# sysmocom

sysmocom - s.f.m.c. GmbH



# OsmoTRX User Manual

by Pau Espin Pedrol

DRAFT

Copyright © 2018 sysmocom - s.f.m.c. GmbH

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

The Asciidoc source code of this manual can be found at http://git.osmocom.org/osmo-gsm-manuals/

| HISTORY | | | |
|---|---|---|---|
| NUMBER | DATE | DESCRIPTION | NAME |
| 1 | March 6, 2019 | Initial version. | PE |

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

# Contents

**1 Foreword** **1**
1.1 Acknowledgements . . . 1
1.2 Endorsements . . . 2

**2 Preface** **2**
2.1 FOSS lives by contribution! . . . 2
2.2 Osmocom and sysmocom . . . 3
2.3 Corrections . . . 3
2.4 Legal disclaimers . . . 3
2.4.1 Spectrum License . . . 3
2.4.2 Software License . . . 3
2.4.3 Trademarks . . . 3
2.4.4 Liability . . . 4
2.4.5 Documentation License . . . 4

**3 Introduction** **4**
3.1 Required Skills . . . 4
3.2 Getting assistance . . . 5

**4 Overview** **5**
4.1 About OsmoTRX . . . 5

**5 Running OsmoTRX** **6**
5.1 SYNOPSIS . . . 6
5.2 OPTIONS . . . 6

**6 Osmocom Control Interface** **6**
6.1 Control Interface Protocol . . . 6
6.1.1 GET operation . . . 7
6.1.2 SET operation . . . 8
6.1.3 TRAP operation . . . 8
6.2 Common variables . . . 8
6.3 Control Interface python examples . . . 9
6.3.1 Getting rate counters . . . 9
6.3.2 Setting a value . . . 9
6.3.3 Getting a value . . . 10
6.3.4 Listening for traps . . . 10

**7 Control interface** **10**

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

**8 The Osmocom VTY Interface** **10**
8.1 Accessing the telnet VTY . . . 11
8.2 VTY Nodes . . . 11
8.3 Interactive help . . . 12
8.3.1 The question-mark (?) command . . . 12
8.3.2 TAB completion . . . 13
8.3.3 The list command . . . 14
8.3.4 The attribute system . . . 15
8.3.5 The expert mode . . . 17

**9 libosmocore Logging System** **17**
9.1 Log categories . . . 18
9.2 Log levels . . . 18
9.3 Log printing options . . . 18
9.4 Log filters . . . 19
9.5 Log targets . . . 19
9.5.1 Logging to the VTY . . . 19
9.5.2 Logging to the ring buffer . . . 20
9.5.3 Logging via gsmtap . . . 20
9.5.4 Logging to a file . . . 22
9.5.5 Logging to syslog . . . 22
9.5.6 Logging to systemd-journal . . . 23
9.5.7 Logging to stderr . . . 25

**10 Counters** **25**
10.1 Rate Counters . . . 25

**11 Osmo Stat Items** **25**

**12 Osmo Counters** **25**
12.1 Rate Counter Configurable Error Thresholds . . . 25

**13 Configuring OsmoTRX** **26**
13.1 Documented example . . . 26
13.2 Multi-ARFCN mode . . . 27

**14 OsmoTRX hardware architecture support** **28**

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

**15 OsmoTRX hardware device support** **28**
15.1 Ettus USRP1 . . . 28
15.1.1 USRP1 in-band USB protocol . . . 28
15.1.1.1 "Data Channel" payload format . . . 30
15.1.1.2 "Control Channel" payload format . . . 30
15.1.1.3 General sub-packet format . . . 30
15.1.1.4 Specific sub-packet formats . . . 30
15.2 Ettus B200 . . . 33
15.3 Ettus B210 . . . 33
15.4 LimeSDR-USB . . . 34
15.5 LimeSDR-mini . . . 34

**16 OsmoTRX backend support** **34**
16.1 `osmo-trx-uhd` for UHD based Transceivers . . . 34
16.2 `osmo-trx-lms` for LimeSuite based Transceivers . . . 34
16.3 `osmo-trx-usrp1` for libusrp based Transceivers . . . 35
16.4 `osmo-trx-ipc` Inter Process Communication backend . . . 35
16.4.1 ipc-device-test . . . 36

**17 Code Architecture** **37**
17.1 Transceiver . . . 37
17.2 RadioInterface . . . 38
17.2.1 RadioInterfaceResamp . . . 38
17.2.2 RadioInterfaceMulti . . . 38
17.3 RadioDevice . . . 39

**18 VTY Process and Thread management** **39**
18.1 Scheduling Policy . . . 39
18.2 CPU-Affinity Mask . . . 39

**19 TRX Manager UDP socket interface** **41**
19.1 Indications on the Master Clock Interface . . . 41
19.2 TRXC protocol . . . 41
19.2.1 Power Control . . . 42
19.2.2 Tuning Control . . . 42
19.2.3 Training Sequence configuration . . . 43
19.2.4 Timeslot Control . . . 43
19.2.4.1 Multiple Training Sequences (optional) . . . 43
19.2.4.2 VAMOS enabled channel combinations (optional) . . . 44
19.2.5 TRXD header version negotiation . . . 46

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

19.3 TRXD protocol . . . 46
19.3.1 PDU versioning . . . 47
19.3.2 Uplink PDU format . . . 47
19.3.2.1 Coding of MTS: Modulation and Training Sequence info . . . 50
19.3.3 Downlink Data Burst . . . 51
19.3.4 PDU batching . . . 52
19.3.5 Coding of VAMOS PDUs . . . 53

**20 osmo-trx-ipc IPC Interface** **54**
20.1 Control plane . . . 54
20.2 Master UD socket . . . 54
20.3 Channel UD Socket . . . 56
20.4 Data Plane . . . 56
20.4.1 Posix Shared Memory structure . . . 56
20.4.2 Posix Shared Memory format . . . 57
20.4.3 Posix Shared Memory procedures . . . 57

**21 Glossary** **58**

**A Osmocom TCP/UDP Port Numbers** **66**

**B Bibliography / References** **68**
B.0.0.0.1 References . . . 68

**C GNU Free Documentation License** **72**
C.1 PREAMBLE . . . 72
C.2 APPLICABILITY AND DEFINITIONS . . . 72
C.3 VERBATIM COPYING . . . 73
C.4 COPYING IN QUANTITY . . . 73
C.5 MODIFICATIONS . . . 74
C.6 COMBINING DOCUMENTS . . . 75
C.7 COLLECTIONS OF DOCUMENTS . . . 75
C.8 AGGREGATION WITH INDEPENDENT WORKS . . . 75
C.9 TRANSLATION . . . 76
C.10 TERMINATION . . . 76
C.11 FUTURE REVISIONS OF THIS LICENSE . . . 76
C.12 RELICENSING . . . 76
C.13 ADDENDUM: How to use this License for your documents . . . 77

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

# 1 Foreword

Digital cellular networks based on the GSM specification were designed in the late 1980s and first deployed in the early 1990s in Europe. Over the last 25 years, hundreds of networks were established globally and billions of subscribers have joined the associated networks.

The technological foundation of GSM was based on multi-vendor interoperable standards, first created by government bodies within CEPT, then handed over to ETSI, and now in the hands of 3GPP. Nevertheless, for the first 17 years of GSM technology, the associated protocol stacks and network elements have only existed in proprietary *black-box* implementations and not as Free Software.

In 2008 Dieter Spaar and I started to experiment with inexpensive end-of-life surplus Siemens GSM BTSs. We learned about the A-bis protocol specifications, reviewed protocol traces and started to implement the BSC-side of the A-bis protocol as something originally called `bs11-abis`. All of this was *just for fun*, in order to learn more and to boldly go where no Free Software developer has gone before. The goal was to learn and to bring Free Software into a domain that despite its ubiquity, had not yet seen any Free / Open Source software implementations.

`bs11-abis` quickly turned into `bsc-hack`, then *OpenBSC* and its *OsmoNITB* variant: A minimal implementation of all the required functionality of an entire GSM network, exposing A-bis towards the BTS. The project attracted more interested developers, and surprisingly quickly also commercial interest, contribution and adoption. This allowed adding support for more BTS models.

After having implemented the network-side GSM protocol stack in 2008 and 2009, in 2010 the same group of people set out to create a telephone-side implementation of the GSM protocol stack. This established the creation of the Osmocom umbrella project, under which OpenBSC and the OsmocomBB projects were hosted.

Meanwhile, more interesting telecom standards were discovered and implemented, including TETRA professional mobile radio, DECT cordless telephony, GMR satellite telephony, some SDR hardware, a SIM card protocol tracer and many others.

Increasing commercial interest particularly in the BSS and core network components has lead the way to 3G support in Osmocom, as well as the split of the minimal *OsmoNITB* implementation into separate and fully featured network components: OsmoBSC, OsmoMSC, OsmoHLR, OsmoMGW and OsmoSTP (among others), which allow seamless scaling from a simple "Network In The Box" to a distributed installation for serious load.

It has been a most exciting ride during the last eight-odd years. I would not have wanted to miss it under any circumstances.

— Harald Welte, Osmocom.org and OpenBSC founder, December 2017.

## 1.1 Acknowledgements

My deep thanks to everyone who has contributed to Osmocom. The list of contributors is too long to mention here, but I'd like to call out the following key individuals and organizations, in no particular order:

- Dieter Spaar for being the most amazing reverse engineer I've met in my career
- Holger Freyther for his many code contributions and for shouldering a lot of the maintenance work, setting up Jenkins - and being crazy enough to co-start sysmocom as a company with me ;)
- Andreas Eversberg for taking care of Layer2 and Layer3 of OsmocomBB, and for his work on OsmoBTS and OsmoPCU
- Sylvain Munaut for always tackling the hardest problems, particularly when it comes closer to the physical layer
- Chaos Computer Club for providing us a chance to run real-world deployments with tens of thousands of subscribers every year
- Bernd Schneider of Netzing AG for funding early ip.access nanoBTS support
- On-Waves ehf for being one of the early adopters of OpenBSC and funding a never ending list of features, fixes and general improvement of pretty much all of our GSM network element implementations
- sysmocom, for hosting and funding a lot of Osmocom development, the annual Osmocom Developer Conference and releasing this manual.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

- Jan Luebbe, Stefan Schmidt, Daniel Willmann, Pablo Neira, Nico Golde, Kevin Redon, Ingo Albrecht, Alexander Huemer, Alexander Chemeris, Max Suraev, Tobias Engel, Jacob Erlbeck, Ivan Kluchnikov
- NLnet Foundation, for providing funding for a number of individual work items within the Osmocom universe, such as LTE support in OsmoCBC or GPRS/EGPRS support for Ericsson RBS6000.
- WaveMobile Ltd, for many years of sponsoring.

May the source be with you!

— Harald Welte, Osmocom.org and OpenBSC founder, January 2016.

### 1.2 Endorsements

This version of the manual is endorsed by Harald Welte as the official version of the manual.

While the GFDL license (see Appendix C) permits anyone to create and distribute modified versions of this manual, such modified versions must remove the above endorsement.

## 2 Preface

First of all, we appreciate your interest in Osmocom software.

Osmocom is a Free and Open Source Software (FOSS) community that develops and maintains a variety of software (and partially also hardware) projects related to mobile communications.

Founded by people with decades of experience in community-driven FOSS projects like the Linux kernel, this community is built on a strong belief in FOSS methodology, open standards and vendor neutrality.

### 2.1 FOSS lives by contribution!

If you are new to FOSS, please try to understand that this development model is not primarily about "free of cost to the GSM network operator", but it is about a collaborative, open development model. It is about sharing ideas and code, but also about sharing the effort of software development and maintenance.

If your organization is benefiting from using Osmocom software, please consider ways how you can contribute back to that community. Such contributions can be many-fold, for example

- sharing your experience about using the software on the public mailing lists, helping to establish best practises in using/operating it,
- providing qualified bug reports, workarounds
- sharing any modifications to the software you may have made, whether bug fixes or new features, even experimental ones
- providing review of patches
- testing new versions of the related software, either in its current "master" branch or even more experimental feature branches
- sharing your part of the maintenance and/or development work, either by donating developer resources or by (partially) funding those people in the community who do.

We're looking forward to receiving your contributions.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

## 2.2 Osmocom and sysmocom

Some of the founders of the Osmocom project have established *sysmocom - systems for mobile communications GmbH* as a company to provide products and services related to Osmocom.

sysmocom and its staff have contributed by far the largest part of development and maintenance to the Osmocom mobile network infrastructure projects.

As part of this work, sysmocom has also created the manual you are reading.

At sysmocom, we draw a clear line between what is the Osmocom FOSS project, and what is sysmocom as a commercial entity. Under no circumstances does participation in the FOSS projects require any commercial relationship with sysmocom as a company.

## 2.3 Corrections

We have prepared this manual in the hope that it will guide you through the process of installing, configuring and debugging your deployment of cellular network infrastructure elements using Osmocom software. If you do find errors, typos and/or omissions, or have any suggestions on missing topics, please do take the extra time and let us know.

## 2.4 Legal disclaimers

### 2.4.1 Spectrum License

As GSM and UMTS operate in licensed spectrum, please always double-check that you have all required licenses and that you do not transmit on any ARFCN or UARFCN that is not explicitly allocated to you by the applicable regulatory authority in your country.



**Warning**
Depending on your jurisdiction, operating a radio transmitter without a proper license may be considered a felony under criminal law!

### 2.4.2 Software License

The software developed by the Osmocom project and described in this manual is Free / Open Source Software (FOSS) and subject to so-called *copyleft* licensing.

Copyleft licensing is a legal instrument to ensure that this software and any modifications, extensions or derivative versions will always be publicly available to anyone, for any purpose, under the same terms as the original program as developed by Osmocom.

This means that you are free to use the software for whatever purpose, make copies and distribute them - just as long as you ensure to always provide/release the *complete and corresponding* source code.

Every Osmocom software includes a file called `COPYING` in its source code repository which explains the details of the license. The majority of programs is released under GNU Affero General Public License, Version 3 (AGPLv3).

If you have any questions about licensing, don't hesitate to contact the Osmocom community. We're more than happy to clarify if your intended use case is compliant with the software licenses.

### 2.4.3 Trademarks

All trademarks, service marks, trade names, trade dress, product names and logos appearing in this manual are the property of their respective owners. All rights not expressly granted herein are reserved.

For your convenience we have listed below some of the registered trademarks referenced herein. This is not a definitive or complete list of the trademarks used.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

*Osmocom*® and *OpenBSC*® are registered trademarks of Holger Freyther and Harald Welte.

*sysmocom*® and *sysmoBTS*® are registered trademarks of *sysmocom - systems for mobile communications GmbH*.

*ip.access*® and *nanoBTS*® are registered trademarks of *ip.access Ltd.*

#### 2.4.4 Liability

The software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the License text included with the software for more details.

#### 2.4.5 Documentation License

Please see Appendix C for further information.

# 3 Introduction

## 3.1 Required Skills

Please note that even while the capital expenses of running mobile networks has decreased significantly due to Osmocom software and associated hardware like sysmoBTS, GSM networks are still primarily operated by large GSM operators.

Neither the GSM specification nor the GSM equipment was ever designed for networks to be installed and configured by anyone but professional GSM engineers, specialized in their respective area like radio planning, radio access network, back-haul or core network.

If you do not share an existing background in GSM network architecture and GSM protocols, correctly installing, configuring and optimizing your GSM network will be tough, irrespective whether you use products with Osmocom software or those of traditional telecom suppliers.

GSM knowledge has many different fields, from radio planning through site installation to core network configuration/administration.

The detailed skills required will depend on the type of installation and/or deployment that you are planning, as well as its associated network architecture. A small laboratory deployment for research at a university is something else than a rural network for a given village with a handful of cells, which is again entirely different from an urban network in a dense city.

Some of the useful skills we recommend are:

- general understanding about RF propagation and path loss in order to estimate coverage of your cells and do RF network planning.
- general understanding about GSM network architecture, its network elements and key transactions on the Layer 3 protocol
- general understanding about voice telephony, particularly those of ISDN heritage (Q.931 call control)
- understanding of GNU/Linux system administration and working on the shell
- understanding of TCP/IP networks and network administration, including tcpdump, tshark, wireshark protocol analyzers.
- ability to work with text based configuration files and command-line based interfaces such as the VTY of the Osmocom network elements

### 3.2 Getting assistance

If you do have a support package / contract with sysmocom (or want to get one), please contact support@sysmocom.de with any issues you may have.

If you don't have a support package / contract, you have the option of using the resources put together by the Osmocom community at https://projects.osmocom.org/, checking out the wiki and the mailing-list for community-based assistance. Please always remember, though: The community has no obligation to help you, and you should address your requests politely to them. The information (and software) provided at osmocom.org is put together by volunteers for free. Treat them like a friend whom you're asking for help, not like a supplier from whom you have bought a service.

If you would like to obtain professional/commercial support on Osmocom CNI, you can always reach out to sales@sysmocom.de to discuss your support needs. Purchasing support from sysmocom helps to cover the ongoing maintenance of the Osmocom CNI software stack.

## 4 Overview

### 4.1 About OsmoTRX

OsmoTRX is a C/C++ language implementation of the GSM radio modem, originally developed as the *Transceiver* part of OpenBTS. This radio modem offers an interface based on top of UDP streams.

The OsmoBTS bts_model code for OsmoTRX is called `osmo-bts-trx`. It implements the UDP stream interface of OsmoTRX, so both parts can be used together to implement a complete GSM BTS based on general-purpose computing SDR.

As OsmoTRX is general-purpose software running on top of Linux, it is thus not tied to any specific physical hardware. At the time of this writing, OsmoTRX supports a variety of Lime Microsystems and Ettus USRP SDRs via the UHD driver, as well as the Fairwaves UmTRX and derived products.

OsmoTRX is not a complete GSM PHY but *just* the radio modem. This means that all of the Layer 1 functionality such as scheduling, convolutional coding, etc. is actually also implemented inside OsmoBTS. OsmoTRX is a software-defined radio transceiver that implements the Layer 1 physical layer of a BTS comprising the following 3GPP specifications:

- TS 05.01 "Physical layer on the radio path"
- TS 05.02 "Multiplexing and Multiple Access on the Radio Path"
- TS 05.04 "Modulation"
- TS 05.10 "Radio subsystem synchronization

As such, the boundary between OsmoTRX and `osmo-bts-trx` is at a much lower interface, which is an internal interface of other more traditional GSM PHY implementations.

Besides OsmoTRX, there are also other implementations (both Free Software and proprietary) that implement the same UDP stream based radio modem interface.

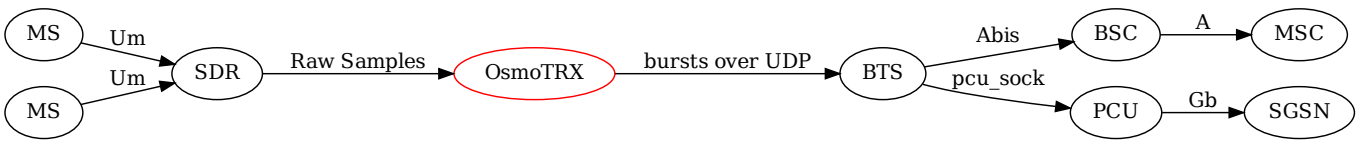



Figure 1: GSM network architecture with OsmoTRX and OsmoBTS

For more information see https://osmocom.org/projects/osmotrx/wiki/OsmoTRX

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

# 5 Running OsmoTRX

The OsmoTRX executable (`osmo-trx`) offers the following command-line options:

## 5.1 SYNOPSIS

**osmo-trx** [-h] [-C *CONFIGFILE*]

## 5.2 OPTIONS

**-h**
Print a short help message about the supported options

**-C *CONFIGFILE***
Specify the file and path name of the configuration file to be used. If none is specified, use `osmo_trx.cfg` in the current working directory.

# 6 Osmocom Control Interface

The VTY interface as described in Section 8 is aimed at human interaction with the respective Osmocom program.

Other programs **should not** use the VTY interface to interact with the Osmocom software, as parsing the textual representation is cumbersome, inefficient, and will break every time the formatting is changed by the Osmocom developers.

Instead, the *Control Interface* was introduced as a programmatic interface that can be used to interact with the respective program.

## 6.1 Control Interface Protocol

The control interface protocol is a mixture of binary framing with text based payload.

The protocol for the control interface is wrapped inside the IPA multiplex header with the stream identifier set to IPAC_PROTO_OSMO (0xEE).

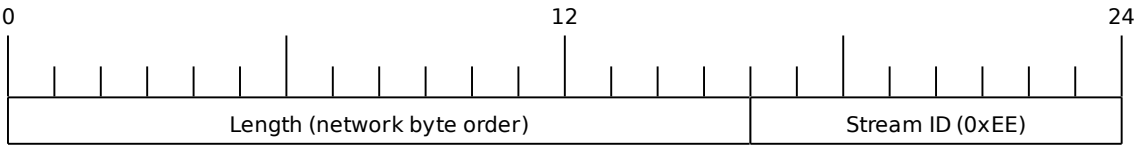



Figure 2: IPA header for control protocol

Inside the IPA header is a single byte of extension header with protocol ID 0x00 which indicates the control interface.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

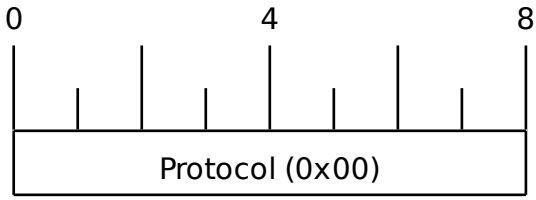



Figure 3: IPA extension header for control protocol

After the concatenation of the two above headers, the plain-text payload message starts. The format of that plain text is illustrated for each operation in the respective message sequence chart in the chapters below.

The fields specified below follow the following meaning:

**<id>**
A numeric identifier, uniquely identifying this particular operation. Value `0` is not allowed unless it's a TRAP message. It will be echoed back in any response to a particular request.

**<var>**
The name of the variable / field affected by the GET / SET / TRAP operation. Which variables/fields are available is dependent on the specific application under control.

**<val>**
The value of the variable / field

**<reason>**
A text formatted, human-readable reason why the operation resulted in an error.

#### 6.1.1 GET operation

The GET operation is performed by an external application to get a certain value from inside the Osmocom application.

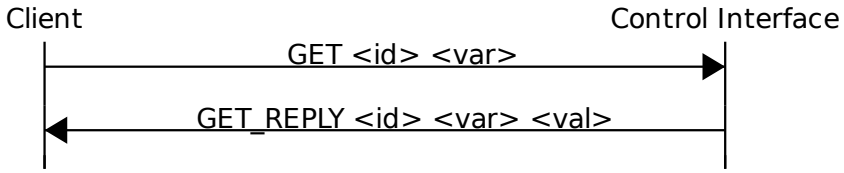



Figure 4: Control Interface GET operation (successful outcome)

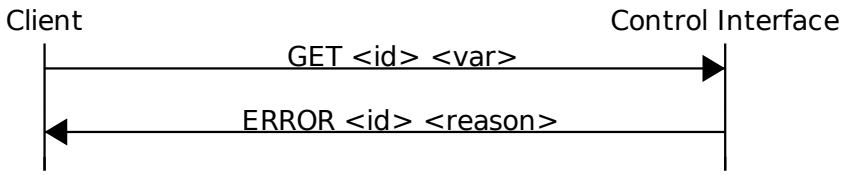



Figure 5: Control Interface GET operation (unsuccessful outcome)

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

#### 6.1.2 SET operation

The SET operation is performed by an external application to set a value inside the Osmocom application.

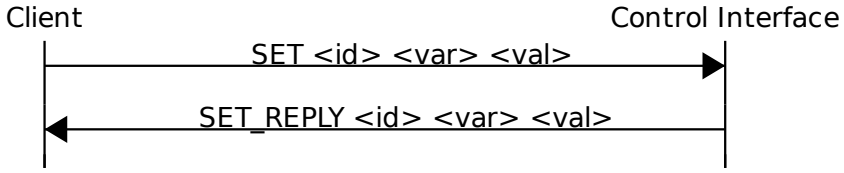



Figure 6: Control Interface SET operation (successful outcome)

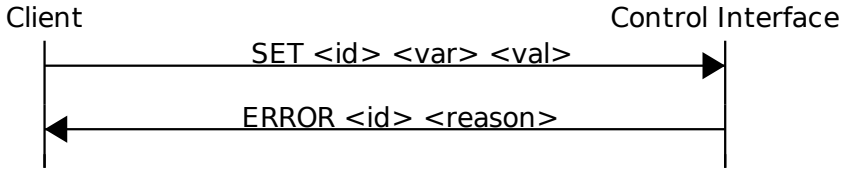



Figure 7: Control Interface SET operation (unsuccessful outcome)

#### 6.1.3 TRAP operation

The program can at any time issue a trap. The term is used in the spirit of SNMP.

