Further hacks on the Calypso platform

or How to turn a phone into a BTS

Sylvain Munaut

29C3, December 29th, 2012

About the speaker

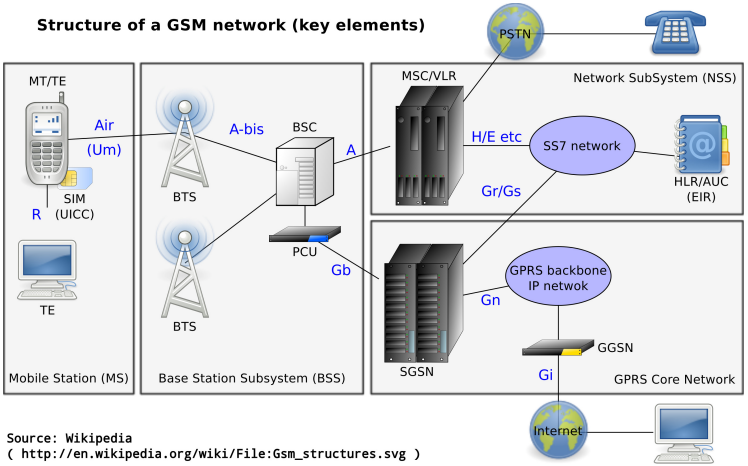
- Linux and free software "geek" since 1999
- M.Sc. in C.S. + some E.E.
- General orientation towards low level
  - Embedded, Kernel, Drivers and such.
  - Hardware (Digital stuff, FPGA, RF, ...)
- Interest in GSM projects for about 3 years
  - OpenBTS, OpenBSC, Airprobe, Osmocom-BB, ...
  - 27C3 GSM Intercept demo
  - Mostly in my spare time

Outline

The goal

- Can a phone act as the network ?
- Why ?
  - Mostly ... Just to see if we can
  - Cheap BTS for experimentation
  - $YOUR_IDEA
- Target hardware: C123
  - Osmocom-BB support
  - Classic TI Calypso design
    - Lots of alternatives platform if needed
    - Some leaked sources and documentation
  - Cheap and readily available

GSM
Network overview



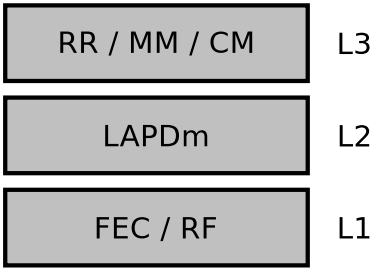
Source: Wikipedia
(http://en.wikipedia.org/wiki/File:Gsm_structures.svg)

Today, we'll focus on the air interface `Um`

GSM Um

Layer stackup

- Layer 3
  - "Higher" level logic
  - See GSM 04.{07,08,10,11}
- Layer 2
  - Data-Link layer
  - See GSM 04.06
- Layer 1
  - Physical layer
  - Channel coding and RF
  - See GSM 05.xx



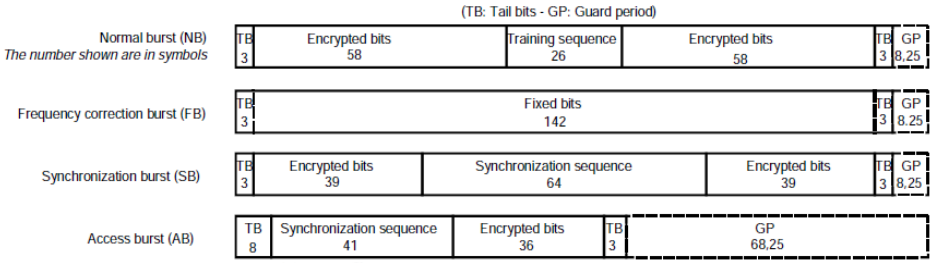
GSM Um
Frequencies

- Several bands
  - `GSM-850`, `EGSM-900`, `DCS1800`, `PCS1900`, ...
  - `http://en.wikipedia.org/wiki/GSM_frequency_bands`
- Frequency Division Duplex (FDD)
  - Downlink from Network to MS (e.g. DCS1800: 1710.2 to 1784.8 MHz)
  - Uplink, from MS to Network (e.g. DCS1800: 1805.2 to 1879.8 MHz)
- `ARFCN` = Absolute Radio-Frequency Channel Number
  - maps to a given frequency pair (UL/DL)
  - 200 kHz spacing
- Precision is critical
  - 0.1 ppm for pico-bts

GSM Um
Bursts

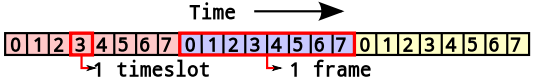
4 types of bursts:

