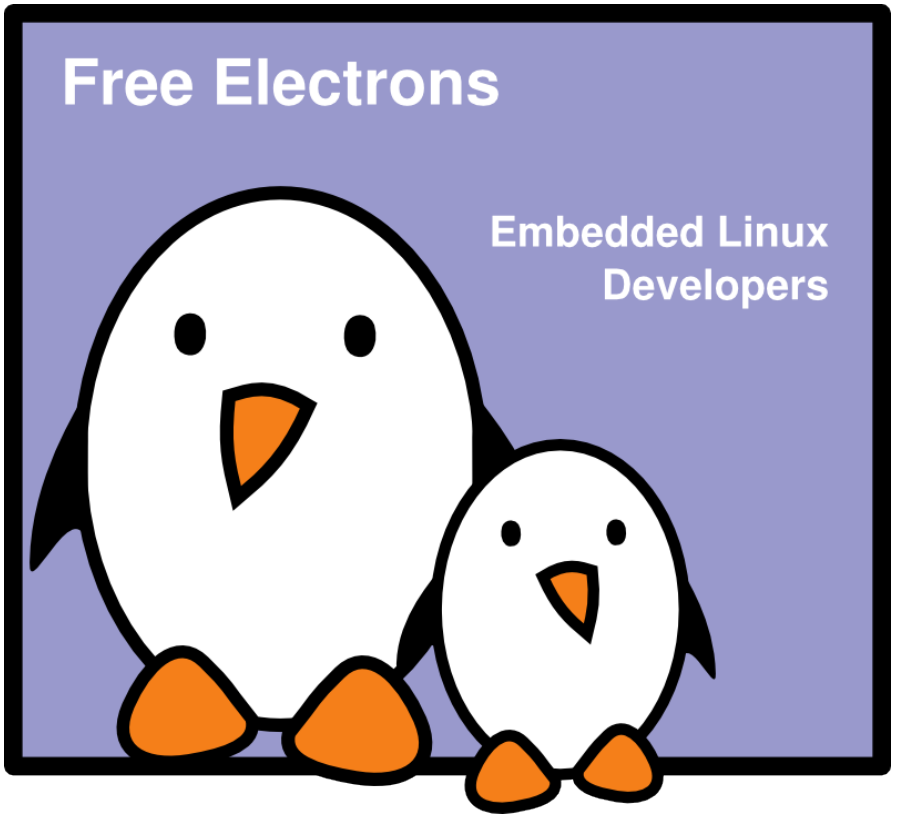
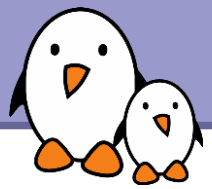
The blob bootloader

Thomas Petazzoni
Free Electrons



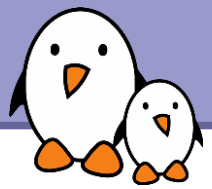


Rights to copy

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

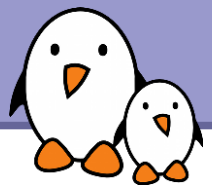
Corrections, suggestions, contributions and translations are welcome!

Latest update: Sep 15, 2009



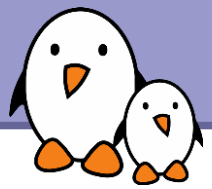
Blob bootloader

- Blob, Boot Loader Object,
  bootloader for StrongARM based platforms
  http://www.lartmaker.nl/lartware/blob/
- Written by Jan-Derk Bakker and Erik Mouw, GPL-licensed
- Simple bootloader: < 15k lines of code, including the code for all supported boards
- Inactive mainline
  - Latest stable release in august 2001
  - Latest development version in january 2002
  - Last serious activity in the CVS in 2004, few commits in 2006



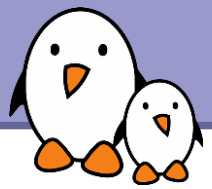
Supported boads

- Few boards supported in the released version
  - Assabet
  - Assabet+Neponset
  - LART
  - iPaq
  - System3
  - Badge4
  - NESA
  - PLEB
  - Shannon (aka Tuxscreen)
  - Vercel UD-1
  - Jornada 720
- Support for other boards can be added fairly easily
- Non-mainlined ports
  - Intel Lubbock
  - Motorola Linux phones, with download over USB
  - Others ?