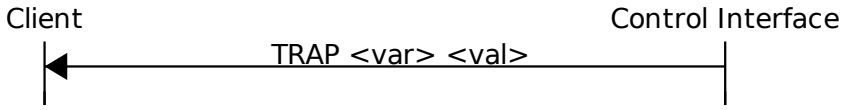



Figure 8: Control Interface TRAP operation

### 6.2 Common variables

There are several variables which are common to all the programs using control interface. They are described in the following table.

Table 1: Variables available over control interface

| Name | Access | Value | Comment |
|---|---|---|---|
| counter.* | RO | | Get counter value. |
| rate_ctr.* | RO | | Get list of rate counter groups. |
| rate_ctr.IN.GN.GI.name | RO | | Get value for interval IN of rate counter name which belong to group named GN with index GI. |

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

Those read-only variables allow to get value of arbitrary counter using its name.

For example "rate_ctr.per_hour.bsc.0.handover:timeout" is the number of handover timeouts per hour.

Of course for that to work the program in question have to register corresponding counter names and groups using libosmocore functions.

In the example above, "bsc" is the rate counter group name and "0" is its index. It is possible to obtain all the rate counters in a given group by requesting "rate_ctr.per_sec.bsc.*" variable.

The list of available groups can be obtained by requesting "rate_ctr.*" variable.

The rate counter group name have to be prefixed with interval specification which can be any of "**per_sec**", "**per_min**", "**per_hour**", "**per_day**" or "**abs**" for absolute value.

The old-style counters available via "counter.*" variables are superseded by "rate_ctr.abs" so its use is discouraged. There might still be some applications not yet converted to rate_ctr.

## 6.3 Control Interface python examples

In the osmo-python-tests repository, there is an example python script called scripts/osmo_ctrl.py which implements the Osmocom control interface protocol.

You can use this tool either stand-alone to perform control interface operations against an Osmocom program, or you can use it as a reference for developing your own python software talking to the control interface.

Another implementation is in scripts/osmo_rate_ctr2csv.py which will retrieve performance counters for a given Osmocom program and output it in csv format. This can be used to periodically (using systemd timer for example) retrieve data to build KPI and evaluate how it changes over time.

Internally it uses "rate_ctr.*" variable described in Section 6.2 to get the list of counter groups and than request all the counters in each group. Applications interested in individual metrics can request it directly using rate_ctr2csv.py as an example.

### 6.3.1 Getting rate counters

**Example: Use rate_ctr2csv.py to get rate counters from OsmoBSC**

```
$ ./scripts/osmo_rate_ctr2csv.py --header
Connecting to localhost:4249...
Getting rate counter groups info...
"group","counter","absolute","second","minute","hour","day"
"e1inp.0","hdlc:abort","0","0","0","0","0"
"e1inp.0","hdlc:bad_fcs","0","0","0","0","0"
"e1inp.0","hdlc:overrun","0","0","0","0","0"
"e1inp.0","alarm","0","0","0","0","0"
"e1inp.0","removed","0","0","0","0","0"
"bsc.0","chreq:total","0","0","0","0","0"
"bsc.0","chreq:no_channel","0","0","0","0","0"
...
"msc.0","call:active","0","0","0","0","0"
"msc.0","call:complete","0","0","0","0","0"
"msc.0","call:incomplete","0","0","0","0","0"
Completed: 44 counters from 3 groups received.
```

### 6.3.2 Setting a value

**Example: Use osmo_ctrl.py to set the short network name of OsmoBSC**

```
$ ./osmo_ctrl.py -d localhost -s short-name 32C3
Got message: SET_REPLY 1 short-name 32C3
```

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

### 6.3.3 Getting a value

**Example: Use osmo_ctrl.py to get the mnc of OsmoBSC**

```
$ ./osmo_ctrl.py -d localhost -g mnc
Got message: GET_REPLY 1 mnc 262
```

### 6.3.4 Listening for traps

You can use `osmo_ctrl.py` to listen for traps the following way:

**Example: Using osmo_ctrl.py to listen for traps:**

```
$ ./osmo_ctrl.py -d localhost -m
❶
```

❶ the command will not return and wait for any TRAP messages to arrive

# 7 Control interface

The actual protocol is described in Section 6, the variables common to all programs using it are described in Section 6.2. Here we describe variables specific to OsmoTRX.

Table 2: Variables available over control interface

| Name | Access | Trap | Value | Comment |
|---|---|---|---|---|

# 8 The Osmocom VTY Interface

All human interaction with Osmocom software is typically performed via an interactive command-line interface called the *VTY*.

**Note**
Integration of your programs and scripts should **not** be done via the telnet VTY interface, which is intended for human interaction only: the VTY responses may arbitrarily change in ways obvious to humans, while your scripts' parsing will likely break often. For external software to interact with Osmocom programs (besides using the dedicated protocols), it is strongly recommended to use the Control interface instead of the VTY, and to actively request / implement the Control interface commands as required for your use case.

The interactive telnet VTY is used to

- explore the current status of the system, including its configuration parameters, but also to view run-time state and statistics,
- review the currently active (running) configuration,
- perform interactive changes to the configuration (for those items that do not require a program restart),
- store the current running configuration to the config file,
- enable or disable logging; to the VTY itself or to other targets.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

The Virtual Tele Type (VTY) has the concept of *nodes* and *commands*. Each command has a name and arguments. The name may contain a space to group several similar commands into a specific group. The arguments can be a single word, a string, numbers, ranges or a list of options. The available commands depend on the current node. there are various keyboard shortcuts to ease finding commands and the possible argument values.

Configuration file parsing during program start is actually performed the VTY's CONFIG node, which is also available in the telnet VTY. Apart from that, the telnet VTY features various interactive commands to query and instruct a running Osmocom program. A main difference is that during config file parsing, consistent indenting of parent vs. child nodes is required, while the interactive VTY ignores indenting and relies on the *exit* command to return to a parent node.

**Note**
In the *CONFIG* node, it is not well documented which commands take immediate effect without requiring a program restart. To save your current config with changes you may have made, you may use the `write file` command to **overwrite** your config file with the current configuration, after which you should be able to restart the program with all changes taking effect.

This chapter explains most of the common nodes and commands. A more detailed list is available in various programs' VTY reference manuals, e.g. see [vty-ref-osmomsc].

There are common patterns for the parameters, these include IPv4 addresses, number ranges, a word, a line of text and choice. The following will explain the commonly used syntactical patterns:

Table 3: VTY Parameter Patterns

| Pattern | Example | Explanation |
|---|---|---|
| `A.B.C.D` | `127.0.0.1` | An IPv4 address |
| `A.B.C.D/M` | `192.168.1.0/24` | An IPv4 address and mask |
| `X:X::X:X` | `::1` | An IPv6 address |
| `X:X::X:X/M` | `::1/128` | An IPv6 address and mask |
| `TEXT` | `example01` | A single string without any spaces, tabs |
| `.TEXT` | `Some information` | A line of text |
| `(OptionA\|OptionB\|OptionC)` | `OptionA` | A choice between a list of available options |
| `<0-10>` | `5` | A number from a range |

## 8.1 Accessing the telnet VTY

The VTY of a given Osmocom program is implemented as a telnet server, listening to a specific TCP port.

Please see Appendix A to check for the default TCP port number of the VTY interface of the specific Osmocom software you would like to connect to.

As telnet is insecure and offers neither strong authentication nor encryption, the VTY by default only binds to localhost (127.0.0.1) and will thus not be reachable by other hosts on the network.

**Warning**
By default, any user with access to the machine running the Osmocom software will be able to connect to the VTY. We assume that such systems are single-user systems, and anyone with local access to the system also is authorized to access the VTY. If you require stronger security, you may consider using the packet filter of your operating system to restrict access to the Osmocom VTY ports further.

## 8.2 VTY Nodes

The VTY by default has the following minimal nodes:

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

**VIEW**
When connecting to a telnet VTY, you will be on the *VIEW* node. As its name implies, it can only be used to view the system status, but it does not provide commands to alter the system state or configuration. As long as you are in the non-privileged *VIEW* node, your prompt will end in a > character.

**ENABLE**
The *ENABLE* node is entered by the `enable` command, from the *VIEW* node. Changing into the *ENABLE* node will unlock all kinds of commands that allow you to alter the system state or perform any other change to it. The *ENABLE* node and its children are signified by a # character at the end of your prompt.
You can change back from the *ENABLE* node to the *VIEW* node by using the `disable` command.

**CONFIG**
The *CONFIG* node is entered by the `configure terminal` command from the *ENABLE* node. The config node is used to change the run-time configuration parameters of the system. The prompt will indicate that you are in the config node by a `(config)#` prompt suffix.
You can always leave the *CONFIG* node or any of its children by using the `end` command.
This node is also automatically entered at the time the configuration file is read. All configuration file lines are processed as if they were entered from the VTY *CONFIG* node at start-up.

**Other**
Depending on the specific Osmocom program you are running, there will be few or more other nodes, typically below the *CONFIG* node. For example, the OsmoBSC has nodes for each BTS, and within the BTS node one for each TRX, and within the TRX node one for each Timeslot.

## 8.3 Interactive help

The VTY features an interactive help system, designed to help you to efficiently navigate is commands.

---

**Note**
The VTY is present on most Osmocom GSM/UMTS/GPRS software, thus this chapter is present in all the relevant manuals. The detailed examples below assume you are executing them on the OsmoMSC VTY. They will work in similar fashion on the other VTY interfaces, while the node structure will differ in each program.

---

### 8.3.1 The question-mark (?) command

If you type a single `?` at the prompt, the VTY will display possible completions at the exact location of your currently entered command.

If you type `?` at an otherwise empty command (without having entered even only a partial command), you will get a list of the first word of all possible commands available at this node:

**Example: Typing ? at start of OsmoMSC prompt**

```
OsmoMSC> ❶
  show         Show running system information
  list         Print command list
  exit         Exit current mode and down to previous mode
  help         Description of the interactive help system
  enable       Turn on privileged mode command
  terminal     Set terminal line parameters
  who          Display who is on vty
  logging      Configure logging
  no           Negate a command or set its defaults
  sms          SMS related commands
  subscriber   Operations on a Subscriber
```

❶ Type `?` here at the prompt, the `?` itself will not be printed.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

If you have already entered a partial command, `?` will help you to review possible options of how to continue the command. Let's say you remember that `show` is used to investigate the system status, but you don't remember the exact name of the object. Hitting `?` after typing `show` will help out:

**Example: Typing ? after a partial command**

```
OsmoMSC> show ❶
  version          Displays program version
  online-help      Online help
  history          Display the session command history
  cs7              ITU-T Signaling System 7
  logging          Show current logging configuration
  alarms           Show current logging configuration
  talloc-context   Show talloc memory hierarchy
  stats            Show statistical values
  asciidoc         Asciidoc generation
  rate-counters    Show all rate counters
  fsm              Show information about finite state machines
  fsm-instances    Show information about finite state machine instances
  sgs-connections  Show SGs interface connections / MMEs
  subscriber       Operations on a Subscriber
  bsc              BSC
  connection       Subscriber Connections
  transaction      Transactions
  statistics       Display network statistics
  sms-queue        Display SMSqueue statistics
  smpp          SMPP Interface
```

❶ Type `?` after the `show` command, the `?` itself will not be printed.

You may pick the `bsc` object and type `?` again:

**Example: Typing ? after show bsc**

```
OsmoMSC> show bsc
  <cr>
```

By presenting `<cr>` as the only option, the VTY tells you that your command is complete without any remaining arguments being available, and that you should hit enter, a.k.a. "carriage return".

### 8.3.2 TAB completion

The VTY supports tab (tabulator) completion. Simply type any partial command and press `<tab>`, and it will either show you a list of possible expansions, or completes the command if there's only one choice.

**Example: Use of <tab> pressed after typing only s as command**

```
OsmoMSC> s❶
show       sms        subscriber
```

❶ Type `<tab>` here.

At this point, you may choose `show`, and then press `<tab>` again:

**Example: Use of <tab> pressed after typing show command**

```
OsmoMSC> show ❶
version    online-help history    cs7        logging    alarms
talloc-context stats      asciidoc   rate-counters fsm        fsm-instances
sgs-connections subscriber bsc        connection transaction statistics
sms-queue smpp
```

❶ Type `<tab>` here.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

### 8.3.3 The `list` command

The `list` command will give you a full list of all commands and their arguments available at the current node:

**Example: Typing list at start of OsmoMSC VIEW node prompt**

```
OsmoMSC> list
  show version
  show online-help
  list
  exit
  help
  enable
  terminal length <0-512>
  terminal no length
  who
  show history
  show cs7 instance <0-15> users
  show cs7 (sua|m3ua|ipa) [<0-65534>]
  show cs7 instance <0-15> asp
  show cs7 instance <0-15> as (active|all|m3ua|sua)
  show cs7 instance <0-15> sccp addressbook
  show cs7 instance <0-15> sccp users
  show cs7 instance <0-15> sccp ssn <0-65535>
  show cs7 instance <0-15> sccp connections
  show cs7 instance <0-15> sccp timers
  logging enable
  logging disable
  logging filter all (0|1)
  logging color (0|1)
  logging timestamp (0|1)
  logging print extended-timestamp (0|1)
  logging print category (0|1)
  logging print category-hex (0|1)
  logging print level (0|1)
  logging print file (0|1|basename) [last]
  logging set-log-mask MASK
  logging level (rll|cc|mm|rr|mncc|pag|msc|mgcp|ho|db|ref|ctrl|smpp|ranap|vlr|iucs|bssap| ↩
      sgs|lglobal|llapd|linp|lmux|lmi|lmib|lsms|lctrl|lgtp|lstats|lgsup|loap|lss7|lsccp|lsua ↩
      |lm3ua|lmgcp|ljibuf|lrspro) (debug|info|notice|error|fatal)
  logging level set-all (debug|info|notice|error|fatal)
  logging level force-all (debug|info|notice|error|fatal)
  no logging level force-all
  show logging vty
  show alarms
  show talloc-context (application|all) (full|brief|DEPTH)
  show talloc-context (application|all) (full|brief|DEPTH) tree ADDRESS
  show talloc-context (application|all) (full|brief|DEPTH) filter REGEXP
  show stats
  show stats level (global|peer|subscriber)
  show asciidoc counters
  show rate-counters
  show fsm NAME
  show fsm all
  show fsm-instances NAME
  show fsm-instances all
  show sgs-connections
  show subscriber (msisdn|extension|imsi|tmsi|id) ID
  show subscriber cache
  show bsc
  show connection
  show transaction
  sms send pending
```

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

```
  sms delete expired
  subscriber create imsi ID
  subscriber (msisdn|extension|imsi|tmsi|id) ID sms sender (msisdn|extension|imsi|tmsi|id)  ↩
      SENDER_ID send .LINE
  subscriber (msisdn|extension|imsi|tmsi|id) ID silent-sms sender (msisdn|extension|imsi| ↩
      tmsi|id) SENDER_ID send .LINE
  subscriber (msisdn|extension|imsi|tmsi|id) ID silent-call start (any|tch/f|tch/any|sdcch)
  subscriber (msisdn|extension|imsi|tmsi|id) ID silent-call stop
  subscriber (msisdn|extension|imsi|tmsi|id) ID ussd-notify (0|1|2) .TEXT
  subscriber (msisdn|extension|imsi|tmsi|id) ID ms-test close-loop (a|b|c|d|e|f|i)
  subscriber (msisdn|extension|imsi|tmsi|id) ID ms-test open-loop
  subscriber (msisdn|extension|imsi|tmsi|id) ID paging
  show statistics
  show sms-queue
  logging filter imsi IMSI
  show smpp esme
```

**Tip**

Remember, the list of available commands will change significantly depending on the Osmocom program you are accessing, its software version and the current node you're at. Compare the above example of the OsmoMSC *VIEW* node with the list of the OsmoMSC *NETWORK* config node:

**Example: Typing list at start of OsmoMSC NETWORK config node prompt**

```
OsmoMSC(config-net)# list
  help
  list
  write terminal
  write file
  write memory
  write
  show running-config
  exit
  end
  network country code <1-999>
  mobile network code <0-999>
  short name NAME
  long name NAME
  encryption a5 <0-3> [<0-3>] [<0-3>] [<0-3>]
  authentication (optional|required)
  rrlp mode (none|ms-based|ms-preferred|ass-preferred)
  mm info (0|1)
  timezone <-19-19> (0|15|30|45)
  timezone <-19-19> (0|15|30|45) <0-2>
  no timezone
  periodic location update <6-1530>
  no periodic location update
```

#### 8.3.4 The attribute system

The VTY allows to edit the configuration at runtime. For many VTY commands the configuration change is immediately valid but for some commands a change becomes valid on a certain event only. In some cases it is even necessary to restart the whole process.

To give the user an overview, which configuration change applies when, the VTY implemets a system of attribute flags, which can be displayed using the `show` command with the parameter `vty-attributes`

**Example: Typing show vty-attributes at the VTY prompt**

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

```
OsmoBSC> show vty-attributes
  Global attributes:
    ^  This command is hidden (check expert mode)
    !  This command applies immediately
    @  This command applies on VTY node exit
  Library specific attributes:
    A  This command applies on ASP restart
    I  This command applies on IPA link establishment
    L  This command applies on E1 line update
  Application specific attributes:
    o  This command applies on A-bis OML link (re)establishment
    r  This command applies on A-bis RSL link (re)establishment
    l  This command applies for newly created lchans
```

The attributes are symbolized through a single ASCII letter (flag) and do exist in three levels. This is more or less due to the technical aspects of the VTY implementation. For the user, the level of an attribute has only informative purpose.

The global attributes, which can be found under the same attribute letter in every osmocom application, exist on the top level. The Library specific attributes below are used in various osmocom libraries. Like with the global attributes the attribute flag letter stays the same throughout every osmocom application here as well. On the third level one can find the application specific attributes. Those are unique to each osmocom application and the attribute letters may have different meanings in different osmocom applications. To make the user more aware of this, lowercase letters were used as attribute flags.

The `list` command with the parameter `with-flags` displays a list of available commands on the current VTY node, along with attribute columns on the left side. Those columns contain the attribute flag letters to indicate to the user how the command behaves in terms of how and when the configuration change takes effect.

**Example: Typing list with-flags at the VTY prompt**

```
OsmoBSC(config-net-bts)# list with-flags
  . ...  help
  . ...  list [with-flags]
  . ...  show vty-attributes
  . ...  show vty-attributes (application|library|global)
  . ...  write terminal
  . ...  write file [PATH]
  . ...  write memory
  . ...  write
  . ...  show running-config ❶
  . ...  exit
  . ...  end
  . o..  type (unknown|bs11|nanobts|rbs2000|nokia_site|sysmobts) ❷
  . ...  description .TEXT
  . ...  no description
  . o..  band BAND
  . .r.  cell_identity <0-65535> ❸
  . .r.  dtx uplink [force]
  . .r.  dtx downlink
  . .r.  no dtx uplink
  . .r.  no dtx downlink
  . .r.  location_area_code <0-65535>
  . o..  base_station_id_code <0-63>
  . o..  ipa unit-id <0-65534> <0-255>
  . o..  ipa rsl-ip A.B.C.D
  . o..  nokia_site skip-reset (0|1)
  ! ...  nokia_site no-local-rel-conf (0|1) ❹
  ! ...  nokia_site bts-reset-timer  <15-100> ❺
```

❶ This command has no attributes assigned.

❷ This command applies on A-bis OML link (re)establishment.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

❸ This command applies on A-bis RSL link (re)establishment.

❹, ❺ This command applies immediately.

There are multiple columns because a single command may be associated with multiple attributes at the same time. To improve readability each flag letter gets a dedicated column. Empty spaces in the column are marked with a dot (".")

In some cases the listing will contain commands that are associated with no flags at all. Those commands either play an exceptional role (interactive commands outside "configure terminal", vty node navigation commands, commands to show / write the config file) or will require a full restart of the overall process to take effect.

### 8.3.5 The expert mode

Some VTY commands are considered relatively dangerous if used in production operation, so the general approach is to hide them. This means that they don't show up anywhere but the source code, but can still be executed. On the one hand, this approach reduces the risk of an accidental invocation and potential service degradation; on the other, it complicates intentional use of the hidden commands.

The VTY features so-called *expert* mode, that makes the hidden commands appear in the interactive help, as well as in the XML VTY reference, just like normal ones. This mode can be activated from the *VIEW* node by invoking the `enable` command with the parameter `expert-mode`. It remains active for the individual VTY session, and gets disabled automatically when the user switches back to the *VIEW* node or terminates the session.

A special attribute in the output of the `list with-flags` command indicates whether a given command is hidden in normal mode, or is a regular command:

**Example: Hidden commands in the output of the list with-flags command**

```
OsmoBSC> enable expert-mode ❶
OsmoBSC# list with-flags
  ...
  ^   bts <0-255> (activate-all-lchan|deactivate-all-lchan) ❷
  ^   bts <0-255> trx <0-255> (activate-all-lchan|deactivate-all-lchan) ❸
  .   bts <0-255> trx <0-255> timeslot <0-7> sub-slot <0-7> mdcx A.B.C.D <0-65535> ❹
  ^   bts <0-255> trx <0-255> timeslot <0-7> sub-slot <0-7> (borken|unused) ❺
  .   bts <0-255> trx <0-255> timeslot <0-7> sub-slot <0-7> handover <0-255> ❻
  .   bts <0-255> trx <0-255> timeslot <0-7> sub-slot <0-7> assignment ❼
  .   bts <0-255> smscb-command (normal|schedule|default) <1-4> HEXSTRING ❽
  ...
```

❶ This command enables the *expert* mode.

❷, ❸, ❺ This is a hidden command (only shown in the *expert* mode).

❹, ❻, ❼, ❽ This is a regular command that is always shown regardless of the mode.

# 9 libosmocore Logging System

In any reasonably complex software it is important to understand how to enable and configure logging in order to get a better insight into what is happening, and to be able to follow the course of action. We therefore ask the reader to bear with us while we explain how the logging subsystem works and how it is configured.

Most Osmocom Software (like `osmo-bts`, `osmo-bsc`, `osmo-nitb`, `osmo-sgsn` and many others) uses the same common logging system.

This chapter describes the architecture and configuration of this common logging system.

The logging system is composed of

- log targets (where to log),

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

- log categories (who is creating the log line),
- log levels (controlling the verbosity of logging), and
- log filters (filtering or suppressing certain messages).

All logging is done in human-readable ASCII-text. The logging system is configured by means of VTY commands that can either be entered interactively, or read from a configuration file at process start time.

## 9.1 Log categories

Each sub-system of the program in question typically logs its messages as a different category, allowing fine-grained control over which log messages you will or will not see. For example, in OsmoBSC, there are categories for the protocol layers `rsl`, `rr`, `mm`, `cc` and many others. To get a list of categories interactively on the vty, type: `logging level ?`

## 9.2 Log levels

For each of the log categories (see Section 9.1), you can set an independent log level, controlling the level of verbosity. Log levels include:

**fatal**
Fatal messages, causing abort and/or re-start of a process. This *shouldn't happen*.

**error**
An actual error has occurred, its cause should be further investigated by the administrator.

**notice**
A noticeable event has occurred, which is not considered to be an error.

**info**
Some information about normal/regular system activity is provided.

**debug**
Verbose information about internal processing of the system, used for debugging purpose. This will log the most.

The log levels are inclusive, e.g. if you select *info*, then this really means that all events with a level of at least *info* will be logged, i.e. including events of *notice*, *error* and *fatal*.

So for example, in OsmoBSC, to set the log level of the Mobility Management category to info, you can use the following command: `log level mm info`.

There is also a special command to set all categories as a one-off to a desired log level. For example, to silence all messages but those logged as notice and above issue the command: `log level set-all notice`

Afterwards you can adjust specific categories as usual.

A similar command is `log level force-all <level>` which causes all categories to behave as if set to log level <level> until the command is reverted with `no log level force-all` after which the individually-configured log levels will again take effect. The difference between `set-all` and `force-all` is that `set-all` actually changes the individual category settings while `force-all` is a (temporary) override of those settings and does not change them.

## 9.3 Log printing options

The logging system has various options to change the information displayed in the log message.

**log color 1**
With this option each log message will log with the color of its category. The color is hard-coded and can not be changed. As with other options a *0* disables this functionality.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

**log timestamp 1**
Includes the current time in the log message. When logging to syslog this option should not be needed, but may come in handy when debugging an issue while logging to file.

**log print extended-timestamp 1**
In order to debug time-critical issues this option will print a timestamp with millisecond granularity.

