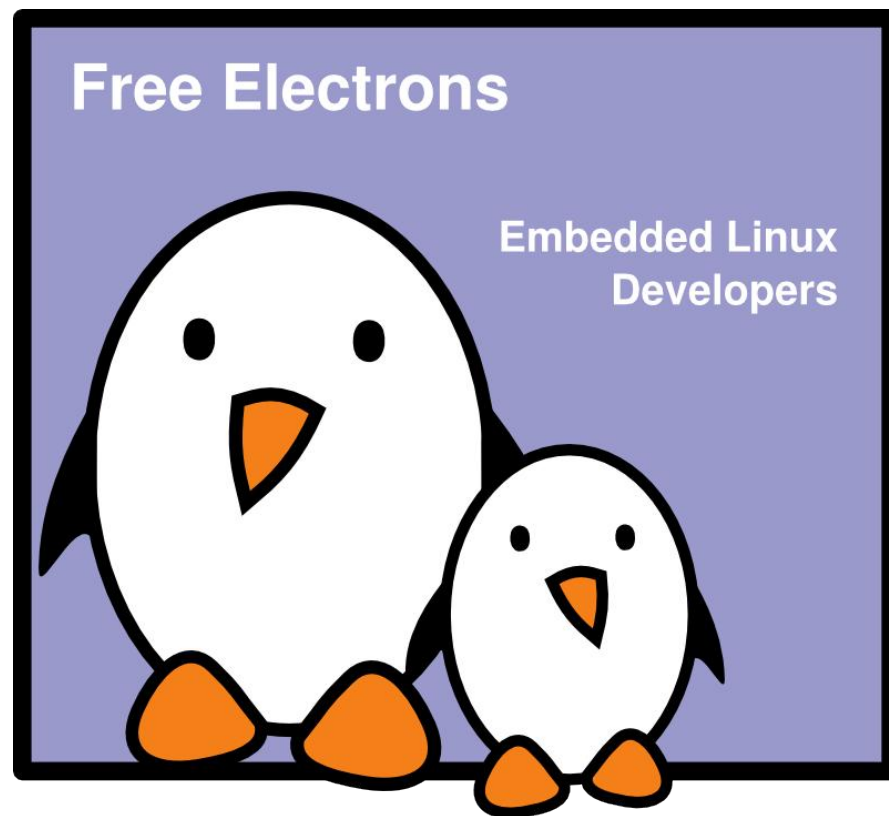




## The U-boot bootloader

Michael Opdenacker  
Thomas Petazzoni  
**Free Electrons**



© Copyright 2004-2009, Free Electrons.

Creative Commons BY-SA 3.0 license

Latest update: Jan 19, 2011,

Document sources, updates and translations:

<http://free-electrons.com/docs/u-boot>

Corrections, suggestions, contributions and translations are welcome!



# U-Boot

U-Boot is a typical free software project

- ▶ Freely available at <http://www.denx.de/wiki/U-Boot>
- ▶ Documentation available at <http://www.denx.de/wiki/U-Boot/Documentation>
- ▶ The latest development source code is available in a Git repository:  
<http://git.denx.de/cgi-bin/gitweb.cgi?p=u-boot.git;a=summary>
- ▶ Development and discussions happen around an open mailing-list <http://lists.denx.de/pipermail/u-boot/>
- ▶ Since the end of 2008, it follows a fixed-interval release schedule. Every two months, a new version is released. Versions are named YYYY.MM.



# Compiling U-Boot (1)

- ▶ Get the source code from the website, and uncompress it
- ▶ The `include/configs/` directory contains one configuration file for each supported board
  - ▶ It defines the CPU type, the peripherals and their configuration, the memory mapping, the U-Boot features that should be compiled in, etc.
  - ▶ It is a simple `.h` file that sets pre-processor constants. See the [README](#) file for the documentation of these constants.
- ▶ Assuming that your board is already supported by U-Boot, there should be one file corresponding to your board, for example `include/configs/omap2420h4.h`.



