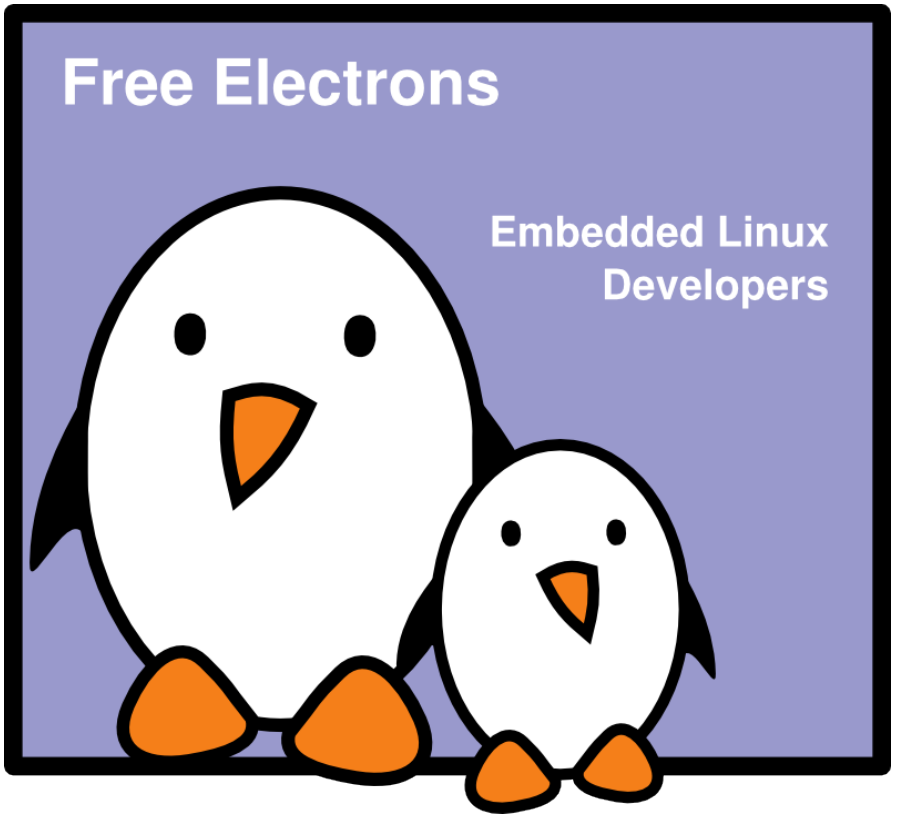
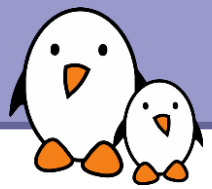
The U-boot bootloader

Michael Opdenacker
Thomas Petazzoni
Free Electrons



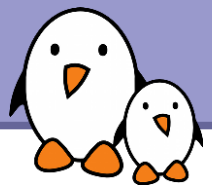


Rights to copy

Document sources, updates and translations:
http://free-electrons.com/docs/u-boot

Corrections, suggestions, contributions and translations are welcome!

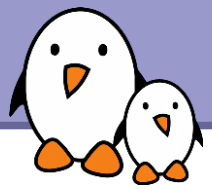
Latest update: Feb 3, 2009



Das U-boot

http://www.denx.de/wiki/UBoot/WebHome

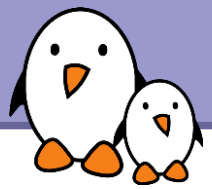
- Das U-Boot: Universal Bootloader from Denx Software
- The most used on `arm`.
- Supports: `arm`, `ppc`, `mips`, `x86`, `m68k`, `nios`...
- See our U-boot presentation for details:
  http://free-electrons.com/docs/u-boot
- Easy to port to new boards.
  See our BSP presentation for porting details:
  http://free-electrons.com/docs/bsp



Postprocessing kernel image for U-boot

The U-boot bootloader needs extra information to be added to the kernel and initrd image files.

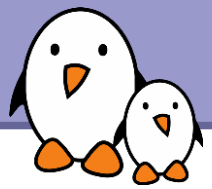
- `mkimage` postprocessing utility provided in U-boot sources
- Kernel image postprocessing:
  `make uImage`



Postprocessing initrd image for U-boot

```
mkimage
  -n initrd \                    Name
  -A arm \                       Architecture
  -O linux \                     Operating System
  -T ramdisk \                   Type
  -C gzip \                      Compression
  -d rd-ext2.gz \                Input file
  uInitrd                        Output file
```

Note: this applies to initramfs images in the same way. Bootloaders don't see the difference between initrds and initramfs.