**log print category 1**
Prefix each log message with the category name.

**log print category-hex 1**
Prefix each log message with the category number in hex (<*000b*>).

**log print level 1**
Prefix each log message with the name of the log level.

**log print file 1**
Prefix each log message with the source file and line number. Append the keyword `last` to append the file information instead of prefixing it.

## 9.4 Log filters

The default behavior is to filter out everything, i.e. not to log anything. The reason is quite simple: On a busy production setup, logging all events for a given subsystem may very quickly be flooding your console before you have a chance to set a more restrictive filter.

To request no filtering, i.e. see all messages, you may use: `log filter all 1`

In addition to generic filtering, applications can implement special log filters using the same framework to filter on particular context.

For example in OsmoBSC, to only see messages relating to a particular subscriber identified by his IMSI, you may use: `log filter imsi 262020123456789`

## 9.5 Log targets

Each of the log targets represent certain destination for log messages. It can be configured independently by selecting levels (see Section 9.2) for categories (see Section 9.1) as well as filtering (see Section 9.4) and other options like `logging timestamp` for example.

### 9.5.1 Logging to the VTY

Logging messages to the interactive command-line interface (VTY) is most useful for occasional investigation by the system administrator.

Logging to the VTY is disabled by default, and needs to be enabled explicitly for each such session. This means that multiple concurrent VTY sessions each have their own logging configuration. Once you close a VTY session, the log target will be destroyed and your log settings be lost. If you re-connect to the VTY, you have to again activate and configure logging, if you wish.

To create a logging target bound to a VTY, you have to use the following command: `logging enable` This doesn't really activate the generation of any output messages yet, it merely creates and attaches a log target to the VTY session. The newly-created target still doesn't have any filter installed, i.e. *all log messages will be suppressed by default*

Next, you can configure the log levels for desired categories in your VTY session. See Section 9.1 for more details on categories and Section 9.2 for the log level details.

For example, to set the log level of the Call Control category to debug, you can use: `log level cc debug`

Finally, after having configured the levels, you still need to set the filter as it's described in Section 9.4.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

**Tip**
If many messages are being logged to a VTY session, it may be hard to impossible to still use the same session for any commands. We therefore recommend to open a second VTY session in parallel, and use one only for logging, while the other is used for interacting with the system. Another option would be to use different log target.

To review the current vty logging configuration, you can use: `show logging vty`

### 9.5.2 Logging to the ring buffer

To avoid having separate VTY session just for logging output while still having immediate access to them, one can use `alarms` target. It lets you store the log messages inside the ring buffer of a given size which is available with `show alarms` command.

It's configured as follows:

```
OsmoBSC> enable
OsmoBSC# configure terminal
OsmoBSC(config)# log alarms 98
OsmoBSC(config-log)#
```

In the example above 98 is the desired size of the ring buffer (number of messages). Once it's filled, the incoming log messages will push out the oldest messages available in the buffer.

### 9.5.3 Logging via gsmtap

GSMTAP is normally a pseudo-header format that enables the IP-transport of GSM (or other telecom) protocols that are not normally transported over IP. For example, the most common situation is to enable GSMTAP in OsmoBTS or OsmoPCU to provide GSM-Um air interface capture files over IP, so they can be analyzed in wireshark.

GSMTAP logging is now a method how Osmocom software can also encapsulate its own log output in GSMTAP frames. We're not trying to re-invent rsyslog here, but this is very handy When debugging complex issues. It enables the reader of the pcap file containing GSMTAP logging together with other protocol traces to reconstruct exact chain of events. A single pcap file can then contain both the log output of any number of Osmocom programs in the same timeline of the messages on various interfaces in and out of said Osmocom programs.

It's configured as follows:

```
OsmoBSC> enable
OsmoBSC# configure terminal
OsmoBSC(config)# log gsmtap 192.168.2.3 ❶
OsmoBSC(config-log)#
```

❶ Configure a GSMTAP logging target recording to 192.168.2.3

The hostname/ip argument is optional: if omitted the default 127.0.0.1 will be used. The log strings inside GSMTAP are already supported by Wireshark. Capturing for `port 4729` on appropriate interface will reveal log messages including source file name and line number as well as application. This makes it easy to consolidate logs from several different network components alongside the air frames. You can also use Wireshark to quickly filter logs for a given subsystem, severity, file name etc.

The GSMTAP logging target also has different implementation I/O modes, which can be configured during the VTY "log gmstap" VTY command:

**nonblocking-io**
Use non-blocking, synchronous I/O. This is the default mode. It will set the GSMTAP source socket as non-blocking, and attempt synchronous write to the UDP socket, discarding a message if the UDP socket's send buffer is full during write.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

**blocking-io**
: Use blocking, synchronous I/O. It will use the GSMTAP source socket in blocking mode, and attempt synchronous write to the UDP socket, blocking the entire thread if the UDP socket's send buffer is full during write. This I/O mode should not be used in any osmocom program driven by an event loop (which includes any program using a VTY), since it may block the thread for an unspecified (potentially long or even infinite) amount of time and hence prevent the event loop to run and process events. This I/O mode should only be used under specific circumstances, like debugging a crash, in which case it's interesting to get the logging output to the UDP socket and hence recorded before the program crashes.

**wq**
: Workqueue mode: Use non-blocking, asynchronous I/O. It will set the GSMTAP source socket as non-blocking, and attempt asynchronous write to the UDP socket by means of a work queue driven by the process poll event loop, delaying writing to the UDP socket until there's enough free space in its send buffer. Since the workqueue has a maximum size, messages will be discarded similar to *nonblocking-io* once the UDP socket's send buffer **and** the GSMTAP log target workqueue are full. This I/O mode provides some extra buffering space in the userspace workqueue, at the expense of extra overhead and delay to transmit messages asynchronously waiting for POLLOUT event from the UDP socket.

In any of the I/O modes above, the UDP socket send buffer size of the GSMTAP log target influences where messages may be dropped (*nonblocking-io*\|*wq*) or the process blocked (*blocking-io*) whenever a high number of messages or test are transmitted in a short time. For that reason, the GSMTAP log target will attempt to set the `SO_SNDBUF` sockopt of the UDP socket to 4MB. In Linux, the `SO_SNDBUF` value being set on the socket will be trimmed by the kernel to a maximum allowed by sysctl `net.core.wmem_max`. Hence, if you want to be on a safer side minimizing dropping messages or blocking the process (depending on the I/O mode set), make sure sysctl `net.core.wmem_max` on your system is set to `4194304` (4MB) or more.

---

**Note**

Linux Kernel used to have a `net.core.wmem_max` of `212992`(208KB) until recently (2025, a6d4f25888b83b8300aef28d9ee22765c1cc9b34), where it was increased to `4194304` (4MB) .

---

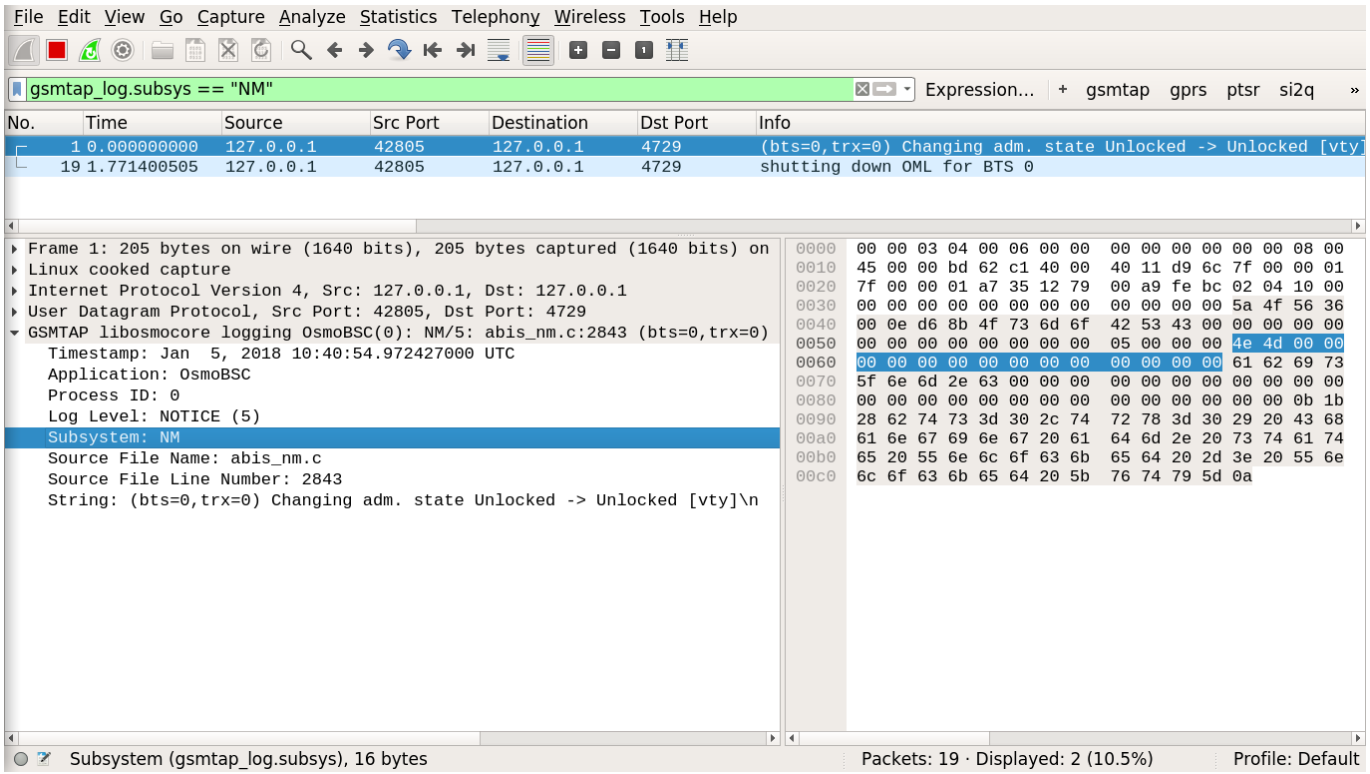



Figure 9: Wireshark with logs delivered over GSMTAP

**Note**
Every time you generate GSMTAP messages and send it to a unicast (non-broadcast/multicast) IP address, please make sure that the destination IP address actually has a socket open on the specified port, or drops the packets in its packet filter. If unicast GSMTAP messages arrive at a closed destination UDP port, the operating system will likely generate ICMP port unreachable messages. Those ICMP messages in turn will, when arriving at the source (the host on which you run the Osmocom software sending GSMTAP), suppress generation of further GSMTAP messages for some time, resulting in incomplete files. In case of doubt, either send GSMTAP to multicast IP addresses, or run something like `nc -l -u -p 4729 > /dev/null` on the destination host to at the GSMTAP port and discard anything arriving at it. If the GSMTAP recording address is identified as a local address, upon creation of the GSMTAP log target, an attempt will be made to create a dummy sink socket to avoid such problems automatically.

### 9.5.4 Logging to a file

As opposed to Logging to the VTY, logging to files is persistent and stored in the configuration file. As such, it is configured in sub-nodes below the configuration node. There can be any number of log files active, each of them having different settings regarding levels / subsystems.

To configure a new log file, enter the following sequence of commands:

```
OsmoBSC> enable
OsmoBSC# configure terminal
OsmoBSC(config)# log file /path/to/my/file ❶
OsmoBSC(config-log)#
```

❶ Configure a file logging target writing to /path/to/my/file

This leaves you at the config-log prompt, from where you can set the detailed configuration for this log file. The available commands at this point are identical to configuring logging on the VTY, they include `logging filter`, `logging level` as well as `logging color` and `logging timestamp`.

The file logging target will by default use a synchronous non-blocking I/O write. If writing cannot be performed synchronously (would block), the message is stored in a workqueue where asynchronous write (driven by poll event loop) will be performed at a later stage. The file logging target can be configured during the VTY `log file` VTY command to perform blocking I/O with the `blocking-io` parameter. This I/O mode should not be used in any osmocom program driven by an event loop (which includes any program using a VTY), since it may block the thread for an unspecified (potentially long or even infinite) amount of time and hence prevent the event loop to run and process events. This I/O mode should only be used under specific circumstances, like debugging a crash, in which case it's interesting to get the logging output to the UDP socket and hence recorded before the program crashes.

**Tip**
Don't forget to use the `copy running-config startup-config` (or its short-hand `write file`) command to make your logging configuration persistent across application re-start.

**Note**
libosmocore provides file close-and-reopen support by SIGHUP, as used by popular log file rotating solutions such as https://github.com/logrotate/logrotate found in most GNU/Linux distributions.

### 9.5.5 Logging to syslog

syslog is a standard for computer data logging maintained by the IETF. Unix-like operating systems like GNU/Linux provide several syslog compatible log daemons that receive log messages generated by application programs.

libosmocore based applications can log messages to syslog by using the syslog log target. You can configure syslog logging by issuing the following commands on the VTY:

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

```
OsmoBSC> enable
OsmoBSC# configure terminal
OsmoBSC(config)# log syslog daemon
OsmoBSC(config-log)#
```

This leaves you at the config-log prompt, from where you can set the detailed configuration for this log file. The available commands at this point are identical to configuring logging on the VTY, they include `logging filter`, `logging level` as well as `logging color` and `logging timestamp`.

**Tip**
Syslog daemons will normally automatically prefix every message with a time-stamp, so you should disable the libosmocore time-stamping by issuing the `logging timestamp 0` command.

**Note**
Typical syslog implementations (such as glibc's one) write to *AF_UNIX* socket in `/dev/log` in a synchronous blocking I/O fashion, which means transmitting a big amount of messages or text over it in a short amount of time (before syslogd has time to read it) could potentially block the osmocom program's thread issuing the logging. It is hence advised avoiding use of this log target to log notable amounts of information in any osmocom program driven by an event loop (which includes any program using a VTY), since it may block the thread for an unspecified (potentially long or even infinite) amount of time and hence prevent the event loop to run and process events.

### 9.5.6 Logging to systemd-journal

systemd has been adopted by the majority of modern GNU/Linux distributions. Along with various daemons and utilities it provides systemd-journald [1] - a daemon responsible for event logging (syslog replacement). libosmocore based applications can log messages directly to systemd-journald.

The key difference from other logging targets is that systemd based logging allows to offload rendering of the meta information, such as location (file name, line number), subsystem, and logging level, to systemd-journald. Furthermore, systemd allows to attach arbitrary meta fields to the logging messages [2], which can be used for advanced log filtering.

[1] https://www.freedesktop.org/software/systemd/man/systemd-journald.service.html [2] https://www.freedesktop.org/software/-systemd/man/systemd.journal-fields.html

It was decided to introduce libsystemd as an optional dependency, so it needs to be enabled explicitly at configure/build time:

```
$ ./configure --enable-systemd-logging
```

**Note**
Recent libosmocore packages provided by Osmocom for Debian and CentOS are compiled **with** libsystemd (https://gerrit.osmocom.org/c/libosmocore/+/22651).

You can configure systemd based logging in two ways:

**Example: systemd-journal target with offloaded rendering**

```
log systemd-journal raw ❶
 logging filter all 1
 logging level set-all notice
```

❶ `raw` logging handler, rendering offloaded to systemd.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

In this example, logging messages will be passed to systemd without any meta information (time, location, level, category) in the text itself, so all the printing parameters like `logging print file` will be ignored. Instead, the meta information is passed separately as *fields* which can be retrieved from the journal and rendered in any preferred way.

```
# Show Osmocom specific fields
$ journalctl --fields | grep OSMO

# Filter messages by logging subsystem at run-time
$ journalctl OSMO_SUBSYS=DMSC -f

# Render specific fields only
$ journalctl --output=verbose \
        --output-fields=SYSLOG_IDENTIFIER,OSMO_SUBSYS,CODE_FILE,CODE_LINE,MESSAGE
```

See `man 7 systemd.journal-fields` for a list of default fields, and `man 1 journalctl` for general information and available formatters.

**Example: systemd-journal target with libosmocore based rendering**

```
log systemd-journal ❶
 logging filter all 1
 logging print file basename
 logging print category-hex 0
 logging print category 1
 logging print level 1
 logging timestamp 0 ❷
 logging color 1 ❸
 logging level set-all notice
```

❶ Generic logging handler, rendering is done by libosmocore.

❷ Disable timestamping, systemd will timestamp every message anyway.

❸ Colored messages can be rendered with `journalctl --output=cat`.

In this example, logging messages will be pre-processed by libosmocore before being passed to systemd. No additional fields will be attached, except the logging level (`PRIORITY`). This mode is similar to *syslog* and *stderr*.

---

**Note**

systemd-journal logging target uses libsystemd, which implements *Native Journal Protocol* (*AF_UNIX _DATAGRAM* socket). That socket is internally configured in synchronous blocking I/O mode, which means transmitting a big amount of messages or text over it in a short amount of time (before systemd-journald has time to read it) could potentially block the osmocom program's thread issuing the logging. libsystemd currently sets the *AF_UNIX* socket `SO_SNDBUF` to 8MB. In Linux, the `SO_SNDBUF` value being set on the socket will be trimmed by the kernel to a maximum allowed by sysctl `net.core.wmem_max`. Hence, if you want to be on a safer side minimizing dropping messages or blocking the process (depending on the I/O mode set), make sure sysctl `net.core.wmem_max` on your system is set to `8388608` (8MB) or more. Besides, the kernel limits the amount of messages queued in *AF_UNIX* sockets to value set in sysctl `net.unix.max_dgram_qlen`, so you may also want to increase that one to be on the safe side if planning to log with a high frequency.

---

---

**Note**

Linux Kernel used to have a `net.core.wmem_max` of `212992`(208KB) until recently (2025, a6d4f25888b83b8300aef28d9ee22765c1cc9b34), where it was increased to `4194304` (4MB) .

---

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

#### 9.5.7 Logging to stderr

If you're not running the respective application as a daemon in the background, you can also use the stderr log target in order to log to the standard error file descriptor of the process.

In order to configure logging to stderr, you can use the following commands:

```
OsmoBSC> enable
OsmoBSC# configure terminal
OsmoBSC(config)# log stderr
OsmoBSC(config-log)#
```

The stderr logging target can be configured during the VTY `log stderr` VTY command to perform blocking I/O with the `blocking-io` parameter. See Section 9.5.4 for more information.

# 10 Counters

These counters and their description are based on OsmoTRX 1.0.0.95-9527 (OsmoTRX).

## 10.1 Rate Counters

Table 4: trx:chan - osmo-trx statistics

| Name | Reference | Description |
|---|---|---|
| device:rx_overruns | [?] | Number of Rx overruns in FIFO queue |
| device:tx_underruns | [?] | Number of Tx underruns in FIFO queue |
| device:rx_drop_events | [?] | Number of times Rx samples were dropped by HW |
| device:rx_drop_samples | [?] | Number of Rx samples dropped by HW |
| device:tx_drop_events | [?] | Number of times Tx samples were dropped by HW |
| device:tx_drop_samples | [?] | Number of Tx samples dropped by HW |

# 11 Osmo Stat Items

# 12 Osmo Counters

## 12.1 Rate Counter Configurable Error Thresholds

Some rate counters such as overruns, underruns and dropped packets indicate events that can really harm correct operation of the BTS served by OsmoTRX, specially if they happen frequently. OsmoTRX is in most cases (depending on maturity of device driver) prepared to dodge the temporary failure and keep running and providing service.

Still, it is sometimes important for this kind of events to not go unnoticed by the operator, since they may indicate issues regarding the set up that may require operator intervention to fix it.

For instance, frequent dropped packets could indicate SDR HW/FW/power errors, or a faulty connection against the host running OsmoTRX.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

They can also indicate issues on the host running OsmoTRX itself: For instance, OsmoTRX may not be running under a high enough priority (hence other processes eventually battling for resources with it), or that simply the HW running OsmoTRX is not powerful enough to accomplish all work in a timely fashion all the time.

As a result, OsmoTRX can be configured to exit the process upon certain conditions being met, in order to let osmoBTS notice something is wrong and thus announcing issues through alarms to the network, where the operator can then investigate the issue by looking at OsmoTRX logs.

These conditions are configured by means of introducing rate counter thresholds in the VTY. The OsmoTRX user can provide those threshold commands either in the VTY cfg file read by OsmoTRX process during startup, or by adding/removing them dynamically through the VTY interactive console.

Each threshold cmd states an event (a rate counter type), a value and an time interval (a second, a minute, an hour or a day). A threshold will be reached (and OsmoTRX stopped) if its value grows bigger than the configured threshold value over the configured time interval. This is the syntax used to manage rate counter thresholds:

```
(no) ctr-error-threshold <EVENT> <VALUE> <INTERVAL>
```

If several rate counter thresholds are set, then all of them are checked over time and the first one reached will stop OsmoTRX.

**Example: rate counter threshold configuration (VTY .cfg file)**

```
trx
 ctr-error-threshold rx_drop_events 2 per-minute ❶
 ctr-error-threshold rx_drop_samples 800 per-second ❷
```

❶ Stop OsmoTRX if dropped event (any amount of samples) during Rx was detected 2 times or more during a minute.

❷ Stop OsmoTRX if 800 or more samples were detected during Rx to be dropped by the HW during a second.

**Example: rate counter threshold configuration (VTY interactive)**

```
OsmoTRX(config-trx)# ctr-error-threshold tx_underruns 3 per-hour ❶
OsmoTRX(config-trx)# no ctr-error-threshold tx_underruns 3 per-hour ❷
```

❶ Stop OsmoTRX if 3 or more underruns were detected during Tx over the last hour

❷ Remove previously set rate counter threshold

# 13 Configuring OsmoTRX

OsmoTRX will read the configuration at startup time and configure the transceiver accordingly after validating the configuration.

OsmoTRX can handle several TRX channels, but at least one must be configured in order to be able to start it successfully. Channels must be present in the configuration file in incremental order, starting from 0 and be consecutive.

Example configuration files for different devices and setups can be found in `doc/examples/` in *osmo-trx* git repository.

## 13.1 Documented example

**Example: Single carrier configuration**

```
trx
 bind-ip 127.0.0.1 ❶
 remote-ip 127.0.0.1 ❷
 base-port 5700 ❸
 egprs disable ❹
 tx-sps 4 ❺
 rx-sps 4 ❻
 chan 0 ❼
  tx-path BAND1 ❽
  rx-path LNAW ❾
```

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

1. Configure the local IP address at the TRX used for the connection against `osmo-bts-trx`.
2. Specify the IP address of `osmo-bts-trx` to connect to.
3. Specify the reference base UDP port to use for communication.
4. Don't enable EDGE support.
5. Use 4 TX samples per symbol. This is device specific.
6. Use 4 RX samples per symbol. This is device specific.
7. Configure the first channel. As no other channels are specified, `osmo-trx` assumes it is using only one channel.
8. Configure the device to use `BAND1` Tx antenna path from all the available ones (device specific).
9. Configure the device to use `LNAW` Rx antenna path from all the available ones (device specific).

## 13.2 Multi-ARFCN mode

The Multi-ARFCN feature allows to have a multi-carrier approach multiplexed on a single physical RF channel, which can introduce several benefits, such as lower cost and higher capacity support.

Multi-ARFCN support is available since osmo-trx release `0.2.0`, and it was added specifically in commit `76764278169d25298085`

osmo-trx must also be built with Multi-ARFCN support enabled, which is the default build configuration. Builds configured with `--disable-multi-arfcn` do not provide this feature.

This feature is useful for instance if you want to run more than 1 TRX with an Ettus B200 device, or more than 2 TRXs with an Ettus B210 device, since they support only 1 and 2 physical RF channels respectively. No device from other providers or even other devices than B200 and B210 from Ettus are known to support this feature.

With multi-ARFCN enabled, ARFCN spacing is fixed at 800 kHz or 4 GSM channels. So if TRX-0 is set to ARFCN 51, TRX-1 *must* be set to 55, and so on. Up to three ARFCN's is supported for multi-TRX.

From BTS and BSC point of view, supporting multiple TRXs through multi-ARFCN feature in OsmoTRX doesn't make any difference from a regular multi-TRX setup, leaving apart of course the mentioned ARFCN limitations explained above and as a consequence physical installation and operational differences.

**Example: osmo-bts-trx.cfg using 2 TRX against an osmo-trx driven device**

```
phy 0
 osmotrx ip local 127.0.0.1
 osmotrx ip remote 127.0.0.1
 instance 0
 instance 1
bts 0
 ...
 band GSM-1800
 trx 0
  phy 0 instance 0
 trx 1
  phy 0 instance 1
```

**Example: osmo-trx.cfg using Multi-ARFCN mode to run 2 TRX**

```
trx
 ...
 multi-arfcn enable
 chan 0
 chan 1
```

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

# 14 OsmoTRX hardware architecture support

OsmoTRX comes out-of-the-box with several algorithms and operations optimized for certain instruction-set architectures, as well as non-optimized fall-back algorithms in case required instruction sets are not supported by the compiler at compile time or by the executing machine at run-time. Support for these optimized algorithms can be enabled and disabled by means of configure flags. Accelerated operations include pulse shape filtering, resampling, sequence correlation, and many other signal processing operations.

