# Introduction to the OsmocomDECT stack

OsmoDevCon2013

4th-8th April 2013

Berlin, Germany

Patrick McHardy <kaber@trash.net>

http://dect.osmocom.org

- DECT stack implementing physical layer, MAC layer, Data Link control layer, Network layer and Interworking unit
- Supports FP (base station) and PP (portable part) modes
- Physical Layer implemented through driver for sc1442x baseband chipsets, including open source firmware and generic transceiver layer
- MAC layers (cell site, cluster control), Data Link control contained in kernel
- Network layer implemented as userspace library

## Drivers: drivers/dect

- Drivers interact with baseband processor and radio
  - Radio programming
  - Baseband programming (runtime firmware patching)
  - Frame reception and transmission
  - Time keeping
  - Ciphering offloading
- Received frames for 1-6 timeslots and current time are encapsulated in "dect_transceiver_event" structure and queued to generic transceiver layer

osmocomDECT

## sc1442x driver: drivers/dect/coa

- Implements support for sc14421/24 basebands
  - sc14421: ComOnAir PCMCIA cards
  - sc14424: ComOnAir PCI cards
  - Features:
    - Cipher offloading
    - Checksum offloading
    - Wideband audio
  - Open source firmware assembled during kernel build
  - "radio_ops" for different radio types

sc1442x driver: drivers/dect/coa

- Baseband processor:
  - Executes one instruction per DECT symbol
  - Call stack of depth 3
  - Synchonization instructions: WT, WNT, EN_SL_ADJ
  - Transmission and reception: B_SR/B_ST, B_AR/B_AT, B_BR/B_BT, B_BRFU/B_BTFU, ...
  - Ciphering: D_LDK/D_PREP, D_LDS/D_WRS
  - Control PINs: P_LD, P_LDL, P_LDH
  - Microwire transmission (radio settings): MEN1N, MEN1, M_WR

## Radios:

- U2785 ATMEL RF IC:
  - PCI and Type II PCMCIA cards
  - "Slow-hopping" radio: needs one timeslot for channel switching
  - Dynamic mapping of DECT bands to divisor/swallow count settings
- LMX3161 NSC Single Chip Radio Transceiver:
  - Type III PCMCIA cards
  - Not supported yet, work is ongoing

## Transceiver layer: net/dect/transceiver.c

- Handling of "transceiver groups": multiple synchronized transceivers
  - Synchronization of secondary transceivers
  - Dequeues events from all transceivers in a group
  - Events are sorted chronologically
  - Virtual clock maintenance
  - Queueing of reordered events to MAC cell site layer
  - Clock replay to MAC cell site layer

## Transceiver layer: net/dect/transceiver.c

- Netlink userspace API:
  - Notification about new/removed transceivers
  - Transceiver configuration
  - Attachment/detachment to/from cells
  - Band configuration
  - Status information
  - Statistics

# Transceiver layer: net/dect/transceiver.c

```
# dect-transceiver-list --name trx9
DECT Transceiver trx9@cell0:
        Type: sc1442x
        RF-band: 00000
        Events: busy: 0 late: 2587
        slot 0: <tx> carrier: 2 (1893.888 MHz)
            RX: bytes 320 packets 40 a-crc-errors 1 x-crc-errors 0 z-crc-errors 0
            TX: bytes 1776 packets 37
        slot 2: <idle> carrier: 0 (1897.344 MHz)
            RX: bytes 0 packets 0 a-crc-errors 0 x-crc-errors 0 z-crc-errors 0
            TX: bytes 0 packets 0
        [...]
        slot 10: <rx,sync> carrier: 9 (1881.792 MHz +0.569 kHz) signal level: -41.94dBm
            RX: bytes 2600 packets 325 a-crc-errors 0 x-crc-errors 0 z-crc-errors 0
            TX: bytes 0 packets 0
        slot 12: <rx> carrier: 2 (1893.888 MHz +0.083 kHz) signal level: -57.47dBm
            RX: bytes 1764 packets 36 a-crc-errors 0 x-crc-errors 0 z-crc-errors 0
            TX: bytes 0 packets 0
        slot 14: <idle> carrier: 0 (1897.344 MHz)
            RX: bytes 0 packets 0 a-crc-errors 0 x-crc-errors 0 z-crc-errors 0
            TX: bytes 0 packets 0
        [...]
```