# Compiling U-Boot (2)

- ▶ U-Boot must be configured before being compiled
  - ▶ `make BOARDNAME_config`
  - ▶ Where `BOARDNAME` is the name of the configuration file in `include/configs/`, without the `.h`
- ▶ Make sure that the cross-compiler is available in `PATH`  
`export PATH=/usr/local/uclibc-0.9.29-2/arm/bin/:$PATH`
- ▶ Compile U-Boot, by specifying the cross-compiler prefix.  
Example, if your cross-compiler executable is `arm-linux-gcc`:  
`make CROSS_COMPILE=arm-linux-`



# Installing U-Boot

- ▶ U-Boot must usually be installed in flash memory to be executed by the hardware. Depending on the hardware, the installation of U-Boot is done in a different way:
  - ▶ The board provides some kind of specific boot monitor, which allows to flash the second stage bootloader. In this case, refer to the board documentation and tools
  - ▶ U-Boot is already installed, and can be used to flash a new version of U-Boot. However, be careful: if the new version of U-Boot doesn't work, the board is unusable
  - ▶ The board provides a JTAG interface, which allows to write to the flash memory remotely, without any system running on the board. It also allows to rescue a board if the bootloader doesn't work.



# U-boot prompt

- ▶ Connect the target to the host through a serial console
- ▶ Power-up the board. On the serial console, you will see something like:

```
U-Boot 1.1.2 (Aug  3 2004 - 17:31:20)
RAM Configuration:
Bank #0: 00000000  8 MB
Flash:  2 MB
In:      serial
Out:     serial
Err:     serial
u-boot #
```

- ▶ The U-Boot shell offers a set of commands. We will study the most important ones, see the documentation for a complete reference or the `help` command.



# Information commands

## Flash information

```
U-Boot> flinfo
DataFlash:AT45DB021
Nb pages:    1024
Page Size:   264
Size= 270336 bytes
Logical address: 0xC0000000
Area 0: C0000000 to C0001FFF (RO) Bootstrap
Area 1: C0002000 to C0003FFF Environment
Area 2: C0004000 to C0041FFF (RO) U-Boot
```

## NAND flash information

```
U-Boot> nand info
Device 0: NAND 256MiB 3,3V 8-bit, sector size 128 KiB
```

## U-Boot information

```
U-Boot> version
U-Boot 2009.08 (Nov 15 2009 - 14:48:35)
```

Can vary from one board to the other  
(according to the U-Boot compile configuration)



# Environment variables (1)

- ▶ U-Boot can be configured through environment variables, which affect the behavior of the different commands.
- ▶ See the documentation for the complete list of environment variables.
- ▶ The `printenv` command also to display all variables or one :

```
u-boot # printenv
```

```
baudrate=19200
```

```
ethaddr=00:40:95:36:35:33
```

```
netmask=255.255.255.0
```

```
ipaddr=10.0.0.11
```

```
serverip=10.0.0.1
```

```
stdin=serial
```

```
stdout=serial
```

```
stderr=serial
```



Network configuration

```
u-boot # printenv serverip
```

```
serverip=10.0.0.2
```





# Environment variables (2)

- ▶ The value of the environment variables can be changed using the `setenv` command :

```
u-boot # setenv serverip 10.0.0.2
```

- ▶ Environment variable changes can be stored to flash using the `saveenv` command. The location in flash is defined at compile time in the U-Boot configuration file.

- ▶ You can even create small scripts stored in environment variables:

```
setenv mmc-boot 'mmc init 0; if fatload mmc 0  
80000000 boot.ini; then source; else if  
fatload mmc 0 80000000 uImage; then run mmc-  
bootargs; bootm; fi; fi'
```

