel bootable image will be build and store at *arch/powerpc/boot/zImage.chrp*. In order to install the modules in */tmp/* type:

```
>make INSTALL_MOD_PATH=/tmp/ modules_install
```

To create a tar ball archive of these modules in /tmp/mymodules.tgz:

```
>cd /tmp
>tar -czvf mymodules.tgz lib/modules/
```

III.5) The PIO/SDMA ATA driver.

WARNING! It is strongly advised to make a backup of your data.

This driver, which is to be considered as a work in progress, takes advantages of the Bestcomm/SDMA engine from the MPC5200b in order to offload the CPU usage and accelerate the data rate. It is a clever mixture of a, so called, "Bestcomm task" (which is part of the OpenFirmware) and an OS driver.

This driver is provided as a Linux block device (*drivers/block/mpc52xx/*). However, it also have some limitations (no ATAPI support...). In order to get maximal performances, we suggest to use this driver. If you encounter any issues, switch back to the normal PIO/CPU driver.

A new revision which will use the libata is planed.

IV) FAQ

Q: How can I get a list of the probed storage devices?
A: the command *show-scsi* will give you a such list. Furthermore, *devalias* gives also a list of current aliases (each storage node has an alias)

Q: What are the correct settings of the OpenFirmware serial console?
A:

- Baud rate: 115200 bauds
- Data: 8bits
- Parity: none
- Stop: 1bit
- Flow control: none

Q: I only get the bplan logo on my screen but no console. What is wrong?
A: The Firmware was not able to detect a keyboard and automacicaly switch to the failsafe console which is forwarded to the serial port. Check that the USB keyboard is correct connected and USB support is enabled in OF variable (the *usb-enable?* variable).

By the way, connecting many USB devices on a single non self-powered HUB may not provide enough current to every devices and cause unpredictable behaviour. It is always recommended to use self-powered hub and/or devices when using loads of USB devices.

Q: I changed the nvscript in a wrong way and I can not get any OF console anymore nor boot. Can you save me?
A: Sure.

- Plug another machine on the serial port to get a console.
- While rebooting, press <ESC>. You will enter a special low-level console 'Ikarus'
- type: "*a80000000*" ('a', '8', seven times '0', enter), then "*q*" ('q', enter). This will disable any kind of nvram configuration

Q: I have mounted an empty ATA disk on my machine. How can I boot an OS installer?
A: The OpenFirmware supports several ways to load and boot a file from the "external" world. Refer to section "Booting from a storage" device and "Network boot"

Q: I would like the OF to be a bit more verbose. Is there any option?
A: Set *diag-switch?* variable to *true*. This will enable the diag mode and print out more information.

Q: I have an hard disk with a fully installed and configured Linux PowerPC and the kernel (image and modules) on another computer. How can I transfer and install these precompiled binaries on my machine?
A: There are two main solutions: Transfer from the source computer to the target one using

network protocol (FTP, SSH, etc...) or using a USB storage device.

For example, using SSH. Assuming the kernel binaries (kernel_efika and modules_efika.tgz) are in your current directory, your target machine is connected to your network with the IP 192.168.0.25 and you have SSH running on both computer.

- Copy the precompiled binaries:
  - *scp kernel_efika root@192.168.0.25:/boot/*
  - *scp modules_efika.tgz root@192.168.0.25:/tmp/*
- Login on the target machine as root
  ssh root@192.168.0.25
- Install the modules. Warning, this may depend of the Linux distribution. Consult your distribution documentation for specfic informations.
  - *cd /*
  - *tar -xzvf /tmp/modules_efika.tgz*
- Reboot
  reboot

Using a USB device. Assuming the kernel binaries (kernel_efika and modules_efika.tgz) are in your current directory, and your USB stick is correctly mounted in /media/ on your source machine.

- Copy the precompiled binaries:
  cp kernel_efika modules_efika.tgz /media/scsidisk/
- Umount the device from your source machine, plug it on your target machine and mount it. /deb/sdb is assumed to be the USB device. You also need to login as root.
  - *umount /media*
  - [unplug, plug and login as root]
  - *mount /dev/sdb /media/*
- Copy the kernel in /boot:
  cp /media/kernel_efika /boot/
- Install the modules:
  - *cd /*
  - *tar -xzvf /media/modules_efika.tgz*
- Reboot:
  reboot

You can now boot from your hard disk. Assuming your disk alias is *hd*, and that the first parition (*/dev/sda1* at the Linux side) is the system root:

```
ok boot hd:0 boot/kernel_efika root=/dev/sda1
```

Q: I do not want to enter a boot command each time I shut my machine on or issue a reboot. How can I automate the boot process?
A:

- First, you will have to set the default boot device. As an example, for partition 1 of the ATA disk:
  setenv boot-device /builtin@F0000000/ata@F0003A00/disk@0,0:0
- then the kernel file and its arguments:
  setenv boot-file zImage.chrp root=/dev/sda2
- finally, automatic boot:
  setenv auto-boot? true
- Optional: You can also modify the *auto-boot-timeout* variable to accelerate or slow down the boot timeout. The value is in milliseconds. To set a timeout of 3 seconds:

  setenv auto-boot-timeout 3000

Q: I would like to install Linux (for example, Debian etch using the Efika's di) but I have already an hard disk installed with important datas. Can I still install this OS without losing them?

A: First, it is **ALWAYS** a good idea to **backup** data. So before playing with what is important to your eyes, store them in a **safe** place. Now the answer: Yes and no.

A small Linux installation requires 1Gb of free unpartitioned disk space or as partition

- If you still have a such space **unpartitioned** then the installer is able to select

  it and create appropriate partitions.
- If you have at least one big enough partition without any data. Then, you can manually select this one as root partition in the installer. This will **erase** the data contained on your partition.
- If you do not have any free unpartitioned space, nor free partition. Then, you will have to make some space, or install another disk.

Q: What is the device entry for the serial port in the Linux kernel?

A: The nomal entry is */dev/ttyPSC0*. Add *console=ttyPSC0* as kernel boot argument to redirect the initial console to the serial port.

Web resources:

- bplan GmbH:
  - Main site: http://www.bplan-gmbh.de/
  - Efika web site: http://www.efika.de

- Official Linux kernel web site:
  http://www.kernel.org/

- OpenFirmware:
  http://www.openfirmware.org/

- CHRP Specification:
  ftp://ftp.software.ibm.com/rs6000/technology/spec/chrp/

- OpenFirmware specification:
  http://playground.sun.com/1275/home.html

- Various Linux development site:
  - http://patchwork.ozlabs.org/linuxppc/
  - https://ozlabs.org/mailman/listinfo/linuxppc-dev
  - http://git.kernel.org/git/?p=linux/kernel/git/paulus/powerpc.git
  - http://gitbits.246tnt.com/gitweb.cgi?p=linux-2.6-mpc52xx.git
  - http://git.secretlab.ca/