On Intel processors, OsmoTRX makes heavy use of the Streaming SIMD Extensions (SSE) instruction set. SSE3 is the minimum requirement for accelerated use. SSE3 is present in the majority of Intel processors since later versions of the Pentium 4 architecture and is also present on low power Atom processors. Support is automatically detected at build time. SSE4.1 instruction set is supported too. This feature is enabled by default unless explicitly disabled by passing the configure flag *--with-sse=no*. When enabled, the compiler will build an extra version of each of the supported algorithms using each of the supported mentioned instruction sets. Then, at run-time, OsmoTRX will auto-detect capabilities of the executing machine and enable an optimized algorithm using the most suitable available (previously compiled) instruction set.

On ARM processors, NEON and NEON FMA are supported. Different to the x86, there is no auto-detection in this case, nor difference between compile and runtime. NEON support is disabled by default and can be enabled by passing the flag *--with-neon=yes* to the configure script; the used compiler must support NEON instruction set and the resulting binary will only run fine on an ARM board supporting NEON extensions. Running OsmoTRX built with flag *--with-neon* on a board without NEON instruction set support, will most probably end up in the process being killed with a *SIGILL* Illegal Instruction signal by the operating system. NEON FMA (Fused Multiply-Add) is an extension to the NEON instruction set, and its use in OsmoTRX can be enabled by passing the *--with_neon_vfpv4* flag, which will also implicitly enable NEON support (*--with_neon*).

# 15 OsmoTRX hardware device support

OsmoTRX consists of a *common* part that applies to all TRX devices as well as *hardware-specific* parts for each TRX device. The hardware-specific parts are usually provided by vendor-specific or device-specific libraries that are then handled by some OsmoTRX glue code presenting a unified interface towards the rest of the code by means of a *RadioDevice* class.

The common part includes the core TRX architecture as well as code for implementing the external interfaces such as the TRX Manager UDP socket, control, and VTY interfaces.

The hardware-specific parts include support for driving one particular implementation of a radio modem. Such a physical layer implementation can come in many forms. Sometimes it runs on a general purpose CPU, sometimes on a dedicated ARM core, a dedicated DSP, a combination of DSP and FPGA.

Joining the common part with each of the available backends results in a different binary with different suffix for each backend. For instance, when OsmoTRX is built with UHD backend, an *osmo-trx-uhd* binary is generated; when OsmoTRX is built with LimeSuite backend, an *osmo-trx-lms* binary is generated. Build of different backend can be enabled and disabled by means of configure flags, which can be found in each subsection relative to each backend below.

## 15.1 Ettus USRP1

The binary *osmo-trx-usrp1* is used to drive this device, see Section 16.3.

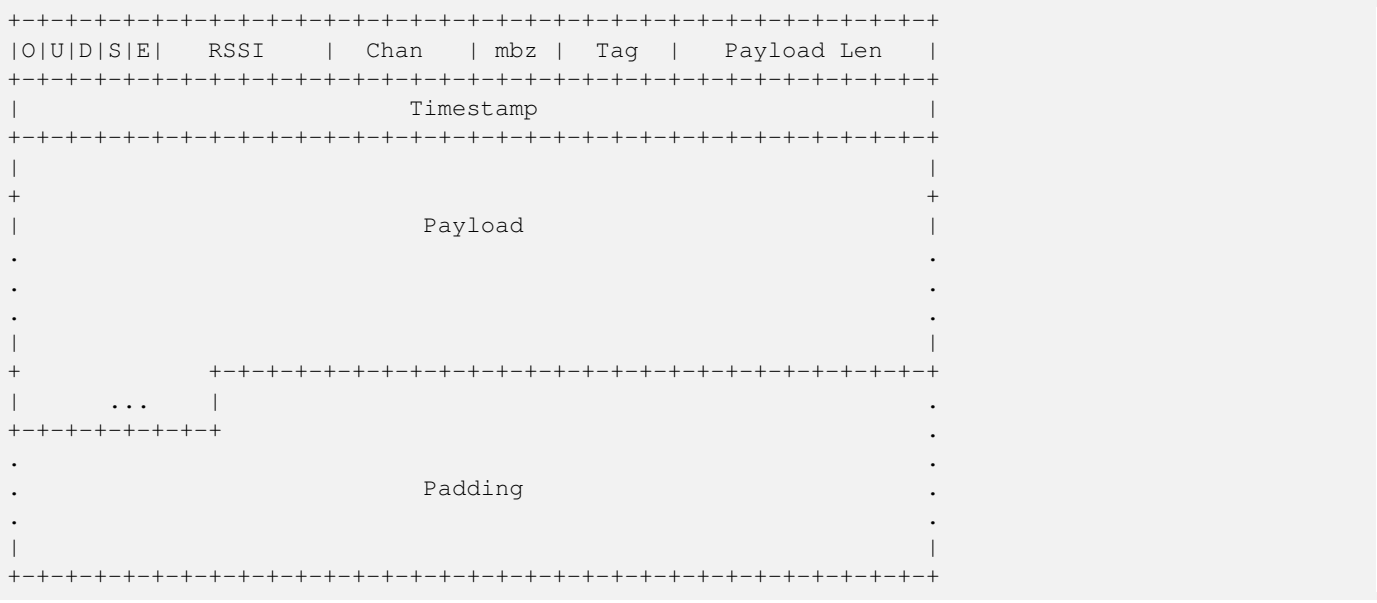
### 15.1.1 USRP1 in-band USB protocol

This section specifies the format of USB packets used for in-band data transmission and signaling on the USRP1. All packets are 512-byte long, and are transferred using USB "bulk" transfers.

IN packets are sent towards the host. OUT packets are sent away from the host.

The layout is 32-bits wide. All data is transmitted in little-endian format across the USB.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 




**mbz: Must be Zero**
These bits must be zero in both IN and OUT packets.

**O: Overrun Flag**
Set in an IN packet if an overrun condition was detected. Must be zero in OUT packets. Overrun occurs when the FPGA has data to transmit to the host and there is no buffer space available. This generally indicates a problem on the host. Either it is not keeping up, or it has configured the FPGA to transmit data at a higher rate than the transport (USB) can support.

**U: Underrun Flag**
Set in an IN packet if an underrun condition was detected. Must be zero in OUT packets. Underrun occurs when the FPGA runs out of samples, and it's not between bursts. See the "End of Burst flag" below.

**D: Dropped Packet Flag**
Set in an IN packet if the FPGA discarded an OUT packet because its timestamp had already passed.

**S: Start of Burst Flag**
Set in an OUT packet if the data is the first segment of what is logically a continuous burst of data. Must be zero in IN packets.

**E: End of Burst Flag**
Set in an OUT packet if the data is the last segment of what is logically a continuous burst of data. Must be zero in IN packets. Underruns are not reported when the FPGA runs out of samples between bursts.

**RSSI: 6-bit Received Strength Signal Indicator**
Must be zero in OUT packets. In IN packets, indicates RSSI as reported by front end. FIXME The format and interpretation are to be determined.

**Chan: 5-bit logical channel number**
Channel number 0x1f is reserved for control information. See "Control Channel" below. Other channels are "data channels". Each data channel is logically independent of the others. A data channel payload field contains a sequence of homogeneous samples. The format of the samples is determined by the configuration associated with the given channel. It is often the case that the payload field contains 32-bit complex samples, each containing 16-bit real and imaginary components.

**Tag**
4-bit tag for matching IN packets with OUT packets.

**Payload Len**
9-bit field that specifies the length of the payload field in bytes. Must be in the range 0 to 504 inclusive.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

**Timestamp: 32-bit timestamp**
On IN packets, the timestamp indicates the time at which the first sample of the packet was produced by the A/D converter(s) for that channel. On OUT packets, the timestamp specifies the time at which the first sample in the packet should go out the D/A converter(s) for that channel. If a packet reaches the head of the transmit queue, and the current time is later than the timestamp, an error is assumed to have occurred and the packet is discarded. As a special case, the timestamp 0xffffffff is interpreted as "Now". The time base is a free running 32-bit counter that is incremented by the A/D sample-clock.

**Payload**
Variable length field Length is specified by the Payload Len field.

**Padding**
This field is 504 - Payload Len bytes long, and its content is unspecified. This field pads the packet out to a constant 512 bytes.

#### 15.1.1.1 "Data Channel" payload format

If `Chan != 0x1f`, the packet is a "data packet" and the payload is a sequence of homogeneous samples. The format of the samples is determined by the configuration associated with the given channel. It is often the case that the payload field contains 32-bit complex samples, each containing 16-bit real and imaginary components.

#### 15.1.1.2 "Control Channel" payload format

If `Chan == 0x1f`, the packet is a "control packet". The control channel payload consists of a sequence of 0 or more sub-packets.

Each sub-packet starts on a 32-bit boundary, and consists of an 8-bit Opcode field, an 8-bit Length field, Length bytes of arguments, and 0, 1, 2 or 3 bytes of padding to align the tail of the sub-packet to a 32-bit boundary.

Control channel packets shall be processed at the head of the queue, and shall observe the timestamp semantics described above.

#### 15.1.1.3 General sub-packet format

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-//-+-+-+-+-+-+-+-+
|     Opcode    |    Length     |        <length bytes> ...     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-//-+-+-+-+-+-+-+-+
```

#### 15.1.1.4 Specific sub-packet formats

**RID: 6-bit Request-ID**
Copied from request sub-packet into corresponding reply sub-packet. RID allows the host to match requests and replies.

**Reg Number**
10-bit Register Number.

**Ping Fixed Length**

- Opcode: OP_PING_FIXED

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |       2       |    RID    |     Ping Value    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**Ping Fixed Length Reply**

- Opcode: OP_PING_FIXED_REPLY

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |       2       |    RID    |     Ping Value    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**Write Register**

- Opcode: OP_WRITE_REG

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |       6       |    mbz    |     Reg Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Register Value                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**Write Register Masked**

Only the register bits that correspond to 1's in the mask are written with the new value. `REG[Num] = (REG[Num] & ~Mask) | (Value & Mask)`

- Opcode: OP_WRITE_REG_MASKED

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |      10       |    mbz    |     Reg Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Register Value                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Mask Value                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**Read Register**

- Opcode: OP_READ_REG

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |       2       |    RID    |     Reg Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**Read Register Reply**

- Opcode: OP_READ_REG_REPLY

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |       6       |    RID    |     Reg Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Register Value                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**I2C Write**

- Opcode: OP_I2C_WRITE
- I2C Addr: 7-bit I2C address
- Data: The bytes to write to the I2C bus
- Length: Length of Data + 2

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |    Length     |       mbz       |   I2C Addr  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Data ...                                                   .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**I2C Read**

- Opcode: OP_I2C_READ
- I2C Addr: 7-bit I2C address
- Nbytes: Number of bytes to read from I2C bus

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |       3       |    RID    | mbz |   I2C Addr  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Nbytes    |              unspecified padding              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**I2C Read Reply**

- Opcode: OP_I2C_READ_REPLY
- I2C Addr: 7-bit I2C address
- Data: Length - 2 bytes of data read from I2C bus.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |     Length    |    RID    | mbz |   I2C Addr  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Data ...                                                   .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**SPI Write**

- Opcode: OP_SPI_WRITE
- Enables: Which SPI enables to assert (mask)
- Format: Specifies format of SPI data and Opt Header Bytes
- Opt Header Bytes: 2-byte field containing optional Tx bytes; see Format
- Data: The bytes to write to the SPI bus
- Length: Length of Data + 6

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |    Length     |              mbz              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Enables    |    Format     |        Opt Header Bytes       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Data ...                                                   .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**SPI Read**

- Opcode: OP_SPI_READ
- Enables: Which SPI enables to assert (mask)
- Format: Specifies format of SPI data and Opt Header Bytes

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

- Opt Header Bytes: 2-byte field containing optional Tx bytes; see Format
- Nbytes: Number of bytes to read from SPI bus.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |       7       |    RID    |        mbz        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Enables    |    Format     |        Opt Header Bytes       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Nbytes    |              unspecified padding              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**SPI Read Reply**

- Opcode: OP_SPI_READ_REPLY
- Data: Length - 2 bytes of data read from SPI bus.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |     Length    |    RID    |        mbz        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Data ...                                                   .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**Delay**

- Opcode: OP_DELAY
- Ticks: 16-bit unsigned delay count
- Delay Ticks clock ticks before executing next operation.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |       2       |             Ticks             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

## 15.2 Ettus B200

The binary *osmo-trx-uhd* is used to drive this device, see Section 16.1.

Comes only with 1 RF channel. It can still be used in a multi-TRX setup by using the Section 13.2 feature. By using this feature, one can drive up to 3 TRX (with the restrictions explained there).

## 15.3 Ettus B210

The binary *osmo-trx-uhd* is used to drive this device, see Section 16.1.

Comes with 2 RF channels, which can be used to set up a multi-TRX BTS. However, due to a shared local oscillator for both RF channels, ARFCN separation can be up about 25 MHz.

This device also supports the Section 13.2 feature. By using this feature, one can drive up to 3 TRX (with the restrictions explained there). Please note that the above configurations cannot be combined, which means maximum number of TRX one can achieve is 2 by using separate physical RF channels, or 3 by using multi-ARFCN method. You cannot support, for example, 6 ARFCN operation on B210 using 3 TRX on side A and another 3 TRX on side B.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

### 15.4 LimeSDR-USB

The binary *osmo-trx-lms* is used to drive this device, see Section 16.2.

This device comes with 2 RF channels, so it should theoretically be possible to run a multi-TRX setup with it, but there are yet no records that this kind of setup was tested with this device.

This device has 3 different Rx paths with different antenna connectors in the PCB, each with a different frequency and bandwidth range. One should make sure the physical antenna is connected to the correct connector matching the Rx path you want to use. If one wants to be able to use the device in both 900 and 1800 MHz GSM bands and easily switch between them, then Rx Path `LNAW` should be used ,since it is the only one covering both bands, and the antenna physically plugged accordingly. Following example shows how to then configure *osmo-trx-lms* to use that Rx path to read samples.

**Example: Configure osmo-trx-lms to use LNAW as Rx path and BAND1 as Tx Path**

```
trx
 ...
 chan 0
  tx-path BAND1
  rx-path LNAW
```

### 15.5 LimeSDR-mini

The binary *osmo-trx-lms* is used to drive this device, see Section 16.2.

As a smaller brother of the , this device comes only with 1 RF channel. As a result, it can only hold 1 TRX as of today.

## 16 OsmoTRX backend support

### 16.1 `osmo-trx-uhd` for UHD based Transceivers

This OsmoTRX model uses *libuhd* (UHD, USRP Hardware Driver) to drive the device, that is configuring it and reading/writing samples from/to it.

So far, this backend has been mostly used to drive devices such as the Ettus B200 family and Fairwaves UmTRX family, and used to be the default backend used for legacy @osmo-trx@ binary when per-backend binaries didn't exist yet.

Any device providing generic support for UHD should theoretically be able to be run through this backend without much effort, but practical experience showed that some devices don't play well with it, such as the LimeSDR family of devices, which showed far better results when using its native interface.

Related code can be found in the *Transceiver52M/device/uhd/* directory in *osmo-trx.git*.

### 16.2 `osmo-trx-lms` for LimeSuite based Transceivers

This OsmoTRX model uses LimeSuite API and library to drive the device, that is configuring it and reading/writing samples from/to it.

This backend was developed in order to be used together with LimeSDR-USB and LimeSDR-mini devices, due to to the poor results obtained with the UHD backend, and to simplify the stack.

Related code can be found in the *Transceiver52M/device/lms/* directory in *osmo-trx.git*.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

## 16.3 `osmo-trx-usrp1` for libusrp based Transceivers

This OsmoTRX model uses the legacy libusrp driver provided in GNU Radio 3.4.2.

As this code was dropped from GNU Radio at some point and was found very difficult to build, some work was done to create a standalone libusrp which can be nowadays found as a separate git repository together with other osmocom git repositories, in https://git.osmocom.org/libusrp/

Related code can be found in the *Transceiver52M/device/usrp1/* directory in *osmo-trx.git*.

The USRPDevice module is basically a driver that reads/writes packets to a USRP with two RFX900 daughterboards, board A is the Tx chain and board B is the Rx chain.

The `radioInterface` module is basically an interface between the transceiver and the USRP. It operates the basestation clock based upon the sample count of received USRP samples. Packets from the USRP are queued and segmented into GSM bursts that are passed up to the transceiver; bursts from the transceiver are passed down to the USRP.

The transceiver basically operates "layer 0" of the GSM stack, performing the modulation, detection, and demodulation of GSM bursts. It communicates with the GSM stack via three UDP sockets, one socket for data, one for control messages, and one socket to pass clocking information. The transceiver contains a priority queue to sort to-be-transmitted bursts, and a filler table to fill in timeslots that do not have bursts in the priority queue. The transceiver tries to stay ahead of the basestation clock, adapting its latency when underruns are reported by the radioInterface/USRP. Received bursts (from the radioInterface) pass through a simple energy detector, a RACH or midamble correlator, and a DFE-based demodulator.

---

**Note**

There's a `SWLOOPBACK` #define statement, where the USRP is replaced with a memory buffer. In this mode, data written to the USRP is actually stored in a buffer, and read commands to the USRP simply pull data from this buffer. This was very useful in early testing, and still may be useful in testing basic Transceiver and radioInterface functionality.

---

## 16.4 `osmo-trx-ipc` Inter Process Communication backend

This OsmoTRX model provides its own Inter Process Communication (IPC) interface to drive the radio device driver (from now on the Driver), allowing for third party processes to implement the lowest layer device-specific bits without being affected by copyleft licenses of OsmoTRX.

For more information on such interface, see section Section 20.

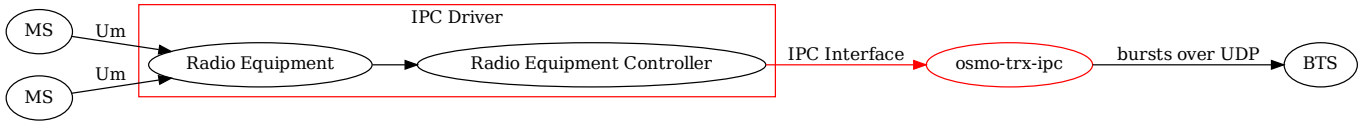



Figure 10: Architecture with *osmo-trx-ipc* and its IPC *Driver*

A sample config file for this OsmoTRX model can be found in *osmo-trx.git* doc/examples/osmo-trx-ipc/osmo-trx-ipc.cfg

In the config file, the following VTY command can be used to set up the IPC UD Master Socket *osmo-trx-ipc* will connect to at startup:

**Example: osmo-trx-ipc will connect to UD Master Socket /tmp/ipc_sock0 upon startup**

```
dev-args ipc_msock=/tmp/ipc_sock0
```

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

### 16.4.1 ipc-device-test

When built with `--with-ipc --with-uhd` configure options, *osmo-trx.git* will build the test program called *ipc-driver-test*. This program implements the *Driver* side of the osmo-trx-ipc interface (see Section 20 for more information) on one side, and also interacts internally with UHD (eg B210 as when using osmo-trx-uhd).

You can use this small program as a reference to:

- Test and experiment with *osmo-trx-ipc*.
- Write your own IPC *Driver* connecting to osmo-trx-ipc.

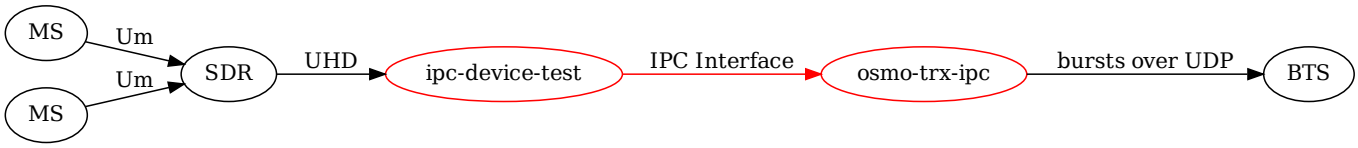



Figure 11: Architecture with *osmo-trx-ipc* and ipc-device-test as IPC *Driver*

The code for this app is found here:

- Transceiver52M/device/ipc/ipc-driver-test.h
- Transceiver52M/device/ipc/ipc-driver-test.c

Those files use the server-side (*Driver* side) code to operate the Posix Shared Memory region implemented in files `shm.c`, `shm.h`, `ipc_shm.c` and `ipc_shm.h` in the same directory.

Most of the code in that same directory is deliverately released under a BSD license (unlike most of *osmo-trx.git*), allowing third parties to reuse/recycle the code on their implemented *Driver* program no matter it being proprietary or under an open license. However, care must be taken with external dependencies, as for instance shm.c uses the talloc memory allocator, which is GPL licensed and hence cannot be used in a proprietary driver.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

# 17 Code Architecture

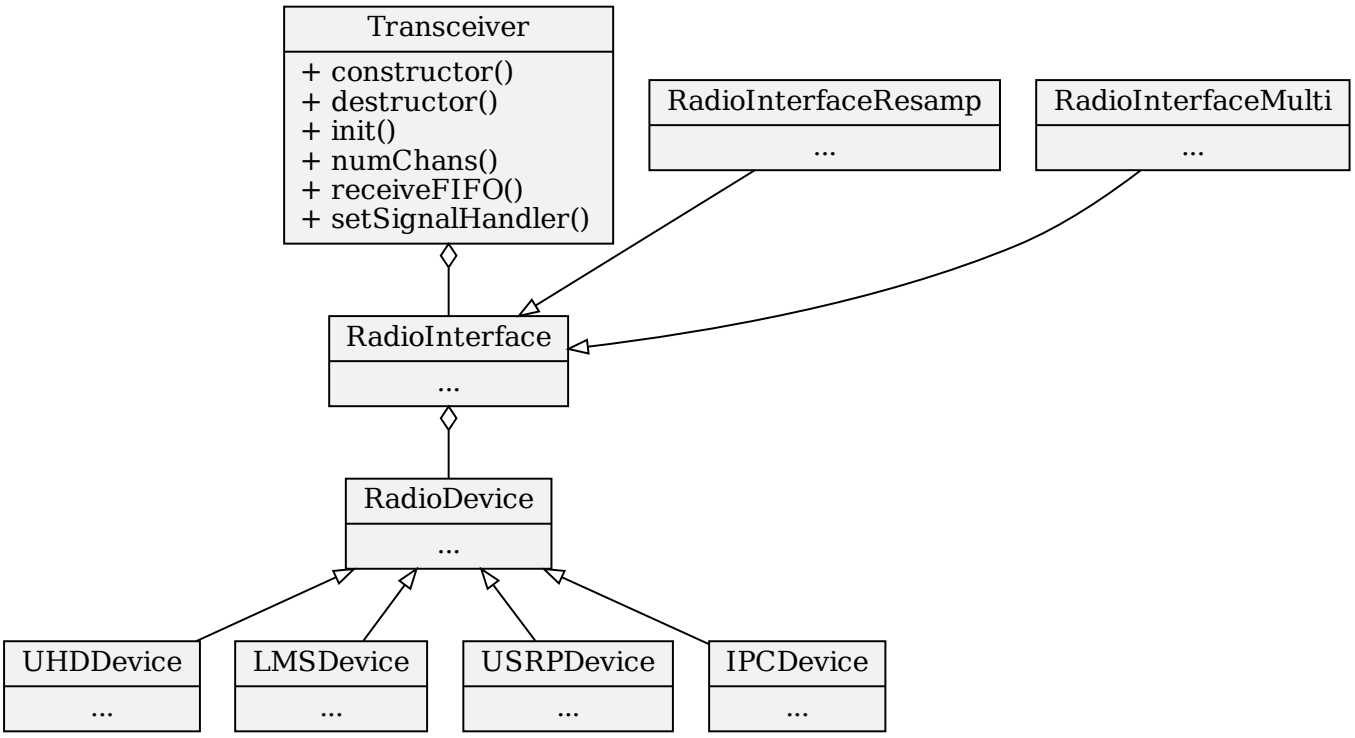



Figure 12: General overview of main OsmoTRX components

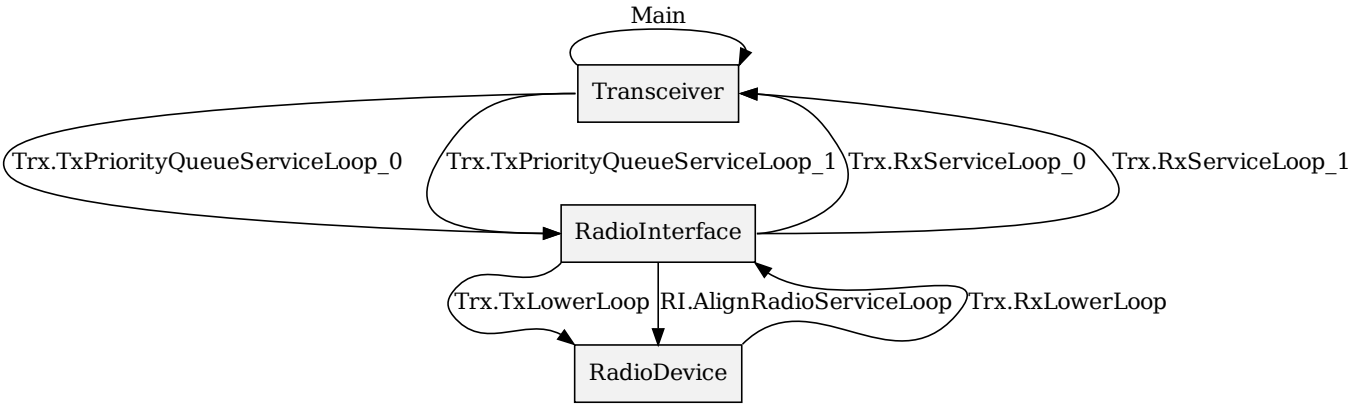



Figure 13: Example of thread architecture with OsmoTRX configured to use 2 logical RF channels (Trx=Transceiver, RI=RadioIface)

## 17.1 Transceiver

The Transceiver is the main component managing the other components running in the OsmoTRX process. There's a unique instance per process.

This class is quite complex from code point of view, as it starts lots of different threads and hence the interaction with this class from the outside is quite limited. Only interaction possible is to:

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

- `Transceiver()`: Create an instance through its constructor, at this time most configuration is handed to it.
- `init()`: Start running all the threads.
- `receiveFIFO()`: Attach a `radioInterface` channel FIFO in order to use it.
- `setSignalHandler()`: Used to set up a callback to receive certain events asynchronously from the Transceiver. No assumptions can be made about from which thread is the callback being called, which means multi-thread locking precautions may be required in certain cases, similar to usual signal handler processing. One important event received through this path is for instance when the Transceiver detected a fatal error which requires it to stop. Since it cannot stop itself (see destructor below), stopping procedure must be delegated to the user who created the instance.
- `~Transceiver()`: The destructor, which stops all running threads created at `init()` time. Destroying the object is the only way to stop the `Transceiver` completely, and must be called from a thread not managed by the `Transceiver`, otherwise it will deadlock. Usually it is stopped from the main thread, the one that called the constructor during startup.

During `init()` time, `Transceiver` will create a noticeable amount of threads, which may vary depending on the amount of RF channels requested.

Static amount of Threads (1 per `Transceiver` instance):

- `RxLowerLoop`: This thread is responsible for reading bursts from the `RadioInterface`, storing them into its FIFO and sending Clock Indications (Section 19.1) to *osmo-bts_trx*.
- `TxLowerLoop`: Manages pushing bursts from buffers in the FIFO into the `RadioInterface` at expected correct time based on the Transceiver clock.

Dynamic amount of Threads (1 per RF logical channel on the `Transceiver` instance):

- `RxServiceLoop`: Each thread of this type pulls bursts from the `RadioInterface` FIFO for one specific logical RF channel and handles it according to the slot and burst correlation type, finally sending proper data over the TRX Manager UDP socket (Section 19).
- `TxPriorityQueueServiceLoop`: Blocks reading from one ARFCN specific TRX Manager UDP socket (Section 19), and fills the `RadioInterface` with it setting clock related information.

All the Per-ARFCN Control Interface socket (Section 19.2) commands are handled by the event loop runnnig on the main thread. This is the only thread expected to use the private `start()` and `stop()` methods.

## 17.2 RadioInterface

The `RadioInterface` sits between the `Transceiver` and the `RadioDevice`, and provides extra features to the pipe like channelizers, resamplers, Tx/Rx synchronization on some devices, etc.

If the `RadioDevice` it drives requires it (only *USRP1* so far), the `RadioIntercace` will start and manage a thread internally called `AlignRadioServiceLoop` which will align current RX and TX timestamps.

Different features are offered through different `RadioInterface` subclasses which are selected based on configuration and device detected at runtime. Using these features may impact on the amount of CPU required to run the entire pipe.

### 17.2.1 RadioInterfaceResamp

This subclass of `RadioInterface` is automatically selected when some known specific UHD are to be used, since they require resampling to work properly. Some of this devices are for instance Ettus B100, USRP2 and X3XX models.

### 17.2.2 RadioInterfaceMulti

This subclass of `RadioInterface` is used when Section 13.2 is requested.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

### 17.3 RadioDevice

The `RadioDevice` class is responsible for driving the actual Hardware device. It is actually only an interface, and it is implemented in each backend which in turn becomes a specific OsmoTRX binary, see Section 16.

## 18 VTY Process and Thread management

Most Osmocom programs provide, some support to tune some system settings related to the running process, its threads, its scheduling policies, etc.

All of these settings can be configured through the VTY, either during startup by means of usual config files or through direct human interaction at the telnet VTY interface while the process is running.

### 18.1 Scheduling Policy

The scheduler to use as well as some of its properties (such as realtime priority) can be configured at any time for the entire process. This sort of functionality is useful in order to increase priority for processes running time-constrained procedures, such as those acting on the Um interface, like *osmo-trx* or *osmo-bts*, where use of this feature is highly recommended.

**Example: Set process to use RR scheduler**

