Mercury Quick Manual

Mercury Quick Manual

Structure, API Functions and Configuration Options

Fadi Jerji, Ph.D.

Quick Manual for Mercury version 0.2 issue 1.

Author: Fadi Jerji

Email: fadi.jerji@ <gmail.com, caisresearch.com, ieee.org>

ORCID: 0000-0002-2076-5831

Contents

GLOSSARY

AGC	Automatic Gain Controller
API	Application Programming Interface
ARQ	Automatic Repeat Request
AWGN	Additive White Gaussian Noise
BB	Base Band
BER	Bit Error Rate
BP	Band Pass
CFO	Carrier Frequency Offset
CRC	Cyclic Redundancy Check
DSP	Digital Signal Processing
FEC	Forward Error Correction
FFT	Fast Fourier Transfer
FIFO	First In First Out
FIR	Finite Impulse Response
GBF	Gradient Bit-Flipping Algorithm
GI	Guard Interval
GRC	GNU Radio Companion
HPF	High Pass Filter
LDPC	Low-Density Parity-Check
LPF	Low Pass Filter
LS	Least Square
MER	Modulation Error Rate
OFDM	Orthogonal Frequency-Division Multiplexing

P2P	Peer to Peer
PAPR	Peak-to-Average Power Ratio
PB	Pass Band
PSK	Phase Shift Keying
PTT	Push To Talk
SNR	Signal to Noise Ratio
SPA	Sum-Product Algorithm
ZF	Zero Forcing

Chapter 1

Introduction

1.0.1 What is Mercury

Mercury is a configurable open-source software-defined modem. Mercury is written fully in C++ for performance.

While the current version of Mercury is developed and tested on Linux-based operating systems, it doesn't rely on any non-C++-standard library, thus it can be easily ported to other operating systems.

1.0.2 History

1.0.3 Older versions of Mercury

The development of Mercury started on January 2022 and the first pre-release was version 0.1 on December 2022 containing 9.5K lines of code. The Mercury 0.1 featured a physical layer with Low-Density Parity-Check (LDPC) Forward Error Correction (FEC), an Orthogonal Frequency-Division Multiplexing (OFDM), a Zero Forcing (ZF) channel estimator, Finite Impulse Response (FIR) filters, a Dx Yx pilot framing, a Schmidl&Cox-based Guard Interval (GI)-based time synchronization, a Carrier Frequency Offset (CFO) compensator, a Base Band (BB) ad Pass Band (PB) simulation, and a Dummy datalink layer with a TCP/IP interface. The Mercury 0.1 could handle Signal to Noise Ratio (SNR) of 5 dB.

Mercury 0.1.3 was released on July 2023 containing 18K lines of code. This pre-release introduced a datalink layer with an Automatic Repeat Request (ARQ) Peer to Peer (P2P) a new TCP/IP Application Programming Interface (API), a Gearshift auto-negotiation to auto-select the modulation and FEC, an uplink and downlink SNR measurement, a low overhead structured data block and data batches division to avoid retransmission.

Mercury 0.1.5 was released on January 2024 containing 25K lines of code. This pre-release introduced several enhancements to the physical layer including a Sum-Product Algorithm (SPA) LDPC decoder, a configurable preamble, a new Schmidl&Cox-based preamble-based time synchronization, a preamble-based CFO compensation, an enhanced ZF channel estimator, a Peak-to-Average Power Ratio (PAPR) reduction stage, a Band Pass (BP) transmission FIR filter, an enhanced reception logic with an increased reception buffer, a time and frequency interleavers and an Automatic Gain Controller (AGC) stage.

Mercury 0.1.5 of Mercury was successfully tested using Sbitx radios using Raspberry Pi 4 microcomputers. With a transmission power of 20 Watts, over a 70 km distance, a received SNR= -3 dB using TX_test, RX_test, and Config_0.