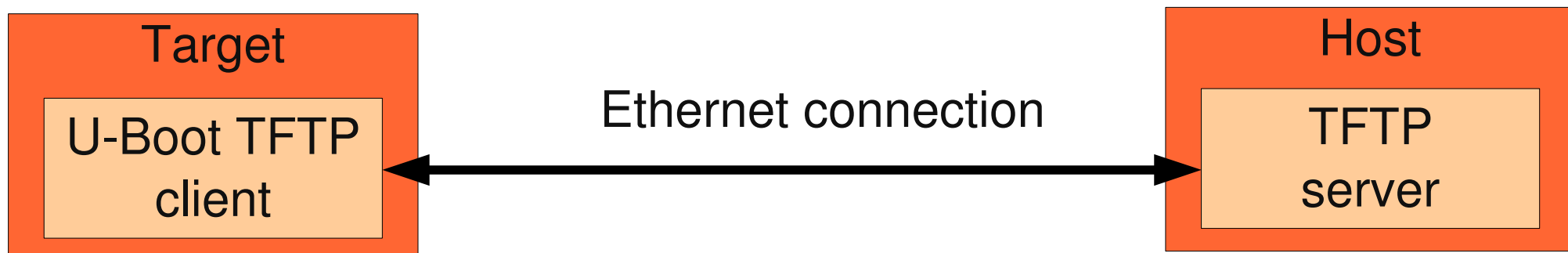
- ▶ You can then execute the script:

```
run mmc-boot
```



# Transferring files to the target

- ▶ U-Boot is mostly used to load and boot a kernel image, but it also allows to change the kernel image and the root filesystem stored in flash.
- ▶ Files must be exchanged between the target and the development workstation. This is possible :
  - ▶ Through the network if the target has an Ethernet connection, and U-Boot contains a driver for the Ethernet chip. If so, the TFTP protocol can be used to exchange files
  - ▶ Through the serial line if no Ethernet connection is available.





# Configuring and testing tftp

On GNU/Linux systems based on Debian: Ubuntu, Knoppix

- ▶ Install the `tftpd-hpa` package (tftp server):  
`apt-get install tftpd-hpa`
- ▶ Copy files to the root directory of the tftp server. Example:  
`cp arch/arm/boot/uImage /var/lib/tftpboot`
- ▶ To test the server, install a tftp client on your workstation:  
`apt-get install tftp-hpa`
- ▶ Use it to download a file (`-4` to force the use of IPv4)  
`tftp -4 localhost`  
`> get uImage`



# U-boot mkimage

- ▶ The kernel image that U-Boot loads and boots must be prepared, so that an U-Boot specific header is added in front of the image
- ▶ This is done with a tool that comes in U-Boot, `mkimage`
- ▶ Debian / Ubuntu: just install the `uboot-mkimage` package.
- ▶ Or, compile it by yourself: simply configure U-Boot for any board of any architecture and compile it. Then install `mkimage`:  
`cp tools/mkimage /usr/local/bin/`
- ▶ The special target `uImage` of the kernel Makefile can then be used to generate a kernel image suitable for U-Boot.



# Flashing a kernel image

- ▶ Compile your kernel and generate the U-Boot header running `make uImage`
- ▶ Copy the kernel image to the directory exported by the TFTP server
- ▶ On the board, in U-Boot, download the kernel image to memory :  
`u-boot # tftp 8000 uImage`
- ▶ Unprotect NOR flash  
`u-boot # protect off 1:0-4`
- ▶ Erase NOR flash  
`u-boot # erase 1:0-4`
- ▶ Copy to NOR flash (`0x01000000`: first sector)  
`u-boot # cp.b ${fileaddr} 1000000 ${filesize}`
- ▶ Restore NOR flash sector protection:  
`u-boot # protect on 1:0-4`

See our practical labs for details handling NAND flash.



# boot commands

- Specify kernel boot parameters:

```
u-boot # setenv bootargs mem=64M \  
console=ttyS0,115200 init=/sbin/init \  
root=/dev/mtdblock0
```


Continues on  
the same line

- Execute the kernel from a given physical address  
(RAM or flash):

```
bootm 0x01030000
```



# Related documents



## Free Electrons

Embedded Freedom

HOME DEVELOPMENT SERVICES TRAINING DOCS COMMUNITY COMPANY BLOG

### Recent blog posts

ELC Europe in Grenoble

Free Electrons at ELC

Linux kernel 2.6.29 - New features for embedded users

The Buildroot project begins a new life

FOSDEM 2009 videos

USB-Ethernet device for Linux

Program for Embedded Linux Conference 2009 announced

Public session changes


Real hardware in our training sessions

Call for presentations for the LSM embedded track

### Docs

Most of the below documents are presentations used in our [training sessions](#), or in technical conferences.

