OpenMoko
Building a truly hackable device

by

Harald Welte <laforge@openmoko.org>

Introduction

Who is speaking to you?

- an independent Free Software developer, consultant and trainer
- who is a member of the free software community for 10 years
- who has worked a lot on the Linux kernel
- who had originally started OpenEZX for Motorola phones
- and who’s been lead hardware + system software architect for OpenMoko until recently

WARNING

While I have been the Lead System Architect for hardware and system level software, throughout the first 16 months of the project,

I have quit working for OpenMoko, Inc. or the FIC group in November 2007.

Thus, I do not officially represent either of these entities!

What is OpenMoko

The commercial side

- First International Computer, Inc.
  - A large Taiwanese hardware vendor
  - Has a FIC Mobility business unit
  - Hardware R&D and production of Neo1973 GTA01 and GTA02 handsets

- OpenMoko, Inc., ("OpenMoko, the Company")
  - Part of First International Computer (FIC) Group
  - Funding the OpenMoko software R&D
  - Responsible for product definitil, sales, marketing, PR, ...

What is OpenMoko

The community side

- OpenMoko, the overall Free Software project
  - A FOSS project working on
    - OpenMoko kernel/u-boot patches (hardware support)
    - OpenMoko GNU/Linux distribution
    - OpenMoko UI / framework
  - Funded by OpenMoko, Inc.

- OpenMoko, the embedded GNU/Linux distribution
  - An OE-built embedded GNU/Linux distribution for mobile communications devices
  - Primarily targetted at OpenMoko/FIC handsets
  - Is being ported to other devices by the community
  - Maintained by OE coreteam member employed by OpenMoko, Inc.

What is OpenMoko about?

- Open
  - Opening up the formerly-closed mobile world
  - on any achievable level
- Mobile
  - Mobile devices are the future
- Free
  - 100% Free Software from driver through UI

What is OpenMoko about?

- FIC provides
  - experience in mass production of consumer electronics
  - experience in production of GSM handsets
  - experience in hardware development of GSM handsets

- OpenMoko provides
  - good contacts within the FOSS communities
  - strong technical knowledge on GNU/Linux
  - software development

Neo1973 GTA01 hardware

Neo1973 GTA01 hardware (07/2007)

- S3C2410 SoC @ 266MHz
- 2.8" 480x640 LCM, 262k colors
- 128MB SDRAM
- 64MB SLC NAND (512/16k)
- USB 1.1 device and host (unpowered)
- A-GPS (without processor)
- GSM+GPRS chipset (ARM7 based)
- Wolfson audio codec
- 2 stereo speakers (1.2W)
- 2.5mm headset jack
- CSR4 based Bluetooth
- NXP PCF50606 power management unit

Application Processor

Closer look at Application Processor

- SC2410 SoC @ 266MHz
  - three UART’s
  - 133MHz SDRAM interface
  - 66MHz external bus
  - Two channels SPI
  - IIS
  - I2C
  - SDIO
  - TFT controller
  - NAND controller

Neo1973 GTA02 hardware

Neo1973 GTA02 hardware ("soon")

- S3C2442B SoC @ 400 MHz (500MHz option)
- 2.8" 480x640 LCM, 262k colors
- 128MB SDRAM
- 256MB SLC NAND (2048/128k)
- USB 1.1 device and host (with power)
- A-GPS (fully autonomous firmware-based)
- GSM+GPRS chipset (Ti Calypso, ARM7 based)
- CSR4 based Bluetooth
- Atheros AR6k based 802.11b/g WiFi
- 2 3D accelerometers
- Smedia Glamo 3362 GPU
- NXP PCF50633 power management unit

GTA02: Smedia Glamo GPU

Smedia Glamo 3362 GPU

- 8MB internal SDRAM
- 16bit local bus interface to S3C2410
- 2D acceleration
- 3D acceleration
- H.263 codec (encode/decode)
- LCM controller
- SD-Card controller
- hardware JPEG encoder/decoder
- Camera interface and imapge processing (unused)

OpenMoko is writing 100% FOSS drivers (GPL/MIT licensed)

- kernel driver for core and framebuffer
- Xglamofb accelerated X server

GSM Modem

Closer look at the GSM Modem

- Ti Calypso/Iota based chipset
- As proprietary as any other phone
  - runs proprietary nucleus OS
  - runs proprietary GSM stack
- Supports GSM voice/data/fax and GPRS
- Tri-Band GSM
- Very good TS 07.05 / 07.07 / 07.10 compliance
  - eveyone can download the protocol docs from ETSI.org
  - no user/hacker needs access to NDA’d documents

Free Software stack

Free Software stack

- bootloader: u-boot current git (post-1.3)
- kernel: linux 2.6.24 based
- xserver: kdrive
- glibc
- glib
- gtk+
- pulseaudio
- gsmd / libgsmd

Development Model

Development Model

- "do embedded GNU/Linux the right way"
  - use and track current mainline code
  - actively contribute our code upstream
  - all code is immediately committed to public svn repository
  - development discussions happen on public mailinglists
  - all code developed by OpenMoko is FOSS licensed
  - everyone can contribute
  - no copyright assignments to OpenMoko

Build System

- We build
  - an embedded Linux distribution
  - split in ipk packages (just like dpkg/rpm)
  - ipk feeds (just like apt-get/yum)
- We release
  - full source code in svn
  - all patches to all packages
  - the entire build system (built with OE)
- Our build system is public
  - Everyone can rebuild everything
    - cross-toolchain
    - u-boot / kernel image
    - application/library packages