Features

- Initialize hardware
- Download a kernel and ramdisk through the serial line
- Write kernel and ramdisk to Flash
- Boot the kernel
- Determine the memory layout
- Give a command line to the kernel

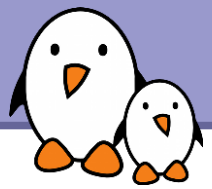


Using Blob

- On boot, by default, Blob initializes the hardware, loads and boots the kernel.
- Messages are available through the serial port, 9600 bauds, 8 bits, no parity, 1 stop bit

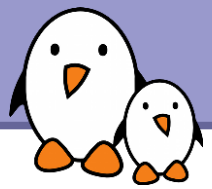
```
blob version 2.0.3
[...]
Memory Map:
  0x08000000 @ 0xC0000000 (8MB)
[...]
  0x08000000 @ 0xC9000000 (8MB)
Loading blob from flash . done
Loading kernel from flash ....... done
Loading ramdisk from flash ............... done
Autoboot in progress, press any key to stop ...
Starting kernel ...
Uncompressing Linux...done.
```

- To access the command line, press any key during the ten seconds break before kernel boot



Using Blob : Commands

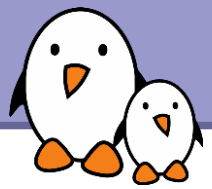
- `boot [kernel options]`
  Boot the kernel
- `download { blob | kernel | ramdisk }`
  Download blob, kernel or ramdisk from serial to RAM
- `flash { blob | kernel | ramdisk }`
  Write blob, kernel or ramdisk from RAM to Flash
- `Reblob`
  Restart blob
- `Reboot`
  Restart the system
- `reload { blob | kernel | ramdisk }`
  Reload the blob, kernel or ramdisk from Flash to RAM
- `reset`
  Reset the terminal
- `speed`
  Set the download speed
- `Status`
  Get status information



Using Blob: download (1)

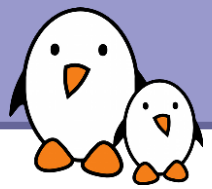
- Downloading blob, the kernel or ramdisk is done through the same serial line as the console
- When starting the download command, the speed is set to 115200 bauds and blob waits for an uuencoded version of blob, kernel or ramdisk
- On blob side

```
blob> download kernel
Switching to 115200 baud
You have 60 seconds to switch your terminal emulator
to the same speed and start downloading. After that
blob will switch back to 9600 baud.
```



Using Blob: download (2)