Compiling Das U-boot

- Get the U-boot sources from http://www.denx.de/wiki/UBoot
- In the U-boot source directory:
  Find the name of the config file for your board in `include/configs`
  (for example: `omap1710h3.h`)
- Configure U-boot:
  `make omap1710h3_config` (`.h` replaced by `_config`)
- If needed, change the cross-compiler prefix in `Makefile`:

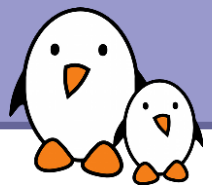
```
ifeq ($(ARCH),arm)
CROSS_COMPILE = arm-linux-
endif
```

- Specify the path to your cross-compiler. For example:

```
export PATH=/usr/local/uclibc-0.9.28-2/arm/bin:$PATH
```

- Compile:

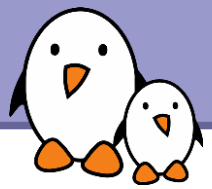
```
make
```



Compiling U-boot mkimage

If you just need `mkimage`
and U-boot is already installed on your board:

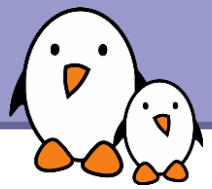
- `mkimage` is completely architecture and board independent.
- Configure U-boot sources for any board on any architecture (see previous slide).
- Compile:
  `make` (or `make -k` if you have minor failures)
- Install `mkimage`:
  `cp tools/mkimage /usr/local/bin/`



Configuring tftp (1)

Often in development: downloading a kernel image from the network. Instructions for xinetd based systems (Fedora Core, Red Hat...)

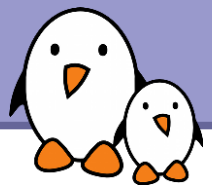
- Install the `tftp-server` package if needed
- Remove `disable = yes` in `/etc/xinetd.d/tftp`
- Copy your image files to the `/tftpboot/` directory (or to the location specified in `/etc/xinetd.d/tftp`)
- You may have to disable SELinux in `/etc/selinux/config`
- Restart xinetd:
  `/etc/init.d/xinetd restart`



Configuring tftp (2)

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

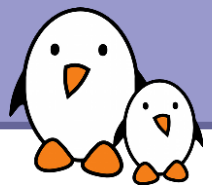
- Install the `tftpd-hpa` package if needed
- Set `RUN_DAEMON="yes"`
  in `/etc/default/tftpd-hpa`
- Copy your images to `/var/lib/tftpboot`
- `/etc/hosts.allow`:
  Replace `ALL : ALL@ALL : DENY` by `ALL : ALL@ALL : ALLOW`
- `/etc/hosts.deny`:
  Comment out `ALL: PARANOID`
- Restart the server:
  `/etc/init.d/tftpd-hpa restart`



U-boot prompt

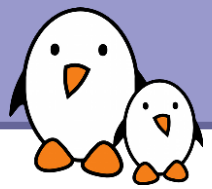
- Connect the target to the host through a serial console
- Power-up the board.
  On the serial console, you will see something like:

```
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 #
```



Board information

```
u-boot # bdinfo
DRAM bank = 0x00000000
-> start = 0x00000000
-> size = 0x00800000
ethaddr = 00:40:95:36:35:33
ip_addr = 10.0.0.11
baudrate = 19200 bps
```



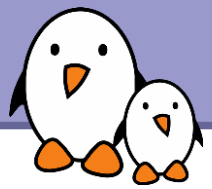
Environment variables (1)

```
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

u-boot # setenv serverip 10.0.0.2

u-boot # printenv serverip
serverip=10.0.0.2
```

Network settings for TFTP and NFS



Environment variables (2)

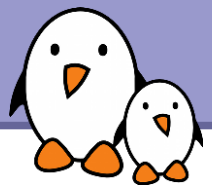
- Environment variable changes can be stored to flash using the `saveenv` command.
- You can even create small shell scripts stored in environment variables:

```
setenv myscript 'tftp 0x21400000 uImage ;
bootm 0x21400000'
```

- You can then execute the script:

```
run myscript
```

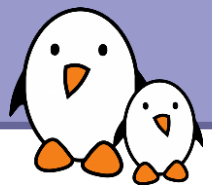
- More elaborate scripting is available with script files, to be processed with `mkimage`.



Network commands

```
u-boot # tftp 8000 u-boot.bin
From server 10.0.0.1; our IP address is
10.0.0.11
Filename 'u-boot.bin'.
Load address: 0x8000
Loading: ####################
done
Bytes transferred = 95032 (17338 hex)
```

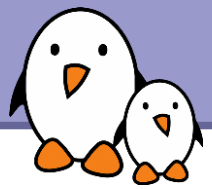
The address and size of the downloaded file are stored in the `fileaddr` and `filesize` environment variables.