```
cpu-sched
 policy rr 1 (1)
```

(1) Configure process to use *SCHED_RR* policy with real time priority 1

---

**Note**
Running a process under real time priority requires CAP_SYS_NICE capabilities. In order to avoid running the process as *root* user, you can either modify the binary permissions with `setcap "CAP_SYS_NICE+ep" /path/to/binary`, or allow the user running the program to set rtprio by adding a file */etc/security/limits.d/youruser_allow-rtprio.conf* with something like `youruser - rtprio 99`. See `man 7 sched` for more information on the topic.

---

### 18.2 CPU-Affinity Mask

Most operating systems allow for some sort of configuration on restricting the amount of CPUs a given process or thread can run on. The procedure is sometimes called as *cpu-pinning* since it allows to keep different processes pinned on a subset of CPUs to make sure the scheduler won't run two CPU-hungry processes on the same CPU.

The set of CPUs where each thread is allowed to run on is expressed by means of a bitmask in hexadecimal representation, where the right most bit relates to CPU 0, and the Nth most significant bit relates to CPU *N-1*. Setting the bit means the process is allowed to run on that CPU, while clearing it means the process is forbidden to run on that CPU.

Hence, for instance a cpu-affinity mask of *0x00* means the thread is not allowed on any CPU, which will cause the thread to stall until a new value is applied. A mask of *0x01* means the thread is only allowed to run on the 1st CPU (CPU 0). A mask of *0xff00* means CPUs 8-15 are allowed, while 0-7 are not.

For single-threaded processes (most of Osmocom are), it is usually enough to set this line in VTY config file as follows:

```
cpu-sched
 cpu-affinity self 0x01 (1)
```

(1) Allow main thread (the one managing the VTY) only on CPU 0

Or otherwise:

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

```
cpu-sched
 cpu-affinity all 0x01 ❶
```

❶ Allow all threads only on CPU 0

For multi-threaded processes, it may be desired to run some threads on a subset of CPUs while another subset may run on another one. In order to identify threads, one can either use the TID of the thread (each thread has its own PID in Linux), or its specific Thread Name in case it has been set by the application.

The related information on all threads available in the process can be listed through VTY. This allows identifying quickly the different threads, its current cpu-affinity mask, etc.

**Example: Get osmo-trx Thread list information from VTY**

```
OsmoTRX> show cpu-sched threads
Thread list for PID 338609:
 TID: 338609, NAME: 'osmo-trx-uhd', cpu-affinity: 0x3
 TID: 338610, NAME: 'osmo-trx-uhd', cpu-affinity: 0x3
 TID: 338611, NAME: 'osmo-trx-uhd', cpu-affinity: 0x3
 TID: 338629, NAME: 'osmo-trx-uhd', cpu-affinity: 0x3
 TID: 338630, NAME: 'osmo-trx-uhd', cpu-affinity: 0x3
 TID: 338631, NAME: 'osmo-trx-uhd', cpu-affinity: 0x3
 TID: 338634, NAME: 'UHDAsyncEvent', cpu-affinity: 0x3
 TID: 338635, NAME: 'TxLower', cpu-affinity: 0x3
 TID: 338636, NAME: 'RxLower', cpu-affinity: 0x3
 TID: 338637, NAME: 'RxUpper0', cpu-affinity: 0x3
 TID: 338638, NAME: 'TxUpper0', cpu-affinity: 0x3
 TID: 338639, NAME: 'RxUpper1', cpu-affinity: 0x3
 TID: 338640, NAME: 'TxUpper1', cpu-affinity: 0x3
```

At runtime, one can change the cpu-affinity mask for a given thread identifying it by either TID or name:

**Example: Set CPU-affinity from VTY telnet interface**

```
OsmoTRX> cpu-affinity TxLower 0x02 ❶
OsmoTRX> cpu-affinity TxLower 0x03 ❷
```

❶ Allow thread named *TxLower* (*338635*) only on CPU 1

❷ Allow with TID *338636* (*RxLower*) only on CPU 0 and 1

Since thread names are set dynamically by the process during startup or at a later point after creating the thread itself, One may need to specify in the config file that the mask must be applied by the thread itself once being configured rather than trying to apply it immediately. To specify so, the *delay* keyword is using when configuring in the VTY. If the *delay* keyword is not used, the VTY will report and error and fail at startup when trying to apply a cpu-affinity mask for a yet-to-be-created thread.

**Example: Set CPU-affinity from VTY config file**

```
cpu-sched
 cpu-affinity TxLower 0x01 delay ❶
```

❶ Allow thread named *TxLower* (*338635*) only on CPU 1. It will be applied by the thread itself when created.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

# 19 TRX Manager UDP socket interface

This is the protocol used between `osmo-trx` (the transceiver) and `osmo-bts-trx` (the BTS or core).

Each TRX Manager UDP socket interface represents a single transceiver (ARFCN). Each of these channels is a pair of UDP sockets, one for control (`TRXC`) and one for data (`TRXD`). Additionally, there's a separate global socket managing the Master Clock Interface, shared among all channels.

Given a base port `B` (5700), and a set of channels `0..N`, the ports related to a channel `0 <= X <= N` are:

- The Master clock interface is located on port `P=B`.
- The `TRXC` interface for channel `X` is located on port `P=B+2X+1`
- The `TRXD` interface for channel `X` is located on port `P=B+2X+2`.

The corresponding interface for every socket is at `P+100` on the BTS side.

---

**Note**
Starting from TRXDv2, it's possible to use only one socket for all channels. In this case, the global `TRXD` interface for all channels shall be established on port `P=B+1`. See Section 19.3.4 for more details.

---

## 19.1 Indications on the Master Clock Interface

The master clock interface is output only (uplink, from the radio to the BTS). Messages are "indications".

CLOCK gives the current value of the transceiver clock to be used by the BTS. This message is usually sent around once per second (217 GSM frames), but can be sent at any time. The clock value is NOT the current transceiver time. It is a time setting that the BTS should use to give better packet arrival times. The initial clock value is taken randomly, and then increased over time as the transceiver submits downlink packets to the radio.

```
IND CLOCK <totalFrames>
```

## 19.2 TRXC protocol

The per-ARFCN control interface uses a command-response protocol. Each command has a corresponding response. Commands are sent in downlink direction (BTS → TRX), and responses are sent in uplink direction (TRX → BTS). Commands and responses are NULL-terminated ASCII strings.

Every command is structured this way:

```
CMD <cmdtype> [params]
```

The `<cmdtype>` is the actual command. Parameters are optional depending on the commands type.

Every response is of the form:

```
RSP <cmdtype> <status> [result]
```

The `<status>` is 0 for success and a non-zero error code for failure. Successful responses may include results, depending on the command type.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

### 19.2.1 Power Control

POWEROFF shuts off transmitter power and stops the demodulator.

```
CMD POWEROFF
RSP POWEROFF <status>
```

POWERON starts the transmitter and starts the demodulator. Initial power level is by default very low, unless set explicitly by SETPOWER/ADJPOWER beforehand while in POWEROFF state. This command fails if the transmitter and receiver are not yet tuned. This command fails if the transmit or receive frequency creates a conflict with another ARFCN that is already running. If the transceiver is already on, it answers successfully to this command.

```
CMD POWERON
RSP POWERON <status>
```

NOMTXPOWER is used by the BTS to retrieve the nominal output transmit power of the transceiver. SETPOWER/ADJPOWER attenuations (dB) are expected to be applied based on this value (dBm).

```
CMD NOMTXPOWER
RSP NOMTXPOWER <status> <dBm>
```

SETPOWER sets transmit power attenuation wrt the nominal transmit power of the transceiver, in dB.

```
CMD SETPOWER <dB>
RSP SETPOWER <status> <dB>
```

ADJPOWER adjusts by the given dB the transmit power attenuation of the transceiver. Response returns resulting transmit power attenuation wrt the nominal transmit power of the transceiver.

```
CMD ADJPOWER <dBStep>
RSP ADJPOWER <status> <dBLevel>
```

RFMUTE locks the RF transceiver, hence disabling emission and reception of information on Air interface of the channel associated to the TRXC connection the command is sent on. Parameter with value of 1 is used to mute, and value of 0 is used to unmute.

```
CMD RFMUTE <1|0>
RSP RFMUTE <status> <1|0>
```

### 19.2.2 Tuning Control

RXTUNE tunes the receiver to a given frequency in kHz. This command fails if the receiver is already running. (To re-tune you stop the radio, re-tune, and restart.) This command fails if the transmit or receive frequency creates a conflict with another ARFCN that is already running.

```
CMD RXTUNE <kHz>
RSP RXTUNE <status> <kHz>
```

TXTUNE tunes the transmitter to a given frequency in kHz. This command fails if the transmitter is already running. (To re-tune you stop the radio, re-tune, and restart.) This command fails if the transmit or receive frequency creates a conflict with another ARFCN that is already running.

```
CMD TXTUNE <kHz>
RSP TXTUNE <status> <kHz>
```

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

#### 19.2.3 Training Sequence configuration

The usual way to configure all timeslots at once involves sending of the SETTSC command with a desired Training Sequence Code <tsc>:

```
CMD SETTSC <tsc>
CMD SETTSC <status> <tsc>
```

This command instructs the transceiver to use the given Training Sequence Code from the TSC set 1 (see 3GPP TS 45.002, table 5.2.3a) for Normal Burst detection on the receive path. It does not affect the transmit path because bursts coming from the BTS do contain the Training Sequence bits.

#### 19.2.4 Timeslot Control

SETSLOT sets the format of a given uplink timeslot in the ARFCN. The <timeslot> indicates the timeslot of interest. The <chantype> indicates the type of channel that occupies the timeslot. A chantype of zero indicates the timeslot is off.

```
CMD SETSLOT <timeslot> <chantype>
RSP SETSLOT <status> <timeslot> <chantype>
```

Here's the list of channel combinations and related values (<chantype>):

Table 5: List of channel combinations and related values (<chantype>)

| value | Channel Combination |
|---|---|
| 0 | Channel is transmitted, but unused |
| 1 | TCH/FS |
| 2 | TCH/HS, idle every other slot |
| 3 | TCH/HS |
| 4 | Downlink: FCCH + SCH + CCCH + BCCH, Uplink: RACH |
| 5 | Downlink: FCCH + SCH + CCCH + BCCH + SDCCH/4 + SACCH/4, Uplink: RACH+SDCCH/4 |
| 6 | Downlink: CCCH+BCCH, Uplink: RACH |
| 7 | SDCCH/8 + SACCH/8 |
| 8 | TCH/F + FACCH/F + SACCH/M |
| 9 | TCH/F + SACCH/M |
| 10 | TCH/FD + SACCH/MD |
| 11 | PBCCH+PCCCH+PDTCH+PACCH+PTCCH |
| 12 | PCCCH+PDTCH+PACCH+PTCCH |
| 13 | PDTCH+PACCH+PTCCH |

##### 19.2.4.1 Multiple Training Sequences (optional)

For some setups it's insufficient to have a single Training Sequence Code assigned to all timeslots of a transceiver. It may be required to use different TSCs for some (or even all) timeslots. This can be achieved by sending SETSLOT command with additional arguments:

```
CMD SETSLOT <timeslot> <chantype> [ C<c>/S<s> ]
RSP SETSLOT <status> <timeslot> <chantype> [ C<c>/S<s> ]
```

where <c> is a Training Sequence Code from TSC set <s>.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

**Note**
The numbering of both Training Sequence Code and set shall be the same as in 3GPP TS 45.002, e.g. `C0S1` corresponds to the first sequence in the first TSC set for a chosen modulation type. TSC Set number 0 (`S0`) does not exist in the specs.

**Example: configuring timeslot 4 to use TCH/F and TSC 7 from set 1**

```
CMD SETSLOT 4 1 C7/S1
RSP SETSLOT 0 4 1 C7/S1
```

Unless explicitly configured as described above, all timeslots will be using the default Training Sequence Code and set configured with the `SETTSC` command.

#### 19.2.4.2 VAMOS enabled channel combinations (optional)

The BTS may at any time re-configure channel combination of a timeslot (primarily during channel activation) to activate or deactivate VAMOS mode in the transceiver. For this purpose, the following additional channel combinations shall be used:

Table 6: List of VAMOS enabled channel combinations and related values

| value | Channel Combination |
|---|---|
| VFF | V0(TCH/F) & V1(TCH/F), 2 channels total |
| VHH | V0(TCH/H0) & V1(TCH/H0) + V0(TCH/H1) & V1(TCH/H1), 4 channels total |
| VFH | V0(TCH/F) & V1(TCH/H0) + V0(TCH/F) & V1(TCH/H1), 3 channels total |
| HVHH | TCH/H0 + V0(TCH/H1) & V1(TCH/H1), 3 channels total (mixed) |

where both `V0` and `V1` define a *VAMOS pair*. Symbols `&` and + indicate simultaneous and sequential transmission in the TDMA domain respectively. Therefore a combination `V0(a) & V1(b)` indicates that both channels `a` and `b` are simultaneously active during a timeslot period.

**Example: VFF in time domain (2 channels)**

```
MS1: | V0(TCH/F) | V0(TCH/F) | V0(TCH/F) | V0(TCH/F) | ...
-----+-----------+-----------+-----------+-----------+-----------> TDMA frames
MS2: | V1(TCH/F) | V1(TCH/F) | V1(TCH/F) | V1(TCH/F) | ...
```

**Example: VHH in time domain (4 channels)**

```
MS1: | V0(TCH/H0) |            | V0(TCH/H0) |            | ...
MS2: |            | V0(TCH/H1) |            | V0(TCH/H1) | ...
-----+------------+------------+------------+------------+--------> TDMA frames
MS3: | V1(TCH/H0) |            | V1(TCH/H0) |            | ...
MS4: |            | V1(TCH/H1) |            | V1(TCH/H1) | ...
```

**Example: VFH in time domain (3 channels)**

```
MS1: |  V0(TCH/F) |  V0(TCH/F) |  V0(TCH/F) |  V0(TCH/F) | ...
-----+------------+------------+------------+------------+--------> TDMA frames
MS2: | V1(TCH/H0) |            | V1(TCH/H0) |            | ...
MS3: |            | V1(TCH/H1) |            | V1(TCH/H1) | ...
```

**Example: HVHH in time domain (3 channels)**

```
MS1: |    TCH/H0  |             |    TCH/H0  |             | ...
MS2: |             | V0(TCH/H1) |             | V0(TCH/H1) | ...
-----+------------+------------+-------------+------------+--------> TDMA frames
MS3: |             | V1(TCH/H1) |             | V1(TCH/H1) | ...
```

**Note**
Combination `HVHH` is special, because it allows the network to multiplex a legacy user device (`MS1`) with a pair of VAMOS capable devices (`MS2` and `MS3`) on the same timeslot, so the former (`MS1`) is neither required to support the new orthogonal TSC sets nor conform to DARP phase I or II (SAIC support).

For all VAMOS enabled channel combinations, it's **required** to specify Training Sequence Code and the TSC set values for all multiplexed subscribers. See 3GPP TS 45.002, table 5.2.3e for more details on TSC set selection.

**Example: configuring a timeslot to use VFF combination**

```
CMD SETSLOT <timeslot> VFF C0/S1 ❶ C0/S2 ❷
RSP SETSLOT <status> <timeslot> VFF C0/S1 C0/S2
```

❶ V0(TCH/F) is configured to use TSC 0 from set 1 (table 5.2.3a).

❷ V1(TCH/F) is configured to use TSC 0 from set 2 (table 5.2.3b).

**Example: configuring a timeslot to use VFF combination (legacy MS)**

```
CMD SETSLOT <timeslot> VFF C7/S1 ❶ C4/S1 ❷
RSP SETSLOT <status> <timeslot> VFF C7/S1 C4/S1
```

❶ V0(TCH/F) is configured to use TSC 7 from set 1 (table 5.2.3a).

❷ V1(TCH/F) is configured to use TSC 4 from set 1 (table 5.2.3a).

**Note**
Using Training Sequences from the same set for a *VAMOS pair* (in this example, `C7/S1 C4/S1`) is not recommended because of their bad cross-correlation properties. The only exception is when two legacy non-VAMOS capable phones are paired together and neither of them does support additional TSC sets.

**Example: configuring a timeslot to use VHH combination**

```
CMD SETSLOT <timeslot> VHH C1/S3 ❶ C1/S4 ❷
RSP SETSLOT <status> <timeslot> VHH C1/S3 C1/S4
```

❶ V0(TCH/H0) and V0(TCH/H1) are configured to use TSC 1 from set 3 (table 5.2.3c).

❷ V1(TCH/H0) and V1(TCH/H1) are configured to use TSC 1 from set 4 (table 5.2.3d).

**Example: configuring a timeslot to use VFH combination**

```
CMD SETSLOT <timeslot> VFH C2/S1 ❶ C2/S4 ❷
RSP SETSLOT <status> <timeslot> VFH C2/S1 C2/S4
```

❶ V0(TCH/F) is configured to use TSC 2 from set 1 (table 5.2.3a).

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

❷ V1(TCH/H0) and V1(TCH/H1) are configured to use TSC 2 from set 4 (table 5.2.3d).

**Example: configuring a timeslot to use HVHH combination**

```
CMD SETSLOT <timeslot> HVHH C0/S1 ❶ C0/S1 ❷ C0/S2 ❸
RSP SETSLOT <status> <timeslot> HVHH C0/S1 C0/S1 C0/S2
```

❶ Legacy TCH/H0 is configured to use TSC 0 from set 1 (table 5.2.3a).

❷ V0(TCH/H1) is configured to use TSC 0 from set 1 (table 5.2.3a).

❸ V1(TCH/H1) is configured to use TSC 0 from set 2 (table 5.2.3b).

**Note**
In the example for `HVHH`, legacy TCH/H0 does not belong to a *VAMOS pair*, so it can be configured to use any Training Sequence Code without restrictions.

#### 19.2.5 TRXD header version negotiation

Messages on DATA interface may have different formats, defined by a version number, which can be negotiated on the control interface. By default, the Transceiver will use the legacy header version (0). See Section 19.3.1.

The format negotiation can be initiated by the BTS using `SETFORMAT` command. If the requested version is not supported by the transceiver, status code of the response message should indicate a preferred (basically, the latest) version. The format of this message is the following:

```
CMD SETFORMAT <ver_req>
RSP SETFORMAT <ver_resp> <ver_req>
```

where:

- `<ver_req>` is the requested version (suggested by the BTS),
- `<ver_rsp>` is either the applied version if matches `<ver_req>`, or a preferred version if `<ver_req>` is not supported.

If the transceiver indicates `<ver_rsp>` different than `<ver_req>`, the BTS is supposed to re-initiate the version negotiation using the suggested `<ver_rsp>`. For example:

```
BTS -> TRX: CMD SETFORMAT 2
BTS <- TRX: RSP SETFORMAT 1 2

BTS -> TRX: CMD SETFORMAT 1
BTS <- TRX: RSP SETFORMAT 1 1
```

If no suitable `<ver_rsp>` is found, or the `<ver_req>` is incorrect, the status code in the response shall be `-1`.

As soon as `<ver_rsp>` matches `<ver_req>` in the response, the process of negotiation is complete. Changing the header version is supposed to be done before `POWERON`, but can be also done afterwards.

### 19.3 TRXD protocol

Messages on the data interface carry one or optionally multiple radio bursts (see Section 19.3.4) per one UDP datagram. Two kinds of TRXD PDU exist:

- `TRX -> L1` (from transceiver to the L1): Uplink messages received from the MS,
- `L1 -> TRX` (from the L1 to transceiver): Downlink messages sent to the MS.

Depending on the origin and the version indicator, PDUs may have different structure.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

### 19.3.1 PDU versioning

The format of a PDU, i.e. presence and ordering of certain fields, is determined by the version number indicated in the first octet. This is usually referred as `TRXDvN`, where `N` is the version number (e.g. TRXDv0 or TRXDv1). A version number indicates the message format to be used for both directions: `TRX -> L1` and `L1 -> TRX`. The same version shall be used for all messages in both directions, mixing in any way is not permitted.

The version negotiation is optionally initiated by the `L1` on the control interface, and expected to be performed before starting the transceiver (i.e. sending `POWERON` command). See Section 19.2.5.

The current header allows to distinguish up to 16 different versions. The following versions are defined so far:

- TRXDv0 - initial version of TRXD protocol, inherited as-is from OpenBTS project.
- TRXDv1 (proposed in July 2019):
  - Introduced the concept of protocol versioning;
  - Introduced NOPE / IDLE indications;
  - New field: MTS (Modulation and Training Sequence);
  - New field: C/I (Carrier-to-interface) ratio;
  - Downlink messages mostly unchanged.
- TRXDv2 (proposed in January 2021):
  - Introduced the concept of burst batching (many bursts in one message);
  - Changed the field ordering (facilitating aligned access);
  - New field: batching indicator;
  - New field: shadow indicator;
  - New field: TRX number;
  - New field: SCPIR for VAMOS.

### 19.3.2 Uplink PDU format

An Uplink TRXD PDU contains a demodulated burst with the associated measurements (signal strength, timing delay, etc.) and TDMA frame/timeslot number. Starting from TRXDv1, a PDU may contain no payload, indicating the upper layers that the transceiver was not able to demodulate a burst (e.g. due to bad signal quality or the lack of signal during IDLE TDMA frames).

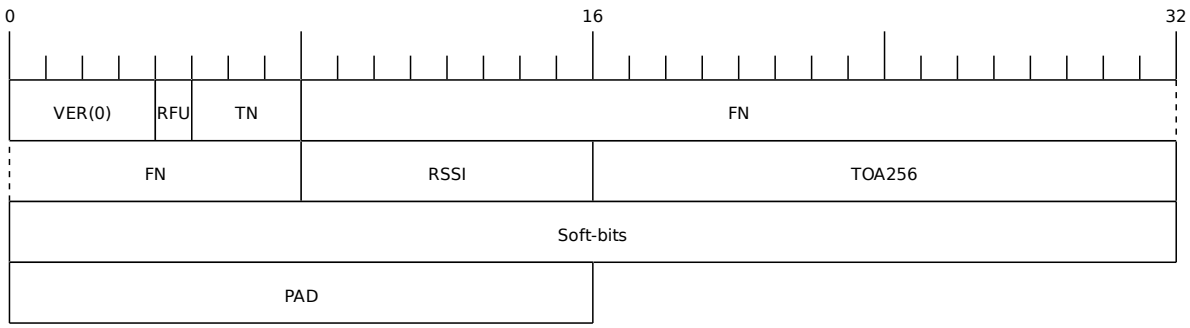



Figure 14: TRXDv0 Uplink data burst message structure

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

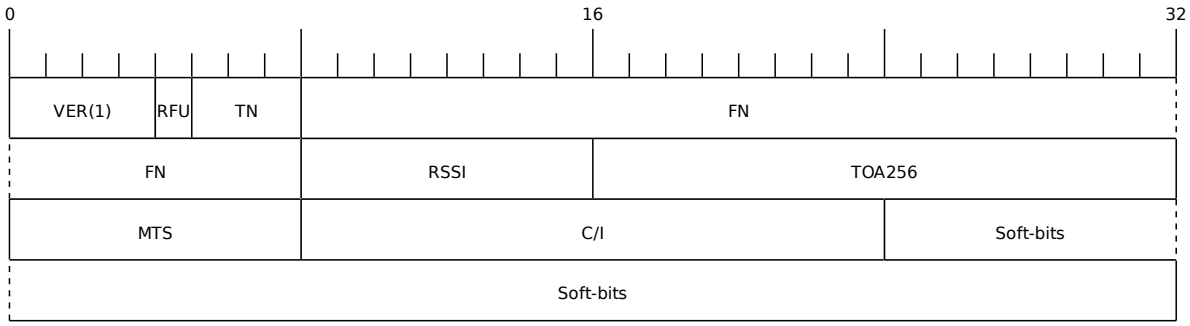



Figure 15: TRXDv1 Uplink data burst message structure

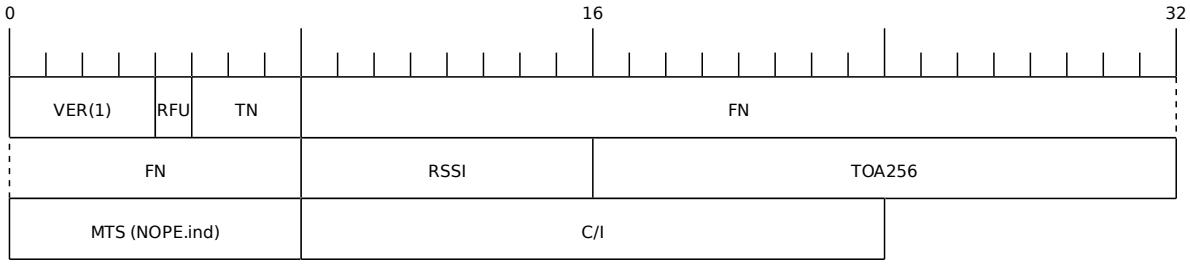



Figure 16: TRXDv1 NOPE / IDLE indication message structure

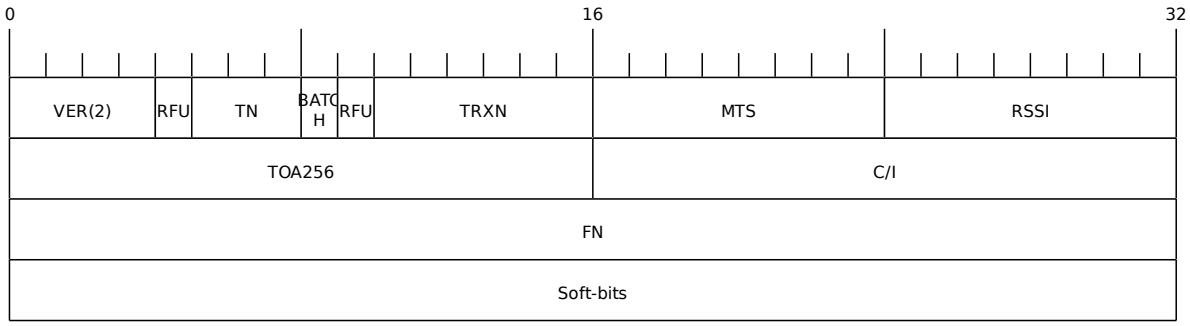



Figure 17: TRXDv2 Uplink message structure

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

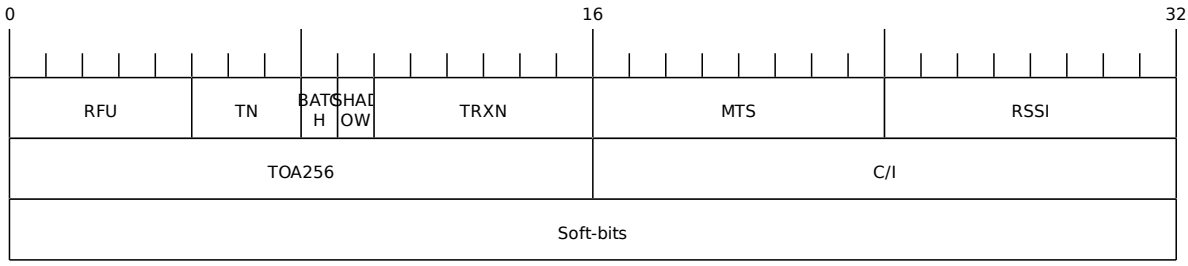



Figure 18: TRXDv2 Uplink message structure (batched part)

**VER: 4 bits**
TRXD header version, common for both `TRX -> L1` and `L1 -> TRX` directions.

**TN: 3 bits**
Timeslot number.

**RFU: variable bit-length**
Reserved for Future Use. The sending side of the PDU shall set all bits to '0'B; the receiving side shall ignore `RFU` fields.

**BATCH: 1 bit**
This bit indicates whether a batched PDU follows (see Section 19.3.4).

**SHADOW: 1 bit**
This bit indicates whether this is a *shadow PDU* (see Section 19.3.5).

**TRXN: 6 bits**
The transceiver (PHY channel) number this PDU is coming from.

**FN: 32 bits (4 bytes)**
GSM frame number, big endian.

**RSSI: 8 bits (1 byte)**
Received Signal Strength Indication in -dBm, encoded without the negative sign.

**TOA256: 16 bits (2 bytes)**
Timing of Arrival in units of 1/256 of symbol, big endian.

**MTS: 8 bits (1 byte)**
Contains the Modulation and Training Sequence information. See Section 19.3.2.1 for more information on the encoding.

**C/I: 16 bits (2 bytes)**
Contains the Carrier-to-Interference ratio in centiBels, big endian. The C/I value is computed from the training sequence of each burst, where the "ideal" training sequence is compared to the actual training sequence and the result expressed in centiBels.

**Soft-bits: 148 x N bytes (variable length, N defined by modulation type)**
Contains the uplink burst. Unlike the downlink bursts, the uplink bursts are designated using the soft-bits notation, so the receiver can indicate its assurance from 0 to -127 that a given bit is 1, and from 0 to +127 that a given bit is 0. The Viterbi algorithm allows to approximate the original sequence of hard-bits (1 or 0) using these values. Each soft-bit (-127..127) of the burst is encoded as an unsigned value in range (0..255) respectively using the constant shift. This way:

- 0 → definite "0"
- 255 → definite "1".

**PAD: 2 bytes (optional)**
Padding at the end, historical reasons (OpenBTS inheritance). Bits can take any value, but 0 is preferred. Only expected on TRXDv0 headers.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

#### 19.3.2.1 Coding of MTS: Modulation and Training Sequence info

3GPP TS 45.002 version 15.1.0 defines several modulation types, and a few sets of training sequences for each type. The most common are GMSK and 8-PSK (which is used in EDGE).

**MTS field structure**

```
+-----------------+-----------------------------------------+
| 7 6 5 4 3 2 1 0 | bit numbers (value range)               |
+-----------------+-----------------------------------------+
| X . . . . . . . | NOPE / IDLE frame indication (0 or 1)   |
+-----------------+-----------------------------------------+
| . X X X X . . . | Modulation, TS set number (see below)   |
+-----------------+-----------------------------------------+
| . . . . . X X X | Training / Synch. Sequence Code (0..7)  |
+-----------------+-----------------------------------------+
```

**NOPE / IDLE frame indication (referred to as NOPE.ind)**
The bit number 7 (MSB) shall be set to '1'B by the transceiver when either nothing has been detected, so the BTS scheduler keeps processing bursts without gaps, or during IDLE frames, so the current noise levels can be delivered. In this case the remaining bits become meaningless and shall be set to '0'B. The payload (`Soft-bits` or `Hard-bits`) is omited.

**Modulation and TS set number**
GMSK has 4 sets of training sequences (see tables 5.2.3a-d), while 8-PSK (see tables 5.2.3f-g) and the others have 2 sets. Access and Synchronization bursts also have several synchronization sequences.

**Modulation and TS set number**

```
+-----------------+-----------------------------------------+---------------+
| 7 6 5 4 3 2 1 0 | Description                             | Burst length  |
+-----------------+-----------------------------------------+---------------+
| . 0 0 X X . . . | GMSK, 4 TS sets (0..3)                  |   148 x 1     |
+-----------------+-----------------------------------------+---------------+
| . 0 1 0 X . . . | 8-PSK, 2 TS sets (0..1)                 |   148 x 3     |
+-----------------+-----------------------------------------+---------------+
| . 0 1 1 0 . . . | GMSK, Access Burst (see note)           |   148 x 1     |
+-----------------+-----------------------------------------+---------------+
| . 0 1 1 1 . . . | RFU (Reserved for Future Use)           |   -------     |
+-----------------+-----------------------------------------+---------------+
| . 1 0 0 X . . . | 16QAM, 2 TS sets (0..1)                 |   148 x 4     |
+-----------------+-----------------------------------------+---------------+
| . 1 0 1 X . . . | 32QAM, 2 TS sets (0..1)                 |   148 x 5     |
+-----------------+-----------------------------------------+---------------+
| . 1 1 X X . . . | AQPSK (Downlink), 4 TS sets (0..3)      |   148 x 2     |
+-----------------+-----------------------------------------+---------------+
```

---

**Note**
A radio block on PDCH is carried by the sequence of four Normal Bursts. The one exception is a special radio block occasionally used on the Uplink consisting of a sequence of four Access Bursts (see 3GPP TS 44.060). The transceiver shall use `0110` as the modulation type to indicate that.

---

**Training / Synch. Sequence Code**
In combination with a modulation type and a TS set number, this field uniquely identifies the Training Sequence of a received Normal Burst (see tables 5.2.3a-d) or Synchronization Burst (see table 5.2.5-3), or the Synch. Sequence of a received Access Burst (see table 5.2.7-3 and 5.2.7-4).

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

#### 19.3.3 Downlink Data Burst

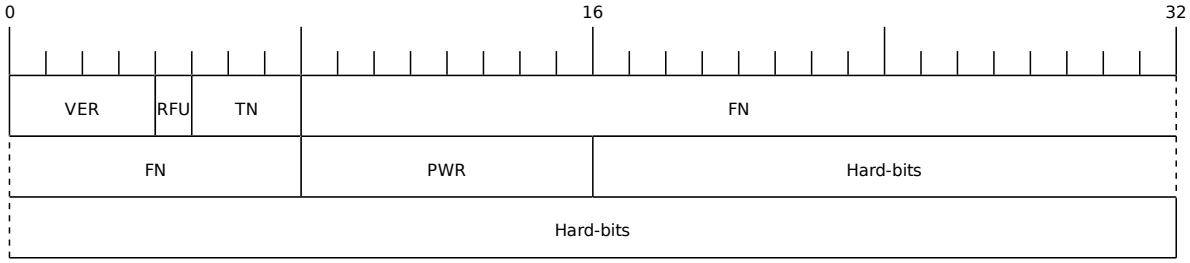



Figure 19: TRXDv0 and TRXDv1 Downlink data burst message structure

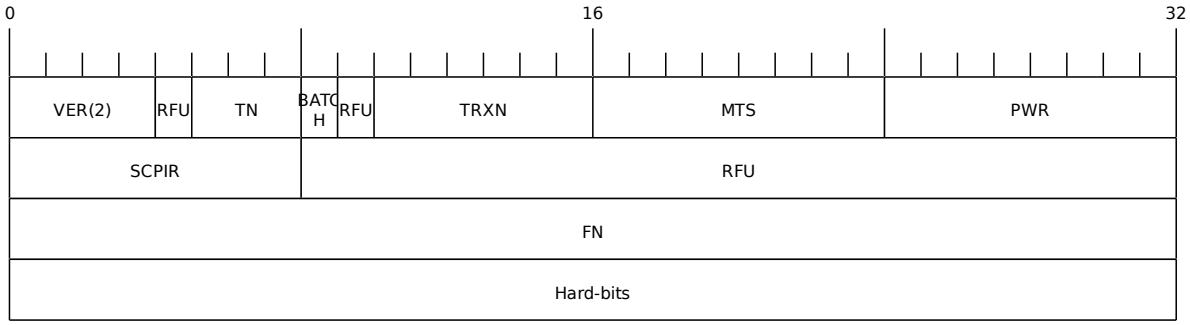



Figure 20: TRXDv2 Downlink data burst message structure

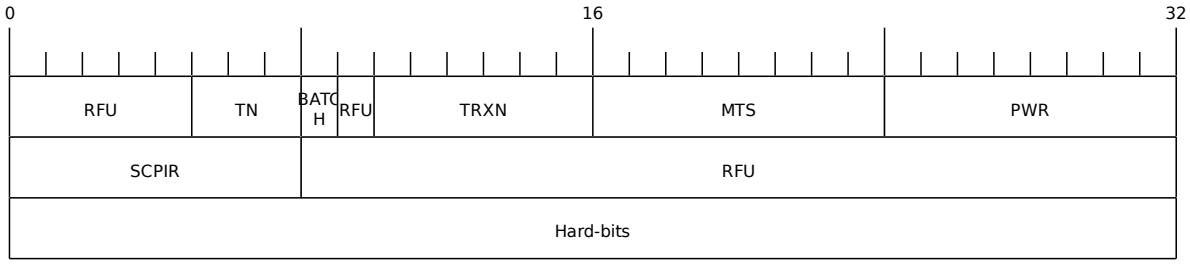



Figure 21: TRXDv2 Downlink PDU structure (batched part)

**VER: 4 bits**
TRXD header version, common for both `TRX -> L1` and `L1 -> TRX` directions.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

**TN: 3 bits**
Timeslot number.

**RFU: variable bit-length**
Reserved for Future Use. The sending side of the PDU shall set all bits to '0'B; the receiving side shall ignore `RFU` fields.

**BATCH: 1 bit**
This bit indicates whether a batched PDU follows (see Section 19.3.4).

**TRXN: 6 bits**
The transceiver (PHY channel) number this PDU is addressed to.

**MTS: 8 bits (1 byte)**
Contains the Modulation and Training Sequence information. See Section 19.3.2.1 for more information on the encoding.

**FN: 32 bits (4 bytes)**
GSM frame number, big endian.

**PWR: 8 bits (1 byte)**
Contains the relative (to the full-scale amplitude) transmit power **reduction** in dB. The absolute value is set on the control interface, so the resulting power is calculated as follows: `full_scale - (absolute_red + relative_red)`.

**SCPIR: 8 bits (1 byte)**
SCPIR (Subchannel Power Imbalance Ratio) - the ratio of power between Q and I channels for a VAMOS pair. This field shall be present when `MTC` field indicates the use of `AQPSK` modulation. Otherwise, all bits shall be set to '0'B. The value is a signed integer with a valid range: -10..10 dB.

**Hard-bits: 148 x N bytes (variable length, N defined by modulation type)**
Contains the downlink burst. Each hard-bit (1 or 0) of the burst is represented using one byte (0x01 or 0x00 respectively).

#### 19.3.4 PDU batching

Starting from TRXDv2, it's possible to combine several PDUs into a single datagram - this is called *PDU batching*. The purpose of *PDU batching* is to reduce socket load and eliminate possible PDU reordering, especially in a multi-TRX setup.

All *batched PDUs* in a datagram must belong to the same TDMA frame number indicated in the first part. The ordering of PDUs in a datagram may be different from the examples below, however it's recommended to batch PDUs in ascending order determined by TDMA timeslot number and/or `TRXN`.

The following PDU combinations in a datagram are possible:

- `a)` one datagram contains PDUs with the same TDMA timeslot number for all transceivers (total N PDUs per a TDMA timeslot);
- one datagram contains complete TDMA frame with PDUs for all 8 timeslots:
  - `b)` either for a single transceiver (total 8 PDUs per a TDMA frame),
  - `c)` or for all transceivers (total 8 x N PDUs per a TDMA frame).

None of these combinations are mandatory to support.

**Note**
Automatic negotiation of the batching algorithm(s) is not yet specified. Currently both sides need to be manually configured to use *PDU batching*.

**Note**
Size of the biggest possible TRXD datagram should be less than the *MTU (Maximum Transmission Unit)* of the network interface connecting both BTS and the transceiver. Otherwise the datagram is split across multiple IP packets, which may negatively affect performance.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

**Example: datagram structure for combination a)**

```
+--------+----------------+---------+-----------------------+
| TRXN=0 | TDMA FN=F TN=T | BATCH=1 | Hard-/Soft-bits       |
+--------+----------------+---------+-----------------------+
| TRXN=1 | TDMA FN=F TN=T | BATCH=1 | Hard-/Soft-bits       |
+--------+----------------+---------+-----------------------+
| TRXN=2 | TDMA FN=F TN=T | BATCH=1 | Hard-/Soft-bits       |
+--------+----------------+---------+-----------------------+
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
+--------+----------------+---------+-----------------------+
| TRXN=N | TDMA FN=F TN=T | BATCH=0 | Hard-/Soft-bits       |
+--------+----------------+---------+-----------------------+
```

**Example: datagram structure for combination b)**

```
+--------+----------------+---------+-----------------------+
| TRXN=N | TDMA FN=F TN=0 | BATCH=1 | Hard-/Soft-bits       |
+--------+----------------+---------+-----------------------+
| TRXN=N | TDMA FN=F TN=1 | BATCH=1 | Hard-/Soft-bits       |
+--------+----------------+---------+-----------------------+
| TRXN=N | TDMA FN=F TN=2 | BATCH=1 | Hard-/Soft-bits       |
+--------+----------------+---------+-----------------------+
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
+--------+----------------+---------+-----------------------+
| TRXN=N | TDMA FN=F TN=7 | BATCH=0 | Hard-/Soft-bits       |
+--------+----------------+---------+-----------------------+
```

**Example: datagram structure for combination c)**