- Normal bursts: Used to carry "real" data traffic
- Frequency correction bursts: Allow the MS to sync its clock and coarse TDMA
- Synchronization burst: Allow the MS to precisely sync to TDMA
- Access burst: Used by the MS to request a dedicated channel

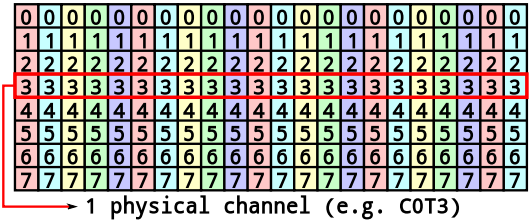


GSM Um
TDMA (1)

- Fully synchronous
- Described as a TDMA nightmare
- 1 frame = 8 timeslots



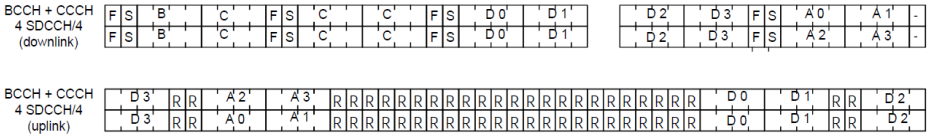
- Physical channel = 1 timeslot on 1 ARFCN



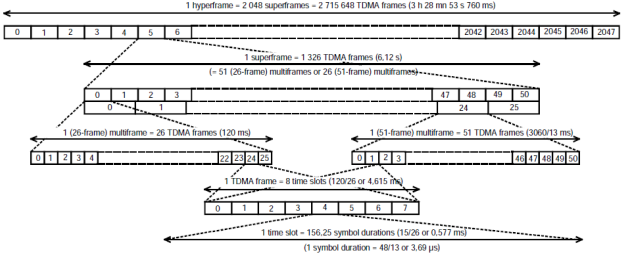
- Timeslots on uplink are delayed by 3 timeslots
  - Therefore phones don't need full duplex

GSM Um
TDMA (2)

- Each frame in multi-frame on a physical channel has a specific purpose, defining logical channels
  - e.g. for Combined BCCH+CCCH+SDCCH/4:

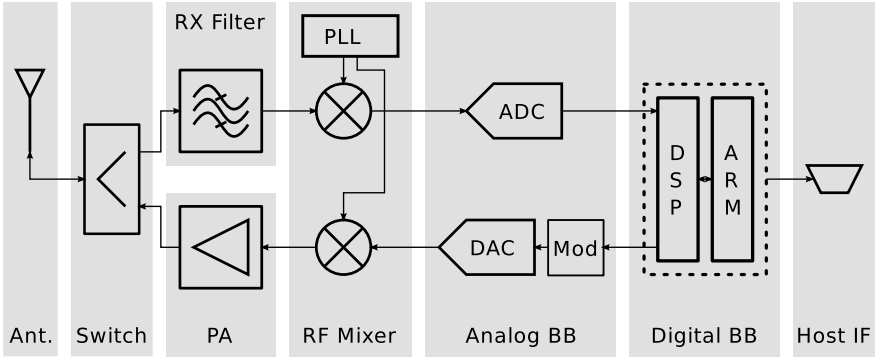


- When everything is put together :



Typical Calypso platform

Block diagram



Typical Calypso platform
Details

- Antenna
- RX/TX switch: Phones don't require full-duplex
- TX path:
  - Power Amplifier
  - Uplink RF mixer (Rita)
  - DAC (Iota)
  - Dedicated hardware GMSK modulation
- RX path:
  - RF SAW Filters: Block out-of-band signals
  - Downlink RF mixer (Rita)
  - ADC
  - No dedicated demodulation hardware. Done by SDR inside the DSP.
- Digital baseband (Calypso)
  - DSP: Mask ROM based L1 functions
  - ARM core: Already under our control with Osmocom-BB

Phone as BTS: Layer 2 and 3

- Role swapped
- Entirely software defined in the phone, and running on the ARM core
  - From Osmocom-BB we know we can change that easily
- Existing open-source stacks:
  - OpenBSC + Osmo-BTS
  - OpenBTS
- So, just re-use one of those !
- Currently, running them on the host (PC)

Phone as BTS: Layer 1

Channel coding

- Entirely implemented in phone DSP
  - ARM core can only send/receive L2 packets
  - No support for multiple channels at once
- What about the open-source stacks ?
  - OpenBSC + Osmo-BTS: Currently rely on closed hardware for this (nanoBTS / DSP in sysmoBTS)
  - OpenBTS: Rely on generic SDR hardware and so has it's own channel coding. Even better, it's already split into two applications:
    - `OpenBTS`: Main application implementing L1FEC/L2/L3 + external SIP
    - `transceiver`: TX and RX of the bursts from/to L1FEC via socket
- Make use of OpenBTS
  - Replace the `transceiver` binary with our own
  - No changes required on OpenBTS main application

Phone as BTS: Layer 1

RF

Things get interesting ...