1.0.4 The current version of Mercury (0.2)

Mercury 0.2 was released on June 2024 containing 47.7K lines of code. This pre-release brought a new Least Square (LS) channel estimator with a configurable estimation window, an enhanced Time and Frequency synchronization for low SNR values, an enhanced TX and RX filtering with separate filters for time synchronization and data messages, an enhanced TX and RX mechanisms for higher robustness and computing efficiency, new Time and Frequency interleavers, new LDPC codes (1/16 to 14/16) optimized for multipath channel and an outer Cyclic Redundancy Check (CRC) code, an energy dispersal for power amplification efficiency, a pre-equalization to combat filters and Digital Signal Processing (DSP) imperfections, a PAPR and Modulation Error Rate (MER) measurements, two pilot distribution modes for different channels and bitrate requirements with further optimized parameters such as number of symbols, number of carriers, and synchronization symbols. Dynamic partial configuration of the physical layer for computing performance enhancement, an enhanced API of the physical layer to allow for byte or bit transfer, as separate Data and Acknowledge message robustness configuration, a new Ladder-based Gearshift mode for the datalink layer for low SNR, 17 new robustness modes for the ARQ/ Gearshift, and several bug fixes and performance enhancements.

1.1 Structure

The mercury software-defined modem is composed of two main layers, the physical layer, and the datalink layer.

1.1.1 The physical layer

The physical layer features an OFDM modulator/demodulator with an LDPC error correction code encoder/decoder with an embedded Additive white an Additive White Gaussian Noise (AWGN) channel simulator.

The physical layer is the part responsible for data protection via the use of LDPC codes and information mapping on the electromagnetic spectrum via modulation using OFDM and other digital signal processing methods.

The physical layer provides a connection-less transmission. In other words, the transmitter doesn't have any information regarding the receiver status, the channel status, or packets received/lost. This task is left to the upper layers.

The physical layer can be interfaced from the upper end and the lower end. The lower end interfaces with the hardware via the sound card using the Alsa driver. The speaker and microphone can be connected to an off-the-shelf radio transceiver to transmit the signal over an electromagnetic channel.

The upper layer can interface with the physical layer via three main connection points, send(), receive(), and load_configuration(). In addition, function get_configuration(SNR) allows the physical layer to inform the upper layer of the appropriate configuration for a specific SNR.

1.1.2 The datalink layer

1.1.2.1 ARQ

The datalink can be in one of two roles, a commander or a responder. The commander is the side responsible for controlling the message flux. The datalink layer provides a connection based using an ARQ via retransmission and acknowledgment mechanisms. This is to guarantee the reception of the information even in cases of message loss. In the case of an ACK message lost, a request to resend the last ACK message can be sent.

The datalink layer packs the messages in batches to avoid a fast TX/RX switch. Nevertheless, the batch size is configurable and can be set to one. Several batches form a block. Writing to the TX data buffer and reading from the RX_data_sync_buffer are done in blocks. The block size is configurable as well.

1.1.2.2 Gearshift

The datalink later exchanges channel state information to negotiate the highest data rate possible for the downlink channel, and in case of a change, adapts to the new channel condition. This "Gearshift" is done via a set_config message that a commander can send at any time and a recovery mechanism in case of a worsening channel condition. The channel evaluation and gearshift operations are done at the beginning of each data block.

The gearshift algorithm is configurable and can be an SNR based that relies on a test message to evaluate the channel condition (downlink) and set the configuration accordingly, or a Ladder algorithm that starts at an initial selectable robustness configuration and switches to the next configuration in the case of crossing a configurable threshold or return to the more robust configuration in case of not achieving a configurable lower transmission success threshold.

The gearshift mechanism can enter a temporary configurable block period to avoid multiple unsuccessful up-gearshifts.

The robustness configurations are numbered (0 to 16) where CONFIG_0 is the most robust one.

The datalink layer connects to a possible application layer via two TCP/IP connections, a data dump buffer, and a control connection with an API.

1.1.2.3 API

A base API was implemented to provide the basic functionality in a way that would be familiar to users of commercially available modems. The API can easily be updated and new commands can be added to provide further control to the application layer.

The base API is composed of the following commands:

MYCALL mycall\r
Sets the AQR call sign to the value of the string ”my_call”.
Reply: OK\r

LISTEN ON\r
Sets the ARQ to a responder role and waits for an incoming connection.
Reply: OK\r

CONNECT my_call destination_call\r
Sets the ARQ to a commander role and initiates the connection sequence to ”destination_call”.
Reply: OK\r

DISCONNECT\r
Terminates the current connection.
Reply: OK\r

BW2300\r
Sets the bandwidth to 2300 Hz.
Reply: OK\r

BW2500\r

Sets the bandwidth to 2500 Hz.