#### License

 All our documents are available under the terms of the [Creative Commons Attribution-ShareAlike 3.0 license](#). This essentially means that you are free to download, distribute and even modify them, provided you mention us as the original authors and that you share these documents under the same conditions.

#### Linux kernel

- [Embedded Linux kernel and driver development](#)
- [New features in Linux 2.6](#) (since 2.6.10)
- [Kernel initialization](#)
- [Porting Linux to new hardware](#)
- [Power management in Linux](#)
- [Linux PCI drivers](#)
- [Block device drivers](#)
- [Linux USB drivers](#)
- [DMA](#)

#### Architecture specific documents

- [ARM Linux specifics](#)
- [Linux on TI OMAP processors](#)

#### Embedded Linux system development

- [Embedded Linux system development](#)
- [Real time in embedded Linux systems](#)
- [Block filesystems](#)
- [Flash filesystems](#)
- [Free software development tools](#)
- [The U-boot bootloader](#)
- [The GRUB bootloader](#)
- [The blob bootloader](#)
- [Hotplugging with udev](#)
- [Introduction to uClinux](#)
- [Java in embedded Linux](#)
- [Embedded Linux optimizations](#)
- [Audio in embedded Linux systems](#)
- [Multimedia in embedded Linux systems](#)
- [Embedded Linux From Scratch... in 40 minutes!](#)
- [Building embedded Linux systems with Buildroot](#)
- [Developing embedded distributions with OpenEmbedded](#)
- [The Scratchbox development environment](#)

#### Miscellaneous

- [Introduction to the Unix command line](#)
- [SSH](#)
- [Linux virtualization solutions](#) (with an embedded perspective)
- [Advantages of Free Software and Open Source in embedded systems](#)
- [Introduction to GNU/Linux and Free Software](#)

All our technical presentations  
on <http://free-electrons.com/docs>

- ▶ Linux kernel
- ▶ Device drivers
- ▶ Architecture specifics
- ▶ Embedded Linux system development



# How to help

You can help us to improve and maintain this document...

- ▶ By sending corrections, suggestions, contributions and translations
- ▶ By asking your organization to order development, consulting and training services performed by the authors of these documents (see <http://free-electrons.com/>).
- ▶ By sharing this document with your friends, colleagues and with the local Free Software community.
- ▶ By adding links on your website to our on-line materials, to increase their visibility in search engine results.



## Linux kernel

- Linux device drivers
- Board support code
- Mainstreaming kernel code
- Kernel debugging

## Embedded Linux Training

***All materials released with a free license!***

- Unix and GNU/Linux basics
- Linux kernel and drivers development
- Real-time Linux, uClinux
- Development and profiling tools
- Lightweight tools for embedded systems
- Root filesystem creation
- Audio and multimedia
- System optimization

# Free Electrons

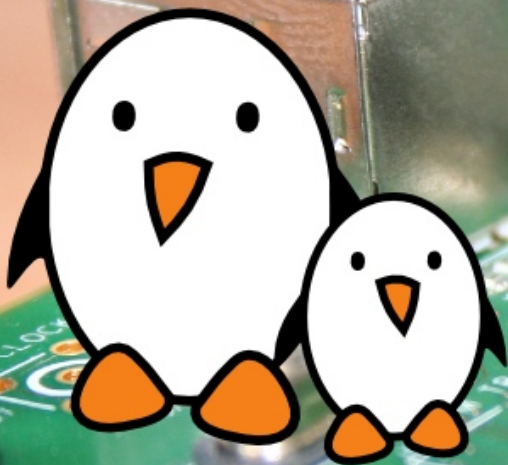
## Our services

### Custom Development

- System integration
- Embedded Linux demos and prototypes
- System optimization
- Application and interface development

### Consulting and technical support

- Help in decision making
- System architecture
- System design and performance review
- Development tool and application support
- Investigating issues and fixing tool bugs



**Free Electrons**  
Embedded Linux Experts

<http://free-electrons.com>