OpenPCD / OpenPICC
Free Software and Hardware for 13.56MHz RFID

Nov 24, 2006
FOSS.in, Bangalore

by

Harald Welte <laforge@openpcd.org>
Milosch Meriac <meriac@openpcd.org>

Introduction

Who is speaking to you?

- an independent Free Software developer
- one of the authors of Linux kernel packet filter
- busy with enforcing the GPL at gpl-violations.org
- working on Free Software for smartphones (openezx.org)
- ...and Free Software for RFID (librfid)
- ...and Free Software for ePassports (libmrtd)
- ...among other things ;)

Introduction RFID

Short introduction on 13.56MHz RFID systems

□Magnetic Coupling

□ISO 14443-A / -B (proximity IC cards)

□ISO 15693 (vicinity IC cards)

□Proprietary: FeliCa, Legic, Mifare Classic, ...

□Applications: RFID tagging (15693), Smartcards (14443)

RFID Reader Designs

Overview on available reader designs

- □Most readers based on ASIC (Philips, TI, ...) + Microcontroller
- □Readers for PC’s usually have USB, RS232 or PCMCIA IF
- □Some reader designs with Ethernet, RS-485
- □Important: If you need Mifare, you need Philips reader ASIC
- □Active readers implement protocols in firmware, passive in host sw

The OpenPCD project

The OpenPCD project

- design a RFID reader that gives full power and all interfaces
- reader hardware design is under CC share alike attribution license
- reader firmware and host software under GPL
- use hardware that doesn’t require proprietary development tools
- don’t license any RTOS but write everything from scratch
- ability to modify firmware
  - can be active or passive
  - can produce protocol violations

The OpenPCD project

The OpenPCD project

- various hardware interfaces
  - connector for analog and digital intermediate demodulation steps
  - connector for firmware-configurable trigger pulse
  - connector for unmodulated (tx) and demodulated (rx) bitstream
  - RS232 (@ 3.3V) port for debug messages
- versatile internal connection between ASIC and microcontroller
  - enables microcontroller to directly modulate carrier
    - using serial bitstream from SSC
    - using PWM signal from TC (timer/counter) unit
  - enables microcontroller to sample Tx and/or Rx signal
    - using SSC Rx

OpenPCD hardware configuration

OpenPCD hardware configuration

- Atmel AT91SAM7S128 microcontroller
  - 48MHz 32bit ARM7TDMI core
  - many integrated peripherals (SPI, SSC, ADC, I2C, ..)
  - USB full speed peripheral controller
  - 128kB user-programmable flash
  - 32kB SRAM
  - integrated SAM-BA emergency bootloader, enables ISP
- Philips CL RC632 reader ASIC
  - documentation ’freely’ available (40bit RC4 / 5days)
  - commonly used by other readers
  - supports 14443-A and B, including higher bitrates up to 424kBps
  - can be configured up to 848kBps, even though it’s not guaranteed

OpenPCD schematics

OpenPCD schematics

- Please see the schematics in PDF form

OpenPCD firmware build environment

OpenPCD firmware build environment

- Standard GNU toolchain for ARM7TDMI (armv4)
  - binutils-2.16.1
  - gcc-4.0.2
- Custom Makefiles to create flash images
- sam7utils for initial flash using SAM-BA
- ’cat dfu.bin firmware.bin > foo.samba’ produces SAM-BA image
- Parts of newlib are linked if DEBUG=1 is used (snprintf, ...)

OpenPCD device firmware

OpenPCD device firmware

○ since firmware is hackable, it should be easy to download a new image

○ USB Forum published "USB Device Firmware Upgrade" (DFU) specification

○ sam7dfu project (developed as part of OpenPCD) implements DFU on SAM7

○ dfu-programmer (sf.net) implemented 90% of what was required on host

○ DFU works by switching from normal (application) mode into separate mode with its own device/configuration/endpoint descriptors

○ since firmware bug could render device in broken ’crashed’ state, we added a button that can be pressed during power-on to force DFU mode

OpenPCD device firmware

OpenPCD device firmware

- The firmware build system allows for different build targets for different firmware images
- Normal reader operation using librfid supported by ’main_dumbreader’ target
- main_librfid: Intelligent firmware with full RFID stack built-in
- main_analog: Analog signals can be output on U.FL socket
- main_pwm: PWM modulation of 13.56MHz carrier (variable frequency/phase)
- main_reqa: Implement 14443-123 (Type A) in reader firmware, send REQA/WUPA/anticol

OpenPCD USB protocol

OpenPCD USB protocol

- All communication on the USB is done using a vendor-specific protocol on three endpoints (BULK OUT, BULK IN, INT IN)
- All messages (usb transfers) have a common four-byte header

main_dumbreader firmware

OpenPCD ’main_dumbreader’ firmware

- The main_dumbreader firmware exports four primitives for RC632 access
  - read register
  - write register
  - read fifo
  - write fifo
- Using those primitives, the full 14443-1234 A+B and 15693 can be implemented in host software (librfid)