Reply: OK\r

BUFFER TX\r

Reads the number of data bytes in the TX buffer yet to be sent.

Reply: BUFFER bytes\r

In the case of an unrecognized command, a reply: NOK\r is provided.

Chapter 2

First steps

2.0.1 Get Mercury on Linux

2.0.1.1 Prerequisites

First, you need to install Mercury dependencies:

ALSA Sound: To use the sound card via ALSA Sound library, the library should be installed and linked to the code at compilation. To install ALSA Sound run the following command:

```
sudo apt-get install libasound2-dev
```

GNU PLOT: In case of switching the configuration of the physical layer plot_active to YES, the Gnuplot is required to be installed on the machine. To install GNU plot run the following command:

```
sudo apt-get install gnuplot-x11
```

Documentation: To generate the documentation, the Doxygen and GraphViz packets should be installed. To install Doxygen and GraphViz run the following commands:

```
sudo apt-get install doxygen
```

```
sudo apt-get install graphviz
```

2.0.1.2 Mercury source code

To get the latest version of Mercury, you need to clone the source code from the GitHub repository:

```
git clone https://github.com/Rhizomatica/mercury.git
```

then navigate to the Mercury directory

```
cd mercury
```

finally, compile the source code using “make”

```
make mercury
```

or generate the documentation

```
make doc
```

optionally you can install the binary files

```
make install
```

2.0.2 Modes

Mercury can operate in one of five different modes, the mode can be selected in “source/main.cc”

```
telecom_system.operation_mode=‘‘selected mode”;
```

ARQ_MODE: this mode engages the datalink layer with ARQ and the physical layer, this is the operational mode.

BER_PLOT_baseband: this mode runs a physical layer-only baseband Bit Error Rate (BER) simulation over an AWGN channel with/without plotting. In this mode, filters, PAPR reduction and up/down frequency converters are not included in the simulation

BER_PLOT_passband: this mode runs a physical layer-only passband BER simulation over an AWGN channel with/without plotting. In this mode, the full TX and RX functions of the physical layers are included in the simulation.

TX_TEST: this mode generates an output signal with random data content.

RX_TEST: this mode demodulates and decodes a received signal with/without plotting.

2.0.3 Testing

In addition to running the BB or the PB simulation for a specific configuration, the physical layer of Mercury can be tested on its own or the full Mercury stack can be tested as a whole.

2.0.3.1 Physical layer

To test Mercury's physical layer, two computers are required with a crossover connection of the speaker and microphone (the speaker of the first computer is connected to the microphone of the second one and vice versa).

In this case (or in the case of connecting Mercury to a physical radio), physical layer parameters such as the signal frequency, power, and filter values need to be adjusted adequately. This can be done from the physical layer configuration file.

Alternatively, an ALSA loopback with GNU Radio Companion (GRC) can be used.

To configure the ALSA loopback run the following command:

```
sudo modprobe snd-aloop
```

To install the GRC software run the following command:

```
sudo apt-get install gnuradio
```

Clone and compile two separate copies of Mercury.

considering the loop-back ALSA card has the number '3', configure the first Mercury (Mercury 1) speaker to

```
speaker_dev_name="plughw:3,0,3";
```