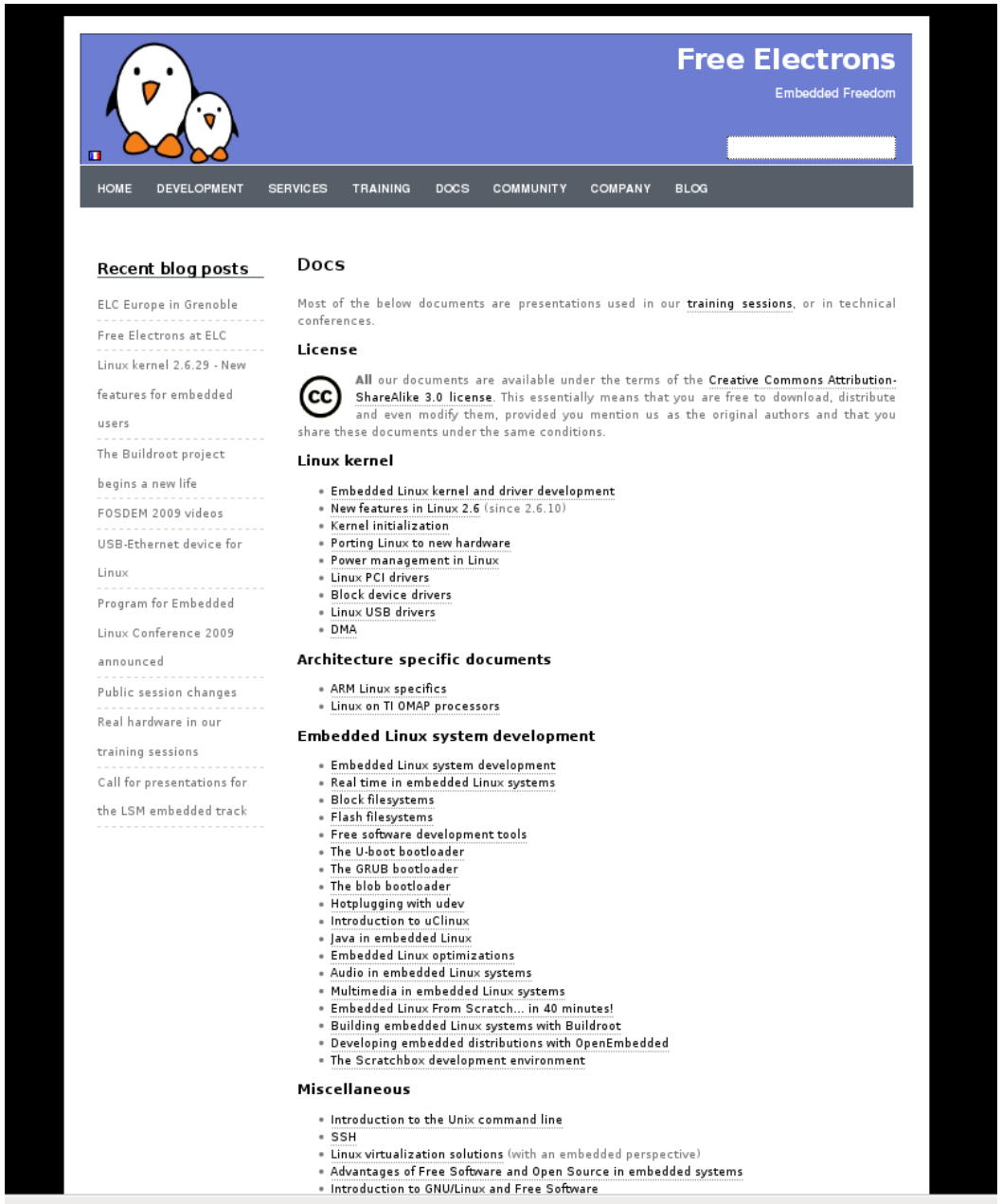
- On the host side, change the speed to 115200 bauds and either:
  - Upload the file with Minicom using the ASCII method
  - In another shell, run
    ```
    uuencode zImage zImage > /dev/ttySx
    ```
- And the end of the upload, on blob side :
  ```
  (Please switch your terminal emulator back to
  9600 baud)
  Received 65536 (0x00010000) bytes.
  ```
- On the host side, switch back the speed to 9600 bauds



Compiling Blob

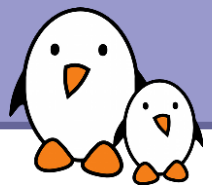
- Requires a cross-compiler and a configured Linux kernel source tree
- ```
  export CC=/path/to/arm-linux-gcc
  export OBJCOPY=/path/to/arm-linux-objcopy
  ./configure
     --with-linux-prefix=/path/to/linux/sources/
     --with-board=boardname
     arm-unknown-linux-gnu
  make
  ```
- Two versions
  - `src/blob`, the bootloader that can directly be booted
  - `src/blob-chain`, the bootloader that can be loaded from another bootloader
  ```

Related documents



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