```
+--------+----------------+---------+-----------------------+
| TRXN=0 | TDMA FN=F TN=0 | BATCH=1 | Hard-/Soft-bits       |
+--------+----------------+---------+-----------------------+
| TRXN=0 | TDMA FN=F TN=1 | BATCH=1 | Hard-/Soft-bits       |
+--------+----------------+---------+-----------------------+
| TRXN=0 | TDMA FN=F TN=2 | BATCH=1 | Hard-/Soft-bits       |
+--------+----------------+---------+-----------------------+
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
+--------+----------------+---------+-----------------------+
| TRXN=N | TDMA FN=F TN=6 | BATCH=1 | Hard-/Soft-bits       |
+--------+----------------+---------+-----------------------+
| TRXN=N | TDMA FN=F TN=7 | BATCH=0 | Hard-/Soft-bits       |
+--------+----------------+---------+-----------------------+
```

#### 19.3.5 Coding of VAMOS PDUs

In VAMOS mode, the throughput of a cell is increased by multiplexing two subscribers on a single TDMA timeslot. Basically, **two** bursts are getting transmitted during one TDMA timeslot period, and both of them need delivered over the TRXD interface.

In the Downlink direction, the two bursts belonging to a *VAMOS pair* shall be concatenated together and sent in one TRXD PDU. The resulting hard-bit sequence shall **not** be interleaved: `V0(0..147) + V1(0..147)` (296 hard-bits total), i.e. one complete burst for subscriber `V0` takes the first 148 bytes, and another complete burst for subscriber `V1` takes the remaining 148 bytes. The `MTS` field shall indicate the use of `AQPSK` modulation, and the `SCPIR` field shall indicate the Power Imbalance Ratio between `V0` and `V1`.

**Example: Downlink datagram containing a VAMOS PDU**

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
+--------+----------------+-----------+---------------------------------------+
| TRXN=N | TDMA FN=F TN=T | Mod=AQPSK | Hard-bits: V0(0..147) + V1(0..147)    |
+--------+----------------+-----------+---------------------------------------+
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

In the Uplink direction though, one or even both of the two bursts may be lost (e.g. due to high noise figures), so they shall always be sent in two separate PDUs. The missing bursts shall be substituted by NOPE indications, so it's always a pair of *batched PDUs*. First PDU in a pair is called *primary PDU*, the second is called *shadow PDU*. This is additionally indicated by the `SHADOW` field, which is set to '0'B and '1'B, respectively. The `MTS` field shall indicate the use of `GMSK` modulation if the burst is present.

**Example: Uplink datagram containing batched VAMOS PDUs (both present)**

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
+--------+---------------+----------+----------+-----------------------------+
| TRXN=N | TDMA FN=F TN=T | SHADOW=0 | Mod=GMSK | Soft-bits for V0 (148 bytes) |
+--------+---------------+----------+----------+-----------------------------+
| TRXN=N | TDMA FN=F TN=T | SHADOW=1 | Mod=GMSK | Soft-bits for V1 (148 bytes) |
+--------+---------------+----------+----------+-----------------------------+
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

**Example: Uplink datagram containing batched VAMOS PDUs (one lost)**

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
+--------+---------------+----------+----------+-----------------------------+
| TRXN=N | TDMA FN=F TN=T | SHADOW=0 | Mod=GMSK | Soft-bits for V0 (148 bytes) |
+--------+---------------+----------+----------+-----------------------------+
| TRXN=N | TDMA FN=F TN=T | SHADOW=1 | NOPE.ind |
+--------+---------------+----------+----------+
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

# 20 osmo-trx-ipc IPC Interface

This interface is the one used by *osmo_trx_ipc* backend to communicate to a third party process in charge of driving the lowest layer device-specific bits (from now on the Driver).

It consists of a set of Unix Domain (UD) sockets for the control plane, plus a shared memory region for the data plane.

Related code can be found in the Transceiver52M/device/ipc/ directory in *osmo-trx.git*.

If you are a potential driver implementator, the various primitives and data structures are publicly available in header file Transceiver52M/device/ipc/shm.h.

## 20.1 Control plane

Control plane protocol is transmitted over Unix Domain (UD) sockets using message based primitives. Each primitive has a type identified by an integer, and each type of primitive has a number of extra attributes attached to it. The IPC interface consists of 2 types of UD sockets:

- *Master* UD socket: One per osmo-trx-ipc process.
- *Channel* UD socket: One for each channel managed by osmo-trx-ipc process.

The *Driver* is in all cases expected to take the server role when creating UD sockets, while *osmo-trx-ipc* takes the client role and connects to sockets provided by the driver.

## 20.2 Master UD socket

During startup, *osmo-trx-ipc* will try connecting to the *Driver* Master UD socket located in the path provided by its own (VTY) configuration. As a result, it means the *Driver* process must be running and listening on the Master UD socket before *osmo-trx-ipc* is started, otherwise *osmo-trx-ipc* will fail and exit.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

Once connected, *osmo-trx-ipc* will submit a `GREETING_REQ` message primitive announcing the maximum supported protocol version (first version ever is `1`, increasing over time).

The *Driver* shall then answer in `GREETING_CNF` message primitive with its own maximum supported version (<= version received), providing 0 if none is supported.

If *osmo-trx-ipc* receives back the requested version, then both sides agreed on the protocol version to use. If *osmo-trx-ipc* receives back a lower version, it shall decide to continue with version negotiation using a lower version, until a supported version or 0 is received. If finally 0 is received, *osmo-trx-ipc* will disconnect and exit with failure.

Once the version is negotiated (`v1` as of current date), *osmo-trx-ipc* will ask for device information and available characeristics to the *Driver* using the `INFO_REQ` message primitive.

The *Driver* shall then answer with a `INFO_CNF` message containing information, such as:

- String containing device description
- Available reference clocks,
- {rx,tx} I/Q scaling factors
- Maximum number of channels supported
- for each channel:
  - List of available {rx,tx} paths/antennas.
  - {min,max}{rx,tx} gains
  - Nominal transmit power

All the information received from the *Driver* during `INFO_CNF` will be used by *osmo-trx-ipc* to decide whether it can fullfil the requested configuration from the user, and proceed to open the device, or exit with a failure (for instance number of channels, referece clock or tx/rx antenna selected by the user cannot be fullfilled).

*osmo-trx-ipc* will then proceed to open the device and do an initial configuration using an `OPEN_REQ` message, where it will provide the *Driver* with the desired selected configuration (such as number of channels, rx/tx paths, clock reference, bandwidth filters, etc.).

The *Driver* shall then configure the device and send back a `OPEN_CNF` with:

- `return_code` integer attribute set to `0` on success or `!0` on error.
- Name of the Posix Shared Memory region where data plane is going to be transmitted.
- One path for each channel, containing the just-created UD socket to manage that channel (for instance by taking Master UD socket path and appending `_$chan_idx`).
- Path Delay: this is the loopback path delay in samples (= used as a timestamp offset internally by *osmo-trx-ipc*), this value contains the analog delay as well as the delay introduced by the digital filters in the fpga in the sdr devices, and is therefore device type and bandwidth/sample rate dependant. This can not be omitted, wrong values will lead to a *osmo-trx-ipc* that just doesn't detect any bursts.

Finally, *osmo-trx-ipc* will connect to each channel's UD socket (see next section).

Upon *osmo-trx-ipc* closing the UD master socket connection, the *Driver* shall go into *closed* state: stop all processing and instruct the device to power off.

---

**Tip**

See Transceiver52M/device/ipc/shm.h for the detailed definition of all the related message primitives and data types for this socket.

---

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

## 20.3 Channel UD Socket

This socket can be used by *osmo-trx-ipc* to start/stop data plane processing or change channel's parameters such as Rx/Tx Frequency, Rx/Tx gains, etc.

A channel can be either in *started* or *stopped* state. When a channel is created (during `OPEN_REQ` in the Master UD Socket), it's by default in *stopped* state. `START_REQ` and `STOP_REQ` messages control this state, and eventual failures can be reported through `START_CNF` and `STOP_CNF` by the *Driver*.

The message `START_REQ` instructs the *Driver* to start processing data in the data plane. Similary, `STOP_REQ` instructs the *Driver* to stop processing data in the data plane.

Some parameters are usually changed only when the channel is in stopped mode, for instance Rx/Tx Frequency.

**Tip**
See Transceiver52M/device/ipc/shm.h for the detailed definition of all the related message primitives and data types for this socket.

## 20.4 Data Plane

Data plane protocol is implemented by means of a ring buffer structure on top of Posix Shared Memory (see `man 7 shm_overview`) between *osmo-trx-ipc* process and the *Driver*.

The Posix Shared Memory region is created and its memory structure prepared by the *Driver* and its name shared with *osmo-trx-ipc* during *OPEN_CNF* message in the Master UD Socket from the Control Plane. Resource allocation for the shared memory area and cleanup is up to the ipc server, as is mutex initialization for the buffers.

### 20.4.1 Posix Shared Memory structure

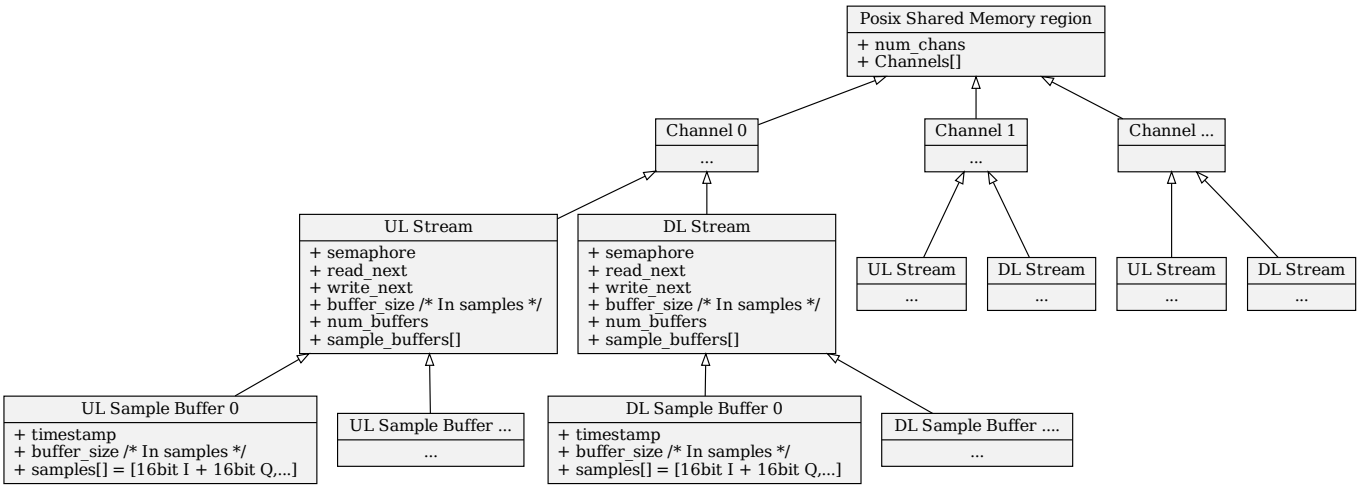



Figure 22: General overview of Posix Shared Memory structure

The Posix Shared Memory region contains an array of *Channels*.

Each *Channel* contains 2 Streams:

- Downlink *Stream*
- Uplink *Stream*

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

Each *Stream* handles a ring buffer, which is implemented as:

- An array of pointers to *Sample Buffer* structures.
- Variables containing the number of buffers in the array, as well as the maximum size in samples for each Sample Buffer.
- Variables containing `next_read` and `next_write` *Sample Buffer* (its index in the array of pointers).
- Unnamed Posix semaphores to do the required locking while using the ring buffer.

Each *Sample Buffer* contains:

- A `timestamp` variable, containing the position in the stream of the first sample in the buffer
- A `data_len` variable, containing the amount of samples available to process in the buffer
- An array of samples of size specified by the stream struct it is part of.

#### 20.4.2 Posix Shared Memory format

The Posix Shared memory region shall be formatted applying the following considerations:

- All pointers in the memory region are encoded as offsets from the start address of the region itself, to allow different processes with different address spaces to decode them.
- All structs must be force-aligned to 8 bytes
- Number of buffers must be power of 2 (2,4,8,16,...) - 4 appears to be plenty
- IQ samples format: One (complex) sample consists of 16bit i + 16bit q, so the buffer size is number of IQ pairs.
- A reasonable per-buffer size (in samples) is 2500, since this happens to be the ususal TX (downlink) buffer size used by *osmo-trx-ipc* with the b210 (rx over-the-wire packet size for the b210 is 2040 samples, so the larger value of both is convenient).

**Tip**
See Transceiver52M/device/ipc/shm.h for the detailed definition of all the objects being part of the Posix Shared memory region structure

#### 20.4.3 Posix Shared Memory procedures

The queue in the shared memory area is not supposed to be used for actual buffering of data, only for exchange, so the general expectation is that it is mostly empty. The only exception to that might be minor processing delays, and during startup.

Care must be taken to ensure that only timed waits for the mutex protecting it and the condition variables are used, in order to ensure that no deadlock occurs should the other side die/quit.

Thread cancellation should be disabled during reads/writes from/to the queue. In general a timeout can be considered a non recoverable error during regular processing after startup, at least with the current timeout value of one second.

Should over- or underflows occur a corresponding message should be sent towards *osmo-trx-ipc*.

Upon **read** of `N` samples, the reader does something like:

1. Acquire the semaphore in the channel's stream object.
2. Read `stream->next_read`, if `next_read==next_write`, become blocked in another sempahore (unlocking the previous one) until writer signals us, then `buff = stream->buffers[next_read]`

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

3. Read `buff->data_len` samples, reset the buffer data (`data_len=0`), increment `next_read` and if read samples is `<N`, continue with next buffer until `next_read==next_write`, then block again or if timeout elapsed, then we reach conditon buffer underflow and `return len < N`.
4. Release the semaphore

Upon **write** of `N` samples, the writer does something like:

1. Acquire the semapore in the channel's stream object.
2. Write samples to `buff = stream->buffers[next_write]`. If `data_len!=0`, signal `buffer_overflow` (increase field in stream object) and probably increase next_read`.
3. Increase `next_write`.
4. If `next_write` was `== next_read`, signal the reader through the other semaphore that it can continue reading.

# 21 Glossary

**2FF**
2nd Generation Form Factor; the so-called plug-in SIM form factor

**3FF**
3rd Generation Form Factor; the so-called microSIM form factor

**3GPP**
3rd Generation Partnership Project

**4FF**
4th Generation Form Factor; the so-called nanoSIM form factor

**A Interface**
Interface between BTS and BSC, traditionally over E1 (*3GPP TS 48.008* [3gpp-ts-48-008])

**A3/A8**
Algorithm 3 and 8; Authentication and key generation algorithm in GSM and GPRS, typically COMP128v1/v2/v3 or MILENAGE are typically used

**A5**
Algorithm 5; Air-interface encryption of GSM; currently only A5/0 (no encryption), A5/1 and A5/3 are in use

**Abis Interface**
Interface between BTS and BSC, traditionally over E1 (*3GPP TS 48.058* [3gpp-ts-48-058] and *3GPP TS 52.021* [3gpp-ts-52-021])

**ACC**
Access Control Class; every BTS broadcasts a bit-mask of permitted ACC, and only subscribers with a SIM of matching ACC are permitted to use that BTS

**AGCH**
Access Grant Channel on Um interface; used to assign a dedicated channel in response to RACH request

**AGPL**
GNU Affero General Public License, a copyleft-style Free Software License

**AQPSK**
Adaptive QPSK, a modulation scheme used by VAMOS channels on Downlink

**ARFCN**
Absolute Radio Frequency Channel Number; specifies a tuple of uplink and downlink frequencies

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

**AUC**
Authentication Center; central database of authentication key material for each subscriber

**BCCH**
Broadcast Control Channel on Um interface; used to broadcast information about Cell and its neighbors

**BCC**
Base Station Color Code; short identifier of BTS, lower part of BSIC

**BTS**
Base Transceiver Station

**BSC**
Base Station Controller

**BSIC**
Base Station Identity Code; 16bit identifier of BTS within location area

**BSSGP**
Base Station Subsystem Gateway Protocol (*3GPP TS 48.018* [3gpp-ts-48-018])

**BVCI**
BSSGP Virtual Circuit Identifier

**CBC**
Cell Broadcast Centre; central entity of Cell Broadcast service

**CBCH**
Cell Broadcast Channel; used to transmit Cell Broadcast SMS (SMS-CB)

**CBS**
Cell Broadcast Service

**CBSP**
Cell Broadcast Service Protocol (*3GPP TS 48.049* [3gpp-ts-48-049])

**CC**
Call Control; Part of the GSM Layer 3 Protocol

**CCCH**
Common Control Channel on Um interface; consists of RACH (uplink), BCCH, PCH, AGCH (all downlink)

**Cell**
A cell in a cellular network, served by a BTS

**CEPT**
Conférence européenne des administrations des postes et des télécommunications; European Conference of Postal and Telecommunications Administrations.

**CGI**
Cell Global Identifier comprised of MCC, MNC, LAC and BSIC

**CSFB**
Circiut-Switched Fall Back; Mechanism for switching from LTE/EUTRAN to UTRAN/GERAN when circuit-switched services such as voice telephony are required.

**dB**
deci-Bel; relative logarithmic unit

**dBm**
deci-Bel (milliwatt); unit of measurement for signal strength of radio signals

**DHCP**
Dynamic Host Configuration Protocol (*IETF RFC 2131* [ietf-rfc2131])

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

**downlink**
Direction of messages / signals from the network core towards the mobile phone

**DSCP**
Differentiated Services Code Point (*IETF RFC 2474* [ietf-rfc2474])

**DSP**
Digital Signal Processor

**dvnixload**
Tool to program UBL and the Bootloader on a sysmoBTS

**EDGE**
Enhanced Data rates for GPRS Evolution; Higher-speed improvement of GPRS; introduces 8PSK

**EGPRS**
Enhanced GPRS; the part of EDGE relating to GPRS services

**EIR**
Equipment Identity Register; core network element that stores and manages IMEI numbers

**ESME**
External SMS Entity; an external application interfacing with a SMSC over SMPP

**ETSI**
European Telecommunications Standardization Institute

**FPGA**
Field Programmable Gate Array; programmable digital logic hardware

**Gb**
Interface between PCU and SGSN in GPRS/EDGE network; uses NS, BSSGP, LLC

**GERAN**
GPRS/EDGE Radio Access Network

**GFDL**
GNU Free Documentation License; a copyleft-style Documentation License

**GGSN**
GPRS Gateway Support Node; gateway between GPRS and external (IP) network

**GMSK**
Gaussian Minimum Shift Keying; modulation used for GSM and GPRS

**GPL**
GNU General Public License, a copyleft-style Free Software License

**Gp**
Gp interface between SGSN and GGSN; uses GTP protocol

**GPRS**
General Packet Radio Service; the packet switched 2G technology

**GPS**
Global Positioning System; provides a highly accurate clock reference besides the global position

**GSM**
Global System for Mobile Communications. ETSI/3GPP Standard of a 2G digital cellular network

**GSMTAP**
GSM tap; pseudo standard for encapsulating GSM protocol layers over UDP/IP for analysis

**GSUP**
Generic Subscriber Update Protocol. Osmocom-specific alternative to TCAP/MAP

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

**GT**
Global Title; an address in SCCP

**GTP**
GPRS Tunnel Protocol; used between SGSN and GGSN

**HLR**
Home Location Register; central subscriber database of a GSM network

**HNB-GW**
Home NodeB Gateway. Entity between femtocells (Home NodeB) and CN in 3G/UMTS.

**HPLMN**
Home PLMN; the network that has issued the subscriber SIM and has his record in HLR

**IE**
Information Element

**IMEI**
International Mobile Equipment Identity; unique 14-digit decimal number to globally identify a mobile device, optionally with a 15th checksum digit

**IMEISV**
IMEI software version; unique 14-digit decimal number to globally identify a mobile device (same as IMEI) plus two software version digits (total digits: 16)

**IMSI**
International Mobile Subscriber Identity; 15-digit unique identifier for the subscriber/SIM; starts with MCC/MNC of issuing operator

**IP**
Internet Protocol (*IETF RFC 791* [ietf-rfc791])

**IPA**
*ip.access GSM over IP* protocol; used to multiplex a single TCP connection

**Iu**
Interface in 3G/UMTS between RAN and CN

**IuCS**
Iu interface for circuit-switched domain. Used in 3G/UMTS between RAN and MSC

**IuPS**
Iu interface for packet-switched domain. Used in 3G/UMTS between RAN and SGSN

**LAC**
Location Area Code; 16bit identifier of Location Area within network

**LAPD**
Link Access Protocol, D-Channel (*ITU-T Q.921* [itu-t-q921])

**LAPDm**
Link Access Protocol Mobile (*3GPP TS 44.006* [3gpp-ts-44-006])

**LLC**
Logical Link Control; GPRS protocol between MS and SGSN (*3GPP TS 44.064* [3gpp-ts-44-064])

**Location Area**
Location Area; a geographic area containing multiple BTS

**LU**
Location Updating; can be of type IMSI-Attach or Periodic. Procedure that indicates a subscriber's physical presence in a given radio cell.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

**M2PA**
MTP2 Peer-to-Peer Adaptation; a SIGTRAN Variant (*RFC 4165* [ietf-rfc4165])

**M2UA**
MTP2 User Adaptation; a SIGTRAN Variant (*RFC 3331* [ietf-rfc3331])

**M3UA**
MTP3 User Adaptation; a SIGTRAN Variant (*RFC 4666* [ietf-rfc4666])

**MCC**
Mobile Country Code; unique identifier of a country, e.g. 262 for Germany

**MFF**
Machine-to-Machine Form Factor; a SIM chip package that is soldered permanently onto M2M device circuit boards.

**MGW**
Media Gateway

**MM**
Mobility Management; part of the GSM Layer 3 Protocol

**MNC**
Mobile Network Code; identifies network within a country; assigned by national regulator

**MNCC**
Mobile Network Call Control; Unix domain socket based Interface between MSC and external call control entity like osmo-sip-connector

**MNO**
Mobile Network Operator; operator with physical radio network under his MCC/MNC

**MO**
Mobile Originated. Direction from Mobile (MS/UE) to Network

**MS**
Mobile Station; a mobile phone / GSM Modem

**MSC**
Mobile Switching Center; network element in the circuit-switched core network

**MSC pool**
A number of redundant MSCs serving the same core network, which a BSC / RNC distributes load across; see also the "MSC Pooling" chapter in OsmoBSC's user manual [userman-osmobsc] and *3GPP TS 23.236* [3gpp-ts-23-236]

**MSISDN**
Mobile Subscriber ISDN Number; telephone number of the subscriber

**MT**
Mobile Terminated. Direction from Network to Mobile (MS/UE)

**MTP**
Message Transfer Part; SS7 signaling protocol (*ITU-T Q.701* [itu-t-q701])

**MVNO**
Mobile Virtual Network Operator; Operator without physical radio network

**NCC**
Network Color Code; assigned by national regulator

**NITB**
Network In The Box; combines functionality traditionally provided by BSC, MSC, VLR, HLR, SMSC functions; see OsmoNITB

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

**NRI**
Network Resource Indicator, typically 10 bits of a TMSI indicating which MSC of an MSC pool attached the subscriber; see also the "MSC Pooling" chapter in OsmoBSC's user manual [userman-osmobsc] and *3GPP TS 23.236* [3gpp-ts-23-236]

**NSEI**
NS Entity Identifier

**NVCI**
NS Virtual Circuit Identifier

**NWL**
Network Listen; ability of some BTS to receive downlink from other BTSs

**NS**
Network Service; protocol on Gb interface (*3GPP TS 48.016* [3gpp-ts-48-016])

**OCXO**
Oven Controlled Crystal Oscillator; very high precision oscillator, superior to a VCTCXO

**OML**
Operation & Maintenance Link (ETSI/*3GPP TS 52.021* [3gpp-ts-52-021])

**OpenBSC**
Open Source implementation of GSM network elements, specifically OsmoBSC, OsmoNITB, OsmoSGSN

**OpenGGSN**
Open Source implementation of a GPRS Packet Control Unit

**OpenVPN**
Open-Source Virtual Private Network; software employed to establish encrypted private networks over untrusted public networks

**Osmocom**
Open Source MObile COMmunications; collaborative community for implementing communications protocols and systems, including GSM, GPRS, TETRA, DECT, GMR and others

**OsmoBSC**
Open Source implementation of a GSM Base Station Controller

**OsmoNITB**
Open Source implementation of a GSM Network In The Box, combines functionality traditionally provided by BSC, MSC, VLR, HLR, AUC, SMSC

**OsmoSGSN**
Open Source implementation of a Serving GPRS Support Node

**OsmoPCU**
Open Source implementation of a GPRS Packet Control Unit

**OTA**
Over-The-Air; Capability of operators to remotely reconfigure/reprogram ISM/USIM cards

**PC**
Point Code; an address in MTP

**PCH**
Paging Channel on downlink Um interface; used by network to page an MS

**PCP**
Priority Code Point (*IEEE 802.1Q* [?])

**PCU**
Packet Control Unit; used to manage Layer 2 of the GPRS radio interface

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

**PDCH**
Packet Data Channel on Um interface; used for GPRS/EDGE signalling + user data