Hackable Device

Hackable Device

- Standards compliance wherever possible
- The device shall be under full user control
- Everyone should be able to hack it, at any level
- Make entry barrier for development as easy as possible
- bootloader prompt via USB serial emulation
- Serial console
- JTAG for the people
- Provide Debug Board with embedded USB JTAG + serial adapter

Standards compliance

Standards compliance

- We use open/documented/available standards wherever possible
- Use official USB device firmware upgrade protocol
- Have charger behave 100% to USB spec (100/500mA)
- Use GSM chipset that follows GSM 07.07/07.10 closely

User control

User control

- The phone needs to be under control of the user, and the free software he uses
- Even backdoors or rogue GSM firmware shall not be able to intrude the privacy fo the user
- So we e.g. put the Audio codec (under explicit control from the Linux-running AP) between microphone/speaker and the GSM modem
- So we enable the Linux-running AP to cut power of the GSM modem
- Thus, free software (and thus the user) remains in ultimate control

Hackable at any level

Hardware Hacking

- we even encourage hardware hacking
- I2C, SPI, GPIO and IRQ line on documented test pads and connector
- allows for attachment of new peripherals to the device
- even the hardware schematics available under FOSS-permissive NDA

Hackable at any level

System-level hacking (bootloader, OS)

- entire bootloader from very first instruction FOSS
- entire kernel including all drivers FOSS
- JTAG accessible on debug connector
- serial console on debug connector
- debug board (USB JTAG adaptor and USB serial converter)
- un-brickable through emergency boot loader in read-only NOR flash (GTA02)
- DFU (Device Firmware Upgrade) for full-system re-flash via USB

Hackable at any level

Userspace and UI level hacking

- entire userspace world FOSS (libraries, daemons, UI, X driver, ...)
- FOSS build system and toolchain/SDK enable anyone to build custom softwar packages and/or flash images
- provide a programming environment as close as possible to the Linux desktop world
- allow developers to re-use their existing Linux development skills

GSM Integration

Application Processor GSM integration

- kernel line discipline implementation for GSM 07.10
- userspace GSM daemon with unix domain socket
- libgsmd with API for applications
- kernel part intended for mainline submission
- will support different phones / gsm chipsets
  - Various HTC devices with Linux
  - Motorola EZX phones using OpenEZX

GSM Integration

But you can’t hack the GSM stack

- yes, that is true.
- pretty much like you can’t hack the firmware of your SCSI or RAID controller, WiFi card, Bluetooth chipset, etc.
- even the firmware of a good old analogue phone line (voice) modem was not hackable
- having proprietary firmware on a dedicated peripheral CPU is even acceptable to the FSF!
- And no doubt, anyone inside OpenMoko would love to ever have a open source GSM stack. Patches welcome :)

GSM Integration

But you can’t hack the GSM stack

- so you get the maximum level of freedom that you can get with any other peripheral device:
  - open source low-level (mux, power mgmt) drivers
  - open source high-level drivers (gsm daemon)
  - openly documented serial protocol (TS 07.05, 07.07, 07.10)
- asking for more freedom on the GSM side is hypocritical when accepting the very same level with other peripheral devices.

GSM Integration

But you can’t hack the GSM stack

- besides that
  - GTA01 has baseband JTAG on test pins
  - OpenMoko does not cryptographically sign GSM firmware images
  - GSM firmware is user-upgradable

Difference

Difference from other Linux phones

- ’others’ discourage third parties from writing apps
  - you need explicit permission? WTF!
- ’others’ try to make customers pay for a device that’s still under manufacturer / GSM operator control
- ’others’ use proprietary kernel modules
  - locks you into some old kernel version
- ’others’ use proprietary bootloaders
- ’others’ dont give you JTAG/serial access
- ’others’ use proprietary UI toolkits
  - vendor lock-in
- ’others’ dont give out their build system
- ’others’ dont give out their firmware update tools

Neo1973 GTA01 Emulator

The Neo1973 GTA01 emulator

- based on populer qemu project
- full GTA01 hardware emulation, including
  - NAND controller
  - LCM controller
  - power management unit
  - GSM modem
  - touchscreen controller
  - SD card controller
  - ...
- you can run the exact same bootloader/kernel/rootfs images
- thus, no need to buy real hardware to start hacking
- e.g. NetBSD port has been done entirely on emulator!
- http://wiki.openmoko.org/wiki/OpenMoko_under_QEMU

How to contribute

- First: get hands-on experience
  - with emulator (free, based on qemu, full GTA01 emulation)
  - with real hardware (GTA01 now, GTA02 soon)
- follow instructions on the wiki, improve it with your feedback
- start local user / developer groups
- go through bugzilla, look for bugs in your favourite components
  - try to reproduce bug with current images
  - provide feedback
  - help by proividing additional debugging information

How to contribute

- write your own gtk+/e17 applications fit for 480x640 screen size and limited CPU
  - do development on your host pc (native)
  - then cross-compile for OpenMoko
  - then test on emulator or hardware
  - then build and package with OE
- go through projects.openmoko.org and contact project teams, help them out
- hang out on mailinglists and #openmoko on freenode.net
  - start sharing your experience with others with your experience

Online Resources

- http://www.openmoko.org/
  - portal site, just links everywhere else
- http://wiki.openmoko.org/
  - everything you (n)ever wanted to know about openmoko ;)
- http://bugzilla.openmoko.org/
  - documents all known bugs, please add/report and debug!
- http://lists.openmoko.org/
  - various mailing lists for Q&A and discussions
- http://planet.openmoko.org/
  - planet aggregating RSS feeds of various blogs
- irc.freenode.net #openmoko
  - lots of developers hanging out there
- https://direct.openmoko.com/
  - for buying actual hardware