Flash commands (1)

```
u-boot # flinfo
Bank # 1: AMD Am29LV160DB 16KB,2x8KB,32KB,31x64KB
Size: 2048 KB in 35 Sectors
Sector Start Addresses:
S00 @ 0x01000000 ! S01 @ 0x01004000 !
S02 @ 0x01006000 ! S03 @ 0x01008000 !
S04 @ 0x01010000 ! S05 @ 0x01020000 !
S06 @ 0x01030000 S07 @ 0x01040000
...
S32 @ 0x011D0000 S33 @ 0x011E0000
S34 @ 0x011F0000
```

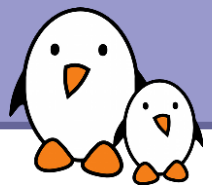
Protected sectors



Flash commands (2)

```
u-boot # protect off 1:0-4
Un-Protect Flash Sectors 0-4 in Bank # 1

u-boot # erase 1:0-4
Erase Flash Sectors 0-4 in Bank # 1
Erasing Sector 0 @ 0x01000000 ... done
Erasing Sector 1 @ 0x01004000 ... done
Erasing Sector 2 @ 0x01006000 ... done
Erasing Sector 3 @ 0x01008000 ... done
Erasing Sector 4 @ 0x01010000 ... done
```



Flash commands (3)

Storing a file in flash

- Downloading from the network:

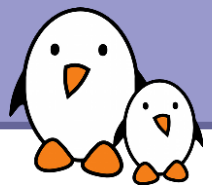
```
u-boot # tftp 8000 u-boot.bin
```

- Copy to flash (`0x01000000`: first sector)

```
u-boot # cp.b ${fileaddr} 1000000 ${filesize}
Copy to Flash... ............... done
```

- Restore flash sector protection:

```
u-boot # protect on 1:0-4
Protect Flash Sectors 0-5 in Bank # 1
```



boot commands

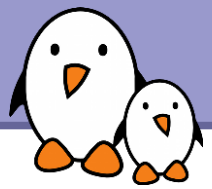
- Specify kernel boot parameters:

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

Continues on the same line

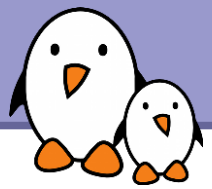
- Execute the kernel from a given physical address (RAM or flash):

```
bootm 0x01030000
```



Useful links

- U-boot home page:
  http://www.denx.de/wiki/UBoot/WebHome
- Very nice overview about U-boot
  (which helped to create this section):
  http://linuxdevices.com/articles/AT5085702347.html
- The U-boot manual:
  http://www.denx.de/wiki/view/DULG/UBoot



Related documents

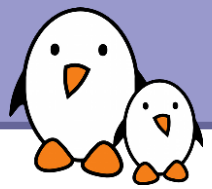
All the technical presentations and training materials created and used by Free Electrons, available under a free documentation license (more than 1500 pages!).

http://free-electrons.com/training

- Introduction to Unix and GNU/Linux
- Embedded Linux kernel and driver development
- Free Software tools for embedded Linux systems
- Audio in embedded Linux systems
- Multimedia in embedded Linux systems

http://free-electrons.com/articles

- Advantages of Free Software in embedded systems
- Embedded Linux optimizations
- Embedded Linux from Scratch... in 40 min!
- Linux USB drivers
- Real-time in embedded Linux systems
- Introduction to uClinux
- Linux on TI OMAP processors
- Free Software development tools
- Java in embedded Linux systems
- Introduction to GNU/Linux and Free Software
- Linux and ecology
- What's new in Linux 2.6?
- How to port Linux on a new PDA



How to help

If you support this work, you can help ...

- By sending corrections, suggestions, contributions and translations
- By asking your organization to order training sessions performed by the author of these documents (see http://free-electrons.com/training)
- By speaking about it to your friends, colleagues and local Free Software community.
- By adding links to our on-line materials on your website, to increase their visibility in search engine results.

Free Electrons services

Embedded Linux Training

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

Custom Development

System integration
Embedded Linux demos and prototypes
System optimization
Linux kernel drivers
Application and interface development

Consulting

Help in decision making
System architecture
Identification of suitable technologies
Managing licensing requirements
System design and performance review

Technical Support

Development tool and application support
Issue investigation and solution follow-up with mainstream developers
Help getting started

http://free-electrons.com

Free Electrons
Free Software for Embedded Systems