**PIN**
Personal Identification Number; a number by which the user authenticates to a SIM/USIM or other smart card

**PLMN**
Public Land Mobile Network; specification language for a single GSM network

**PUK**
PIN Unblocking Code; used to unblock a blocked PIN (after too many wrong PIN attempts)

**RAC**
Routing Area Code; 16bit identifier for a Routing Area within a Location Area

**RACH**
Random Access Channel on uplink Um interface; used by MS to request establishment of a dedicated channel

**RAM**
Remote Application Management; Ability to remotely manage (install, remove) Java Applications on SIM/USIM Card

**RF**
Radio Frequency

**RFM**
Remote File Management; Ability to remotely manage (write, read) files on a SIM/USIM card

**Roaming**
Procedure in which a subscriber of one network is using the radio network of another network, often in different countries; in some countries national roaming exists

**Routing Area**
Routing Area; GPRS specific sub-division of Location Area

**RR**
Radio Resources; Part of the GSM Layer 3 Protocol

**RSL**
Radio Signalling Link (*3GPP TS 48.058* [3gpp-ts-48-058])

**RTP**
Real-Time Transport Protocol (*IETF RFC 3550* [ietf-rfc3550]); Used to transport audio/video streams over UDP/IP

**SACCH**
Slow Associate Control Channel on Um interface; bundled to a TCH or SDCCH, used for signalling in parallel to active dedicated channel

**SCCP**
Signaling Connection Control Part; SS7 signaling protocol (*ITU-T Q.711* [itu-t-q711])

**SDCCH**
Slow Dedicated Control Channel on Um interface; used for signalling and SMS transport in GSM

**SDK**
Software Development Kit

**SGs**
Interface between MSC (GSM/UMTS) and MME (LTE/EPC) to facilitate CSFB and SMS.

**SGSN**
Serving GPRS Support Node; Core network element for packet-switched services in GSM and UMTS.

**SIGTRAN**
Signaling Transport over IP (*IETF RFC 2719* [ietf-rfc2719])

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

**SIM**
Subscriber Identity Module; small chip card storing subscriber identity

**Site**
A site is a location where one or more BTSs are installed, typically three BTSs for three sectors

**SMPP**
Short Message Peer-to-Peer; TCP based protocol to interface external entities with an SMSC

**SMSC**
Short Message Service Center; store-and-forward relay for short messages

**SS7**
Signaling System No. 7; Classic digital telephony signaling system

**SS**
Supplementary Services; query and set various service parameters between subscriber and core network (e.g. USSD, 3rd-party calls, hold/retrieve, advice-of-charge, call deflection)

**SSH**
Secure Shell; *IETF RFC 4250* [ietf-rfc4251] to 4254

**SSN**
Sub-System Number; identifies a given SCCP Service such as MSC, HLR

**STP**
Signaling Transfer Point; A Router in SS7 Networks

**SUA**
SCCP User Adaptation; a SIGTRAN Variant (*RFC 3868* [ietf-rfc3868])

**syslog**
System logging service of UNIX-like operating systems

**System Information**
A set of downlink messages on the BCCH and SACCH of the Um interface describing properties of the cell and network

**TCH**
Traffic Channel; used for circuit-switched user traffic (mostly voice) in GSM

**TCP**
Transmission Control Protocol; (*IETF RFC 793* [ietf-rfc793])

**TFTP**
Trivial File Transfer Protocol; (*IETF RFC 1350* [ietf-rfc1350])

**TOS**
Type Of Service; bit-field in IPv4 header, now re-used as DSCP (*IETF RFC 791* [ietf-rfc791])

**TRX**
Transceiver; element of a BTS serving a single carrier

**TS**
Technical Specification

**u-Boot**
Boot loader used in various embedded systems

**UBI**
An MTD wear leveling system to deal with NAND flash in Linux

**UBL**
Initial bootloader loaded by the TI Davinci SoC

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

**UDP**
User Datagram Protocol (*IETF RFC 768* [ietf-rfc768])

**UICC**
Universal Integrated Chip Card; A smart card according to *ETSI TR 102 216* [etsi-tr102216]

**Um interface**
U mobile; Radio interface between MS and BTS

**uplink**
Direction of messages: Signals from the mobile phone towards the network

**USIM**
Universal Subscriber Identity Module; application running on a UICC to provide subscriber identity for UMTS and GSM networks

**USSD**
Unstructured Supplementary Service Data; textual dialog between subscriber and core network, e.g. **100 → Your extension is 1234*

**VAMOS**
Voice services over Adaptive Multi-user channels on One Slot; an optional extension for GSM specified in Release 9 of 3GPP GERAN specifications (*3GPP TS 48.018* [3gpp-ts-48-018]) allowing two independent UEs to transmit and receive simultaneously on traffic channels

**VCTCXO**
Voltage Controlled, Temperature Compensated Crystal Oscillator; a precision oscillator, superior to a classic crystal oscillator, but inferior to an OCXO

**VLAN**
Virtual LAN in the context of Ethernet (*IEEE 802.1Q* [ieee-802.1q])

**VLR**
Visitor Location Register; volatile storage of attached subscribers in the MSC

**VPLMN**
Visited PLMN; the network in which the subscriber is currently registered; may differ from HPLMN when on roaming

**VTY**
Virtual TeletYpe; a textual command-line interface for configuration and introspection, e.g. the OsmoBSC configuration file as well as its telnet link on port 4242

# A Osmocom TCP/UDP Port Numbers

The Osmocom GSM system utilizes a variety of TCP/IP based protocols. The table below provides a reference as to which port numbers are used by which protocol / interface.

Table 7: TCP/UDP port numbers

| L4 Protocol | Port Number | Purpose | Software |
|---|---|---|---|
| UDP | 1984 | Osmux | osmo-mgw, osmo-bts |
| UDP | 2427 | MGCP GW | osmo-bsc_mgcp, osmo-mgw |
| TCP | 2775 | SMPP (SMS interface for external programs) | osmo-nitb |
| TCP | 3002 | A-bis/IP OML | osmo-bts, osmo-bsc, osmo-nitb |
| TCP | 3003 | A-bis/IP RSL | osmo-bts, osmo-bsc, osmo-nitb |
| TCP | 4227 | telnet (VTY) | osmo-pcap-client |
| TCP | 4228 | telnet (VTY) | osmo-pcap-server |
| TCP | 4236 | Control Interface | osmo-trx |

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

Table 7: (continued)

| L4 Protocol | Port Number | Purpose | Software |
|---|---|---|---|
| TCP | 4237 | telnet (VTY) | osmo-trx |
| TCP | 4238 | Control Interface | osmo-bts |
| TCP | 4239 | telnet (VTY) | osmo-stp |
| TCP | 4240 | telnet (VTY) | osmo-pcu |
| TCP | 4241 | telnet (VTY) | osmo-bts |
| TCP | 4242 | telnet (VTY) | osmo-nitb, osmo-bsc, cellmgr-ng |
| TCP | 4243 | telnet (VTY) | osmo-bsc_mgcp, osmo-mgw |
| TCP | 4244 | telnet (VTY) | osmo-bsc_nat |
| TCP | 4245 | telnet (VTY) | osmo-sgsn |
| TCP | 4246 | telnet (VTY) | osmo-gbproxy |
| TCP | 4247 | telnet (VTY) | OsmocomBB |
| TCP | 4249 | Control Interface | osmo-nitb, osmo-bsc |
| TCP | 4250 | Control Interface | osmo-bsc_nat |
| TCP | 4251 | Control Interface | osmo-sgsn |
| TCP | 4252 | telnet (VTY) | sysmobts-mgr |
| TCP | 4253 | telnet (VTY) | osmo-gtphub |
| TCP | 4254 | telnet (VTY) | osmo-msc |
| TCP | 4255 | Control Interface | osmo-msc |
| TCP | 4256 | telnet (VTY) | osmo-sip-connector |
| TCP | 4257 | Control Interface | osmo-ggsn, ggsn (OpenGGSN) |
| TCP | 4258 | telnet (VTY) | osmo-hlr |
| TCP | 4259 | Control Interface | osmo-hlr |
| TCP | 4260 | telnet (VTY) | osmo-ggsn |
| TCP | 4261 | telnet (VTY) | osmo-hnbgw |
| TCP | 4262 | Control Interface | osmo-hnbgw |
| TCP | 4263 | Control Interface | osmo-gbproxy |
| TCP | 4264 | telnet (VTY) | osmo-cbc |
| TCP | 4265 | Control Interface | osmo-cbc |
| TCP | 4266 | D-GSM MS Lookup: mDNS serve | osmo-hlr |
| TCP | 4267 | Control Interface | osmo-mgw |
| TCP | 4268 | telnet (VTY) | osmo-uecups |
| SCTP | 4268 | UECUPS | osmo-uecups |
| TCP | 4269 | telnet (VTY) | osmo-e1d |
| TCP | 4270 | telnet (VTY) | osmo-isdntap |
| TCP | 4271 | telnet (VTY) | osmo-smlc |
| TCP | 4272 | Control Interface | osmo-smlc |
| TCP | 4273 | telnet (VTY) | osmo-hnodeb |
| TCP | 4274 | Control Interface | osmo-hnodeb |
| TCP | 4275 | telnet (VTY) | osmo-upf |
| TCP | 4276 | Control Interface | osmo-upf |
| TCP | 4277 | telnet (VTY) | osmo-pfcp-tool |
| TCP | 4278 | Control Interface | osmo-pfcp-tool |
| UDP | 4729 | GSMTAP | Almost every osmocom project |
| TCP | 5000 | A/IP | osmo-bsc, osmo-bsc_nat |
| UDP | 23000 | GPRS-NS over IP default port | osmo-pcu, osmo-sgsn, osmo-gbproxy |
| TCP | 48049 | BSC-CBC (CBSP) default port | osmo-bsc, osmo-cbc |

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

# B Bibliography / References

#### B.0.0.0.1 References


[1] [userman-ice1usb] Osmocom Project: icE1usb User Manual.

[2] [userman-ogt] Pau Espin: osmo-gsm-tester User Manual.

[3] [userman-remsim] Harald Welte: osmo-remsim User Manual.

[4] [osmobts-abis-spec] Neels Hofmeyr & Harald Welte. OsmoBTS Abis Protocol Specification. https://ftp.osmocom.org/docs/latest/osmobts-abis.pdf

[5] [userman-osmobsc] Osmocom Project: OsmoBSC User Manual. https://ftp.osmocom.org/docs/latest/osmobsc-usermanual.pdf

[6] [vty-ref-osmobsc] Osmocom Project: OsmoBSC VTY Reference Manual. https://ftp.osmocom.org/docs/-latest/osmobsc-vty-reference.pdf

[7] [userman-osmobts] Osmocom Project: OsmoBTS User Manual. https://ftp.osmocom.org/docs/latest/osmobts-usermanual.pdf

[8] [vty-ref-osmobts] Osmocom Project: OsmoBTS VTY Reference Manual. https://ftp.osmocom.org/docs/-latest/osmobts-trx-vty-reference.pdf https://ftp.osmocom.org/docs/latest/osmobts-sysmo-vty-reference.pdf https://ftp.osmocom.org/docs/latest/osmobts-lc15-vty-reference.pdf https://ftp.osmocom.org/docs/latest/-osmobts-oc2g-vty-reference.pdf https://ftp.osmocom.org/docs/latest/osmobts-octphy-vty-reference.pdf https://ftp.osmocom.org/docs/latest/osmobts-virtual-vty-reference.pdf

[9] [userman-osmocbc] Osmocom Project: OsmoCBC User Manual. https://ftp.osmocom.org/docs/latest/-osmocbc-usermanual.pdf

[10] [vty-ref-osmocbc] Osmocom Project: OsmoCBC VTY Reference Manual. https://ftp.osmocom.org/docs/-latest/osmocbc-vty-reference.pdf

[11] [userman-osmogbproxy] Osmocom Project: OsmoGBProxy User Manual. https://ftp.osmocom.org/docs/-latest/osmogbproxy-usermanual.pdf

[12] [vty-ref-osmogbproxy] Osmocom Project: OsmoGBPRoxy VTY Reference Manual. https://ftp.osmocom.org/-docs/latest/osmogbproxy-vty-reference.pdf

[13] [userman-osmoggsn] Osmocom Project: OpenGGSN User Manual. https://ftp.osmocom.org/docs/latest/-osmoggsn-usermanual.pdf

[14] [vty-ref-osmoggsn] Osmocom Project: OsmoGGSN VTY Reference Manual. https://ftp.osmocom.org/docs/-latest/osmoggsn-vty-reference.pdf

[15] [userman-osmohlr] Osmocom Project: OsmoHLR User Manual. https://ftp.osmocom.org/docs/latest/osmohlr-usermanual.pdf

[16] [vty-ref-osmohlr] Osmocom Project: OsmoHLR VTY Reference Manual. https://ftp.osmocom.org/docs/latest/-osmohlr-vty-reference.pdf

[17] [userman-osmohnbgw] Osmocom Project: OsmoHNBGW User Manual. https://ftp.osmocom.org/docs/latest/-osmohnbgw-usermanual.pdf

[18] [vty-ref-osmohnbgw] Osmocom Project: OsmoHNBGW VTY Reference Manual. https://ftp.osmocom.org/-docs/latest/osmohnbgw-vty-reference.pdf

[19] [userman-osmomgw] Osmocom Project: OsmoMGW User Manual. https://ftp.osmocom.org/docs/latest/-osmomgw-usermanual.pdf

[20] [vty-ref-osmomgw] Osmocom Project: OsmoMGW VTY Reference Manual. https://ftp.osmocom.org/docs/-latest/osmomgw-vty-reference.pdf


Copyright © 2018 sysmocom - s.f.m.c. GmbH 

[21] [userman-osmomsc] Osmocom Project: OsmoMSC User Manual. https://ftp.osmocom.org/docs/latest/-osmomsc-usermanual.pdf

[22] [vty-ref-osmomsc] Osmocom Project: OsmoMSC VTY Reference Manual. https://ftp.osmocom.org/docs/-latest/osmomsc-vty-reference.pdf

[23] [userman-osmonitb] Osmocom Project: OsmoNITB User Manual. https://ftp.osmocom.org/docs/latest/-osmonitb-usermanual.pdf

[24] [vty-ref-osmonitb] Osmocom Project: OsmoNITB VTY Reference Manual. https://ftp.osmocom.org/docs/-latest/osmonitb-vty-reference.pdf

[25] [userman-osmopcu] Osmocom Project: OsmoPCU User Manual. https://ftp.osmocom.org/docs/latest/-osmopcu-usermanual.pdf

[26] [vty-ref-osmopcu] Osmocom Project: OsmoPCU VTY Reference Manual. https://ftp.osmocom.org/docs/-latest/osmopcu-vty-reference.pdf

[27] [userman-osmosgsn] Osmocom Project: OsmoSGSN User Manual. https://ftp.osmocom.org/docs/latest/-osmosgsn-usermanual.pdf

[28] [vty-ref-osmosgsn] Osmocom Project: OsmoSGSN VTY Reference Manual. https://ftp.osmocom.org/docs/-latest/osmosgsn-vty-reference.pdf

[29] [userman-osmosipconnector] Osmocom Project: OsmoSIPconnector User Manual. https://ftp.osmocom.org/-docs/latest/osmosipconnector-usermanual.pdf

[30] [vty-ref-osmosipconnector] Osmocom Project: OsmoSIPconnector VTY Reference Manual. https://ftp.osmocom.org/docs/latest/osmosipconnector-vty-reference.pdf

[31] [userman-osmosmlc] Osmocom Project: OsmoSMLC User Manual. https://ftp.osmocom.org/docs/latest/-osmosmlc-usermanual.pdf

[32] [vty-ref-osmosmlc] Osmocom Project: OsmoSMLC VTY Reference Manual. https://ftp.osmocom.org/docs/-latest/osmosmlc-vty-reference.pdf

[33] [userman-osmostp] Osmocom Project: OsmoSTP User Manual. https://ftp.osmocom.org/docs/latest/osmostp-usermanual.pdf

[34] [vty-ref-osmostp] Osmocom Project: OsmoSTP VTY Reference Manual. https://ftp.osmocom.org/docs/latest/-osmostp-vty-reference.pdf

[35] [userman-osmotrx] Osmocom Project: OsmoTRX User Manual. https://ftp.osmocom.org/docs/latest/osmotrx-usermanual.pdf

[36] [vty-ref-osmotrx] Osmocom Project: OsmoTRX VTY Reference Manual. https://ftp.osmocom.org/docs/-latest/osmotrx-uhd-vty-reference.pdf https://ftp.osmocom.org/docs/latest/osmotrx-lms-vty-reference.pdf https://ftp.osmocom.org/docs/latest/osmotrx-ipc-vty-reference.pdf https://ftp.osmocom.org/docs/latest/-osmotrx-usrp1-vty-reference.pdf

[37] [3gpp-ts-23-041] 3GPP TS 23.041: Technical realization of Cell Broadcast Service (CBS)

[38] [3gpp-ts-23-048] 3GPP TS 23.048: Security mechanisms for the (U)SIM application toolkit; Stage 2 https://www.3gpp.org/DynaReport/23048.htm

[39] [3gpp-ts-23-236] 3GPP TS 23.236: Intra-domain connection of Radio Access Network (RAN) nodes to multiple Core Network (CN) nodes https://www.3gpp.org/DynaReport/23236.htm

[40] [3gpp-ts-24-007] 3GPP TS 24.007: Mobile radio interface signalling layer 3; General Aspects https://www.3gpp.org/DynaReport/24007.htm

[41] [3gpp-ts-24-008] 3GPP TS 24.008: Mobile radio interface Layer 3 specification; Core network protocols; Stage 3. https://www.3gpp.org/dynareport/24008.htm

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

[42] [3gpp-ts-31-101] 3GPP TS 31.101: UICC-terminal interface; Physical and logical characteristics https://www.3gpp.org/DynaReport/31101.htm

[43] [3gpp-ts-31-102] 3GPP TS 31.102: Characteristics of the Universal Subscriber Identity Module (USIM) application https://www.3gpp.org/DynaReport/31102.htm

[44] [3gpp-ts-31-103] 3GPP TS 31.103: Characteristics of the IMS Subscriber Identity Module (ISIM) application https://www.3gpp.org/DynaReport/31103.htm

[45] [3gpp-ts-31-111] 3GPP TS 31.111: Universal Subscriber Identity Module (USIM) Application Toolkit (USAT) https://www.3gpp.org/DynaReport/31111.htm

[46] [3gpp-ts-31-115] 3GPP TS 31.115: Secured packet structure for (Universal) Subscriber Identity Module (U)SIM Toolkit applications https://www.3gpp.org/DynaReport/31115.htm

[47] [3gpp-ts-31-116] 3GPP TS 31.116: Remote APDU Structure for (U)SIM Toolkit applications https://www.3gpp.org/DynaReport/31116.htm

[48] [3gpp-ts-35-205] 3GPP TS 35.205: 3G Security; Specification of the MILENAGE algorithm set: General

[49] [3gpp-ts-35-206] 3GPP TS 35.206: 3G Security; Specification of the MILENAGE algorithm set: Algorithm specification https://www.3gpp.org/DynaReport/35206.htm

[50] [3gpp-ts-44-006] 3GPP TS 44.006: Mobile Station - Base Station System (MS - BSS) interface; Data Link (DL) layer specification https://www.3gpp.org/DynaReport/44006.htm

[51] [3gpp-ts-44-018] 3GPP TS 44.018: Mobile radio interface layer 3 specification; Radio Resource Control (RRC) protocol https://www.3gpp.org/DynaReport/44018.htm

[52] [3gpp-ts-44-064] 3GPP TS 44.064: Mobile Station - Serving GPRS Support Node (MS-SGSN); Logical Link Control (LLC) Layer Specification https://www.3gpp.org/DynaReport/44064.htm

[53] [3gpp-ts-45-002] 3GPP TS 45.002: Digital cellular telecommunications system (Phase 2+) (GSM); GSM/EDGE Multiplexing and multiple access on the radio path https://www.3gpp.org/DynaReport/45002.htm

[54] [3gpp-ts-48-008] 3GPP TS 48.008: Mobile Switching Centre - Base Station system (MSC-BSS) interface; Layer 3 specification https://www.3gpp.org/DynaReport/48008.htm

[55] [3gpp-ts-48-016] 3GPP TS 48.016: General Packet Radio Service (GPRS); Base Station System (BSS) - Serving GPRS Support Node (SGSN) interface; Network service https://www.3gpp.org/DynaReport/48016.htm

[56] [3gpp-ts-48-018] 3GPP TS 48.018: General Packet Radio Service (GPRS); Base Station System (BSS) - Serving GPRS Support Node (SGSN); BSS GPRS protocol (BSSGP) https://www.3gpp.org/DynaReport/48018.htm

[57] [3gpp-ts-48-049] 3GPP TS 48.049: Digital cellular communications system; Base Station Controller - Cell Broadcast Centre (BSC-CBC) interface specification; Cell Broadcast Service Protocol (CBSP) https://www.3gpp.org/DynaReport/48049.htm

[58] [3gpp-ts-48-056] 3GPP TS 48.056: Base Station Controller - Base Transceiver Station (BSC - BTS) interface; Layer 2 specification https://www.3gpp.org/DynaReport/48056.htm

[59] [3gpp-ts-48-058] 3GPP TS 48.058: Base Station Controller - Base Transceiver Station (BSC - BTS) Interface; Layer 3 specification https://www.3gpp.org/DynaReport/48058.htm

[60] [3gpp-ts-51-011] 3GPP TS 51.011: Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) interface

[61] [3gpp-ts-51-014] 3GPP TS 51.014: Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface https://www.3gpp.org/DynaReport/51014.htm

[62] [3gpp-ts-52-021] 3GPP TS 52.021: Network Management (NM) procedures and messages on the A-bis interface https://www.3gpp.org/DynaReport/52021.htm

[63] [etsi-tr102216] ETSI TR 102 216: Smart cards https://www.etsi.org/deliver/etsi_tr/102200_102299/102216/-03.00.00_60/tr_102216v030000p.pdf

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

[64] [etsi-ts102221] ETSI TS 102 221: Smart Cards; UICC-Terminal interface; Physical and logical characteristics https://www.etsi.org/deliver/etsi_ts/102200_102299/102221/13.01.00_60/ts_102221v130100p.pdf

[65] [etsi-ts101220] ETSI TS 101 220: Smart Cards; ETSI numbering system for telecommunication application providers https://www.etsi.org/deliver/etsi_ts/101200_101299/101220/12.00.00_60/ts_101220v120000p.pdf

[66] [etsi-ts102671] ETSI TS 102 671: Smart Cards; Machine to Machine UICC; Physical and logical characteristics https://www.etsi.org/deliver/etsi_ts/102600_102699/102671/18.01.00_60/ts_102671v180100p.pdf

[67] [ieee-802.1q] IEEE 802.1Q: Bridges and Bridged Networks https://ieeexplore.ieee.org/document/6991462

[68] [ietf-rfc768] IETF RFC 768: User Datagram Protocol https://tools.ietf.org/html/rfc768

[69] [ietf-rfc791] IETF RFC 791: Internet Protocol https://tools.ietf.org/html/rfc791

[70] [ietf-rfc793] IETF RFC 793: Transmission Control Protocol https://tools.ietf.org/html/rfc793

[71] [ietf-rfc1035] IETF RFC 1035: Domain Names - Implementation and Specification https://tools.ietf.org/html/-rfc1035

[72] [ietf-rfc1350] IETF RFC 1350: Trivial File Transfer Protocol https://tools.ietf.org/html/rfc1350

[73] [ietf-rfc2131] IETF RFC 2131: Dynamic Host Configuration Protocol https://tools.ietf.org/html/rfc2131

[74] [ietf-rfc2474] IETF RFC 2474: Definition of the Differentiated Services Field (DS Field) in the IPv44 and IPv6 Headers https://tools.ietf.org/html/rfc2474

[75] [ietf-rfc2719] IETF RFC 2719: Signal Transport over IP https://tools.ietf.org/html/rfc2719

[76] [ietf-rfc3331] IETF RFC 3331: Message Transfer Part 2 User Adaptation Layer https://tools.ietf.org/html/-rfc3331

[77] [ietf-rfc3550] IETF RFC 3550: RTP: A Transport protocol for Real-Time Applications https://tools.ietf.org/-html/rfc3550

[78] [ietf-rfc3596] IETF RFC 3596: DNS Extensions to Support IP Version 6 https://tools.ietf.org/html/rfc3596

[79] [ietf-rfc3868] IETF RFC 3868: SCCP User Adaptation Layer https://tools.ietf.org/html/rfc3868

[80] [ietf-rfc4165] IETF RFC 4165: Message Transfer Part 2 Peer-to-Peer Adaptation Layer https://tools.ietf.org/-html/rfc4165

[81] [ietf-rfc4251] IETF RFC 4251: The Secure Shell (SSH) Protocol Architecture https://tools.ietf.org/html/-rfc4251

[82] [ietf-rfc4666] IETF RFC 4666: Message Transfer Part 3 User Adaptation Layer https://tools.ietf.org/html/-rfc4666

[83] [ietf-rfc5771] IETF RFC 5771: IANA Guidelines for IPv4 Multicast Address Assignments https://tools.ietf.org/html/rfc5771

[84] [itu-t-q701] ITU-T Q.701: Functional Description of the Message Transfer Part (MTP) https://www.itu.int/rec/-T-REC-Q.701/en/

[85] [itu-t-q711] ITU-T Q.711: Functional Description of the Signalling Connection Control Part https://www.itu.int/rec/T-REC-Q.711/en/

[86] [itu-t-q713] ITU-T Q.713: Signalling connection control part formats and codes https://www.itu.int/rec/T-REC-Q.713/en/

[87] [itu-t-q714] ITU-T Q.714: Signalling connection control part procedures https://www.itu.int/rec/T-REC-Q.714/en/

[88] [itu-t-q921] ITU-T Q.921: ISDN user-network interface - Data link layer specification https://www.itu.int/rec/-T-REC-Q.921/en

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

[89] [smpp-34] SMPP Develoepers Forum. Short Message Peer-to-Peer Protocol Specification v3.4 https://docs.nimta.com/SMPP_v3_4_Issue1_2.pdf

[90] [gnu-agplv3] Free Software Foundation. GNU Affero General Public License. https://www.gnu.org/licenses/-agpl-3.0.en.html

[91] [freeswitch_pbx] FreeSWITCH SIP PBX https://freeswitch.org

[92] [tw-ts-001] TW-TS-001: Enhanced RTP transport of FR and EFR codec frames in an IP-based GSM RAN https://www.freecalypso.org/specs/tw-ts-001-v010100.txt

[93] [tw-ts-002] TW-TS-002: Enhanced RTP transport of HRv1 codec frames in an IP-based GSM RAN https://www.freecalypso.org/specs/tw-ts-002-v010100.txt

[94] [tw-ts-003] TW-TS-003: BSSMAP extension for selection of enhanced RTP transport formats https://www.freecalypso.org/specs/tw-ts-003-v010002.txt

# C GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. http://fsf.org/

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

## C.1 PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

## C.2 APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Section whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then

it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not Transparent is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, Title Page means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## C.3 VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section Section C.4.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## C.4 COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## C.5 MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

a. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

b. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

c. State on the Title Page the name of the publisher of the Modified Version, as the publisher.

d. Preserve all the copyright notices of the Document.

e. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

f. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

g. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

h. Include an unaltered copy of this License.

i. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

j. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

k. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

l. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

m. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

n. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Sections.

o. Preserve any Warranty Disclaimers.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Section and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## C.6 COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

## C.7 COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## C.8 AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

## C.9 TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## C.10 TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

## C.11 FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

## C.12 RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

### C.13 ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (c)  YEAR  YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with…Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Copyright © 2018 sysmocom - s.f.m.c. GmbH 