## MAC layer overview

- MAC layer
  - CSF (Cell site Functions)
  - CCF (Cluster Control functions)
  - Communication between layers either through handles
  - Either direct function calls or network protocol
  - Network protocol unfinished
  - Transparent

## MAC cell site functions (CSF): net/dect/mac_csf.c

- Maintenance tasks:
  - Transceiver group maintenance (bind/unbind)
  - Frame timer synchronization and maintenance
  - Channel list maintenance (periodic scanning and quality control)
  - Channel selection based on channel lists
  - Transceiver selection
  - Bearer enablement timing
  - Bearer quality control

## MAC cell site functions (CSF): net/dect/mac_csf.c

- Idle receiver control (IRC):
  - Locking to FPs (PP-side only)
  - Secondary transceiver synchronization
  - Periodic channel scanning
  - Channel hopping (receiver channel scanning sequence)
  - Reception of MAC connection requests (usually FP-side only)

## MAC cell site functions (CSF): net/dect/mac_csf.c

- Dummy bearer control (DBC):
  - FP-side only
  - Broadcast bearer
  - Cell identity
  - Timing information
  - Cell capabilities
  - Paging

## MAC cell site functions (CSF): net/dect/mac_csf.c

- Traffic bearer control (TBC):
  - Bi-directional traffic bearer setup and management
  - Muxing/Demuxing of higher layer data and MAC layer information
- Monitor Bearer control (DMB):
  - Used for sniffing
  - Follows FP channel hopping sequence
  - Locks to new MAC connections
  - Passes frames up to AF_DECT raw sockets

## MAC cell site functions (CSF): net/dect/mac_csf.c

- Netlink userspace API:
  - Cell site configuration
  - Binding of cells to clusters
  - Reporting of scan results
  - Status information

## MAC cluster control functions (CCF): net/dect/mac_csf.c

- Maintenance tasks:
  - Cluster MAC layer frame timers
  - Cell site MAC layer configuration
- Broadcast message control (BMC):
  - Dispatch of paging messages to cell site functions (FP-side only)
  - Reception of paging messages from cell site functions (PP-side only)

## MAC cluster control functions (CCF): net/dect/mac_csf.c

- Multi-Bearer control (MBC):
  - Maintains multiple cell-site traffic bearers to form a multi bearer
  - Cipher management of traffic bearers
  - Hand-over
  - Higher layer data distribution to traffic bearers
  - Reception of higher layer data from cell site function
  - Removal of redundant data

## MAC cluster control functions (CCF): net/dect/mac_csf.c

- Netlink userspace API
  - Cluster configuration:
    - Identities
    - Mode,
    - Access rights information
  - MBC status information
    - Identity
    - Service type
    - MAC bearers
    - Cell site information
    - Byte/packet counters
    - Handover attempts
    - Time slots

## Data Link Control (DLC): net/dect/dlc.c

- Routing
  - Routing of C-Plane and U-Plane data to MAC connections
- Logical MAC connection maintenance
  - Multi Bearer setup
  - Multi Bearer handover
  - Passing of C-Plane and U-Plane data between higher and lower layers
  - Connection modification according to higher layer demands

## Data Link Control C-Plane (DLC): net/dect/dlc_cplane.c

- Paging
  - Passing of paging message to higher layer SAP
- Lc entity
  - C-Plane data fragmentation and reassembly
  - Checksumming
  - Instantiating of LAPC entities on connection requests

## Data Link Control C-Plane (DLC): net/dect/dlc_cplane.c

- LAPC
  - Similar to LAPD, LAPDm, ...
  - Unacknowledged point-to-point/broadcast communication
  - Point-to-point class A communication (window size = 1)
  - Point-to-point class B communication (window size = 8), suspend/resume
  - Segmentation of messages

## Data Link control C-Plane SAP: net/dect/dlc_s_sap.c, net/dect/dlc_b_sap:

- S-SAP socket API:
  - Socket interface to LAPC
  - send/recv/...
  - Ciphering API (get/setsockopt)
  - MAC connection attributes API (get/setsockopt)