- Duplex:
  - BTS transmit a continuous beacon to be detected
  - Phones can't do that
  - Either use multiple phones, or attempt half duplex operation
  - Timeslot layout: `Tt_R_ttt`
- Frequencies:
  - Phone usually TX on Uplink band and RX on Downlink band.
  - Some bands overlap:
    - GSM 850 downlink and E-GSM 900 uplink
    - DCS 1800 downlink and PCS 1900 uplink
  - Turns out the RF mixers can be driven out of spec anyway
- Timing:
  - A BTS is required to have very precise timing / frequency
  - Phone are around 20 ppm. BTS need to be less than 0.1 ppm !
  - We can lock the phone crystal to a nearby commerical cell

Phone as BTS: DSP

Analysis

- Mask-ROM based firmware
  - However lots of indirect calls / jump tables loaded in RAM
  - Can put new code in RAM and patch the jump table
  - Used to patch bugs in the ROM firmware
- Bootloader
  - Similar to the one in other TI chips (OMAP)
  - Shared RAM between ARM and DSP
  - For DSP boot we give the 'start' address
- Dump ROM
  - ROM can't be read from code executing from RAM
  - You can work around by using a memcpy from ROM
- Analyze
  - Long hours starring at IDA ...
  - Use interrupts as start points

Phone as BTS: DSP

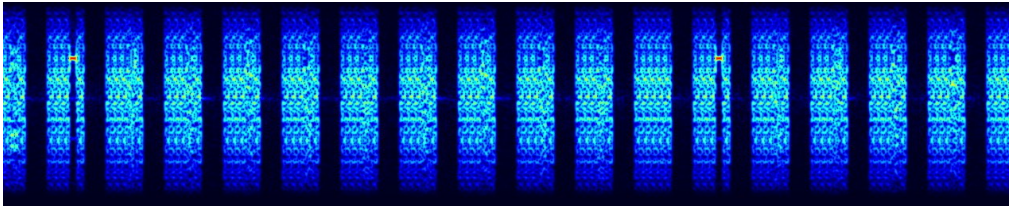
Extensions

- Support for Multislot TX
- Unfortunately Multislot RX is proving challenging
  - Back to back DMA is stubbornly refusing to work
- Transmit of special bursts `FCCH` and `SCH`
- Transmit of arbitrary normal bursts
- Receive of RACH bursts
  - Perform power detection on the phone
  - Send IQ data to the PC for demodulation

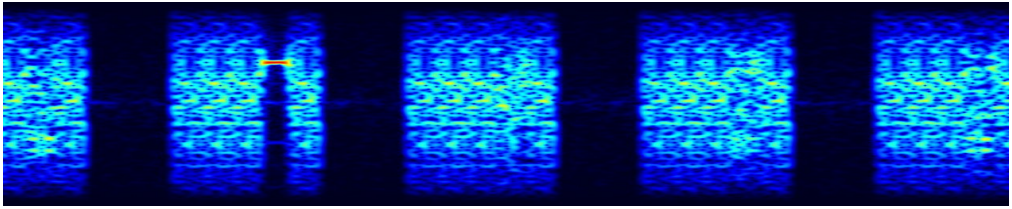
Of course all in hand-coded C54x assembly ...

Phone as BTS: Spectrum

Multiframe



Zoom



Demo

Murphy willing ...

Keep in mind:

- Proof-of-concept
- Non-compliant signal: Network detection is sometimes an issue

Availability

- Early 2013
  - Mostly need to write documentation
  - And split the patch into commits
- Proof-of-concept targetted at developpers
  - Might not work for you in your environment
  - Debug can require expensive RF gear
  - If you can't make the classic Osmocom-BB work, don't try this
- Get a test license !
  - Not that hard / expensive
  - This is restricted spectrum, act responsibly

Summary

- It is possible to make a phone as a BTS
  - Kind of
- Devices are often way more capable than what they were designed for
- Reverse engineering is fun

Future work

- Implement OpenBSC / Osmo-BTS interoperability
- Improve reliability
- Multiphone operation
- Power control
- Multi-slot RX
- ...

Thanks

Thanks to anyone contributing to the various Open Source GSM projects. For this project in particular:

- Harald Welte
- Dieter Spaar
- David Burgess and his team at Range Networks

and of course thanks to the 29C3 team for having me.

Further reading

Airprobe	http://airprobe.org/
OsmocomBB	http://bb.osmocom.org/
OpenBSC	http://openbsc.osmocom.org/
OpenBTS	http://openbts.sourceforge.net/
GSM Specs	http://webapp.etsi.org/key/queryform.asp