OpenPCD host software (librfid)

The librfid project

- predates OpenPCD by 1.5 years
- was originally written as part of the OpenMRTD project for ePassports
- supported Omnikey CM5121 / CM5321 readers
- OpenPCD main_dumbreader support has been added
- implements 14443 -2, -3, -4 (A+B), ISO 15693, Mifare
- http://openmrtd.org/projects/librfid

OpenPCD status

OpenPCD status

- Hardware design finished
- Prototype state is over
- First 40 units shipped to customers
- Orders can be placed (100EUR excl. VAT) at http://shop.openpcd.org/
- DIY folks: We also sell the PCB for 18EUR :)
- We have five readers with us, in case anyone is interested

main_librfid firmware

OpenPCD ’main_librfid’ firmware

- The main_librfid firmware contains the full librfid stack
  - offers librfid C API
  - allows easy port of librfid host applications into device firmware
  - allows OpenPCD to operate 100% autonomous
  - does not have a USB protocol for host applications yet

OpenPCD outlook

OpenPCD outlook

- main_librfid USB protocol specifications
  - ’bset of both worlds’ approach for many applications
- emulate USB-CCID profile (designed for contact based smartcard readers)
  - thus, OpenPCD could be used to transparently access 14443-4 (T=CL) protocol cards just like contact based smartcards
- write nice frontend for Rx/Tx sampling
  - including software decoding on host pc to recover data
  - finally be able to do some cryptoanalysis on e.g. Mifare
- Lots of other interesting projects
  - Volunteers wanted!

The OpenPICC project

- conterpart to OpenPCD
- design RFID transponder simulator that gives full control / all interfaces
- hardware schematics and software licensed like OpenPCD
- based on the same microcontroller
  - much of the firmware (USB stack, SPI driver, ...) is shared
- no ASIC’s for ’transponder side’ available
- analog frontend and demodulator had to be built discrete, from scratch

OpenPICC hardware configuration

OpenPICC hardware configuration

- Atmel AT91SAM7S256
  - almost 100% identical to S128 (OpenPCD)
  - has twice the RAM and flash
- Analog antenna frontend / matching network
- Diode based demodulator
- Two FET and NAND based load modulation circuit
  - subcarrier generated in software
  - SSC clock rate == (2*fSubc) == 2*847.5kHz = 1.695MHz
  - Output of 101010 produces 847.5kHz subcarrier
  - two GPIO pins configure three steps of modulation depth

OpenPICC hardware (Rx path)

- OpenPICC hardware (Rx path)
  - Antenna builds resonant circuit with capacitor
  - low-capacity diode for demodulation
  - active filter + buffering/amplification
  - comparator for quantization of signal
  - resulting serial bitstream fed into SSC Rx of SAM7

OpenPICC hardware (Rx path)

OpenPICC hardware (Rx path)

- □ Problem: bit clock regeneration
  - ○ bitclock is fCarrier / 128
  - ○ PCD modulates 100% ASK => no continuous clock at PICC
- □ Solution:
  - ○ PICC needs to recover/recreate fCarrier using PLL
  - ○ PLL response can be delayed via low pass
- □ Problem:
  - ○ However, PLL will drift in long sequence of bytes
- □ Solution:
- □ ○ Sample-and-Hold in PLL loop can solve this problem

OpenPICC hardware (Rx path)

OpenPICC hardware (Rx path)

- □ Problem: bit clock / sample clock phase coherency
  - ○ bitclock is not coherent over multiple frames
  - ○ PCD can start bitclock at any fCarrier cycle
  - ○ PICC needs to recover bit clock
- □ Solution:
  - ○ OpenPICC uses SAM7 Timer/Counter 0 as fCarrier divider
  - ○ First falling edge of demodulated data resets counter
  - ○ Therefore, sample clock is in sync with bit clock

OpenPICC hardware (Tx path)

OpenPICC hardware (Tx path)

- □ Two FET and NAND based load modulation circuit
  - ○ subcarrier generated in software
  - ○ SSC clock rate == (2*fSubc) == 2*847.5kHz = 1.695MHz
  - ○ Output of 101010 produces 847.5kHz subcarrier
  - ○ two GPIO pins configure three steps of modulation depth

OpenPICC USB protocol

OpenPICC USB protocol

- 100% identical to OpenPCD, just different set of commands
- Most commands based on virtual register set (content: protocol params)
  - modulation width / depth
  - frame delay time for synchronous replies
  - encoding (manchester, OOK / NRZ-L, BPSK)
  - decoding (miller / NRZ)
  - UID for anticollision
  - ATQA content

OpenPICC status

OpenPICC status

- first prototype not yet 100% functional
- still some problems with clock recovery + analog side
- finished ’really soon now’ (december)
- first production units expected for January

Links

Links

- □http://openpcd.org/
- □http://wiki.openpcd.org/
- □http://shop.openpcd.org/
- □http://openmrtd.org/project/librfid/
- □http://openbeacon.org/ (active 2.4GHz RFID)