- B-SAP socket API:
  - Socket interface to paging
  - send/recv/...
  - Duplicating received pages to all listeners
  - Page attributes specified through CMSG

## Data Link control U-Place: net/dect/dlc_uplane.c, dlc_lu1_sap.c:

- Generic U-Plane:
  - Framing (FBx entities)
  - Frame formats (LUx entities)
- LU1 SAP:
  - TRansparent UnProtected Service (TRUP)
  - Socket interface for Audio
  - Audio: min_delay service
  - Seamless Handover: frame offset advances depending on time slot

osmocomDECT

## Network layer: libdect

- libdect overview:
  - LCE (Link Control Entity), roughly comparable to GSM48 RR
  - MM (Mobility Management)
  - CC (Call Control)
  - SS (Supplementary services)
  - CLMS (Connectionless messaging service)
  - LLME (Lower layer management entity)
  - Link and transaction management
  - Message/TLV encoding/decoding
  - Message routing

## Network layer: libdect

- libdect Overview:
  - User registers one or more ops structures: lce_ops, mm_ops, cc_ops, ...
  - Callbacks for indication and confirmation primitives
  - Functions for request and result primitives
  - Encapsulated parameter structures, reference counted parameters and IEs
  - Support functions for authentication, SS, debugging, ...

## Network layer: libdect

```
/** MM_ACCESS_RIGHTS primitive parameters. */
struct dect_mm_access_rights_param {
	struct dect_ie_collection		common;
	struct dect_ie_portable_identity	*portable_identity;
	struct dect_ie_list			fixed_identity;
	struct dect_ie_location_area		*location_area;
	struct dect_ie_auth_type		*auth_type;
	struct dect_ie_cipher_info		*cipher_info;
	struct dect_ie_zap_field		*zap_field;
	struct dect_ie_setup_capability		*setup_capability;
	struct dect_ie_terminal_capability	*terminal_capability;
	struct dect_ie_service_class		*service_class;
	struct dect_ie_model_identifier		*model_identifier;
	struct dect_ie_reject_reason		*reject_reason;
	struct dect_ie_duration			*duration;
	struct dect_ie_iwu_to_iwu		*iwu_to_iwu;
	struct dect_ie_escape_to_proprietary	*escape_to_proprietary;
	struct dect_ie_codec_list		*codec_list;
};
```

## Network layer: libdect

```
struct dect_mm_ops {
        size_t  priv_size;
        /**< Size of the private storage area of an MM endpoint */
        void    (*mm_access_rights_ind)(struct dect_handle *dh,
                                struct dect_mm_endpoint *mme,
                                struct dect_mm_access_rights_param *param);
        /**< MM_ACCESS_RIGHTS-ind primitive */
        void    (*mm_access_rights_cfm)(struct dect_handle *dh,
                                struct dect_mm_endpoint *mme, bool accept,
                                struct dect_mm_access_rights_param *param);
        /**< MM_ACCESS_RIGHTS-cfm primitive */
        ...
};

extern int dect_mm_access_rights_req(struct dect_handle *dh, struct dect_mm_endpoint *mme,
                                const struct dect_mm_access_rights_param *param);
extern void dect_mm_access_rights_res(struct dect_handle *dh, struct dect_mm_endpoint *mme,
                                bool accept, const struct dect_mm_access_rights_param *param);
```

## Network layer: libdect

```
static DECT_SFMT_MSG_DESC(mm_access_rights_request,
    DECT_SFMT_IE(DECT_IE_PORTABLE_IDENTITY,       IE_NONE,    IE_MANDATORY, 0),
    DECT_SFMT_IE(DECT_IE_AUTH_TYPE,               IE_NONE,    IE_OPTIONAL,  0),
    DECT_SFMT_IE(DECT_IE_CIPHER_INFO,             IE_NONE,    IE_OPTIONAL,  0),
    DECT_SFMT_IE(DECT_IE_SETUP_CAPABILITY,        IE_NONE,    IE_OPTIONAL,  0),
    DECT_SFMT_IE(DECT_IE_TERMINAL_CAPABILITY,     IE_NONE,    IE_OPTIONAL,  0),
    DECT_SFMT_IE(DECT_IE_IWU_TO_IWU,              IE_NONE,    IE_OPTIONAL,  0),
    DECT_SFMT_IE(DECT_IE_MODEL_IDENTIFIER,        IE_NONE,    IE_OPTIONAL,  0),
    DECT_SFMT_IE(DECT_IE_ESCAPE_TO_PROPRIETARY,   IE_NONE,    IE_OPTIONAL,  0),
    DECT_SFMT_IE(DECT_IE_CODEC_LIST,              IE_NONE,    IE_OPTIONAL,  0),
    DECT_SFMT_IE_END_MSG
);
```

## Network layer: libdect

```
NWK: 05 42 0b 02 01 88 0c 08 1b 42 27 01 4c 5c 44 84     |.B.......B'.L\D.|
NWK: 0e 08 9e 01 7e 0c 42 ae ec ff                       |....~.B...|
{MM-KEY-ALLOCATE} message:
  IE: <<ALLOCATION-TYPE>> id: b len: 4 dst: 0xfcf440
        authentication algorithm: DSAA (1)
        authentication key number: 8
        authentication code number: 8
  IE: <<RAND>> id: c len: 10 dst: 0xfcf460
        value: 84445c4c0127421b
  IE: <<RS>> id: e len: 10 dst: 0xfcf480
        value: ffecae420c7e019e

NWK: 85 40 0a 03 01 48 00 0c 08 de a7 66 4d 34 fb c2     |.@...H.....fM4..|
NWK: 7f 0d 04 85 6a 5f 9e                                |....j_.|
{MM-AUTHENTICATION-REQUEST} message:
  IE: <<AUTH-TYPE>> id: a len: 5 dst: 0xfcf5e0
        authentication algorithm: DSAA (1)
        authentication key type: Authentication code (4)
        authentication key number: 8
        cipher key number: 0
        INC: 0 DEF: 0 TXC: 0 UPC: 0
  IE: <<RAND>> id: c len: 10 dst: 0xfcf600
```

osmocomDECT

## Network layer Link Control Entity: src/lce.c

- Link maintenance
  - Paging
  - Direct (PP initiated) and indirect (paged) link setup
  - Link attribute modification
  - Cipher management in coordination with MM

## Mobility Management: src/mm.c

- Access rights procedures
  - Pairing
  - Capability exchange
  - Usually coupled with UAK key allocation
  - Access rights revocation
- Key allocation procedure
  - Allocates UAK
  - Derived from AC (Authentication Code)

## Mobility Management: src/mm.c

- Authentication procedure
  - Optional mutual authentication, usually PP only or even none
  - Seperate procedure or integrated into key allocation
  - UAK or UPI (User personal Identity)
  - Session key derivation

- Ciphering procedure
  - Ciphering with either SDK or DCK
  - Always initiated by PP, FP may suggest ciphering to PP

## Mobility Management: src/mm.c

- Location procedures
  - Informes FP of PP location (cell, cluster)
  - Periodic or after location area change
  - Capability exchange
  - TPUI allocation
  - Detach
- Other
  - Identity procedurs
  - External protocol information procedures

## Call Control; src/cc.c

- Call procedures
  - Call setup, modification, termination, ..
  - Codec negotiation
  - Call related supplementary services (CRSS)
  - U-Plane setup and maintenance

## Connectionless messaging service: src/clms.c

- Connectionless packet service

## Interworking Unit: asterisk, channels/chan_dect.c

- Asterisk Channel driver
  - Interacts with libdect
  - Supports access rights, key allocation, authentication, chiphering, ...
  - Asterisk DB used for storing subscription data
  - Narrow-band audio, wide-band unfinished

- libnl-dect:
  - Netlink API for configuration and notifications
  - Example tools used for configuration

- dectmon:
  - DECT protocol decoder using raw sockets
  - Multiple transceiver support
  - Protocol decoding
  - Decryption, life audio
  - Interactive command line interface
  - Can interact with monitored FPs

osmocomDECT

- libpcap
  - libpcap with DECT raw socket support
- ASL
  - ASL macro assembler
  - Used for firmware assembly
  - Patched version with support for modern chipsets (SC1445x/8x)
- Disassembler
  - Firmware disassembler
  - Unreleased so far

- Finishing wideband support
- CoA Type III support
- GAP/DECT-NG profile compliance
- S1445x SoC support
- DVB-T SDR RX support