and the Mercury 1 microphone to

```
speaker_dev_name=”plughw:3,0,2”;
```

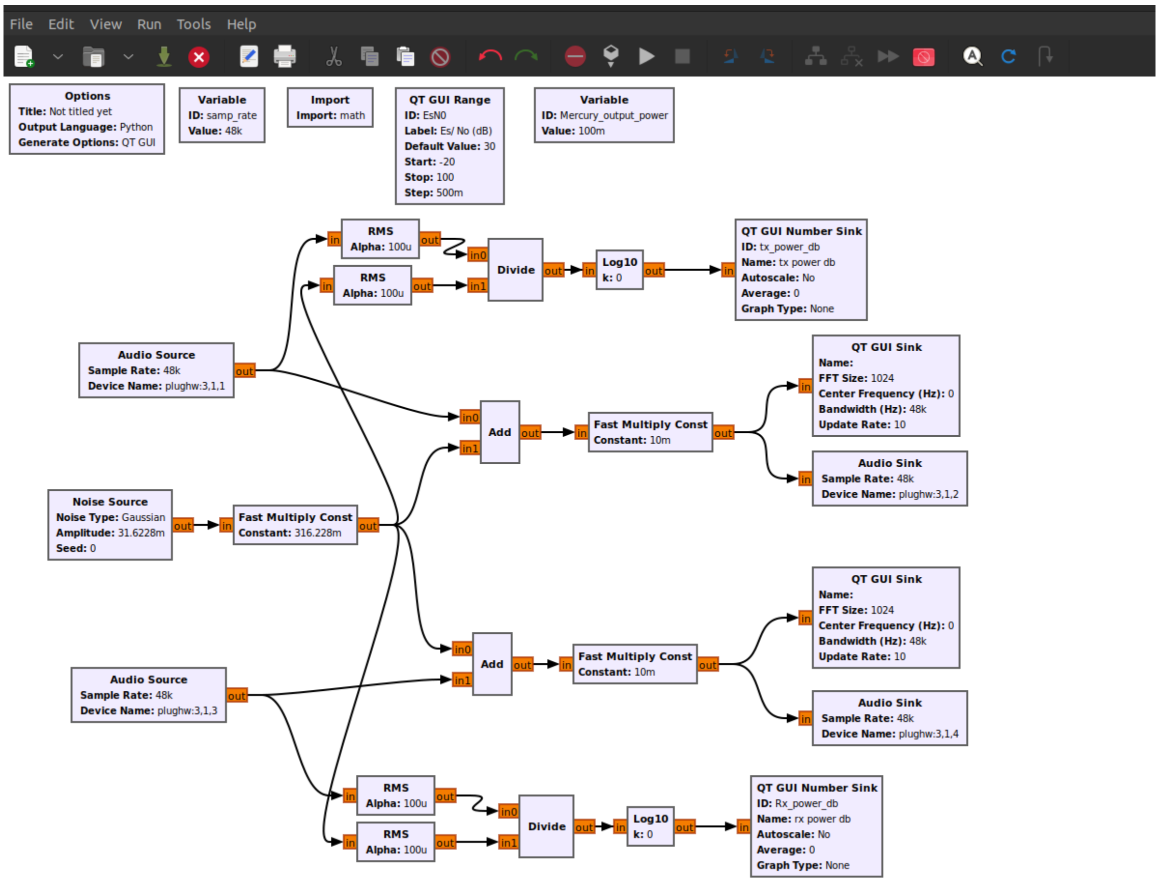
and configure the Mercury 2 speaker to

```
speaker_dev_name=”plughw:3,0,1”;
```

and the Mercury 2 microphone to

```
speaker_dev_name=”plughw:3,0,4”;
```

By starting GRC with the “Mercury_channel_test.grc” project, the connection between the two Mercuries is established.



Options
Title: Not titled yet
Output Language: Python
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 48k

Import
Import: math

QT GUI Range
ID: EsN0
Label: Es/ No (dB)
Default Value: 30
Start: -20
Stop: 100
Step: 500m

Variable
ID: Mercury_output_power
Value: 100m

RMS
Alpha: 100u

RMS
Alpha: 100u

Divide

Log10
k: 0

QT GUI Number Sink
ID: tx_power_db
Name: tx power db
Autoscale: No
Average: 0
Graph Type: None

Audio Source
Sample Rate: 48k
Device Name: plughw:3,1,1

Add

Fast Multiply Const
Constant: 10m

QT GUI Sink
Name:
FFT Size: 1024
Center Frequency (Hz): 0
Bandwidth (Hz): 48k
Update Rate: 10

Audio Sink
Sample Rate: 48k
Device Name: plughw:3,1,2

Noise Source
Noise Type: Gaussian
Amplitude: 31.6228m
Seed: 0

Fast Multiply Const
Constant: 316.228m

QT GUI Sink
Name:
FFT Size: 1024
Center Frequency (Hz): 0
Bandwidth (Hz): 48k
Update Rate: 10

Add

Fast Multiply Const
Constant: 10m

Audio Source
Sample Rate: 48k
Device Name: plughw:3,1,3

Audio Sink
Sample Rate: 48k
Device Name: plughw:3,1,4

RMS
Alpha: 100u

RMS
Alpha: 100u

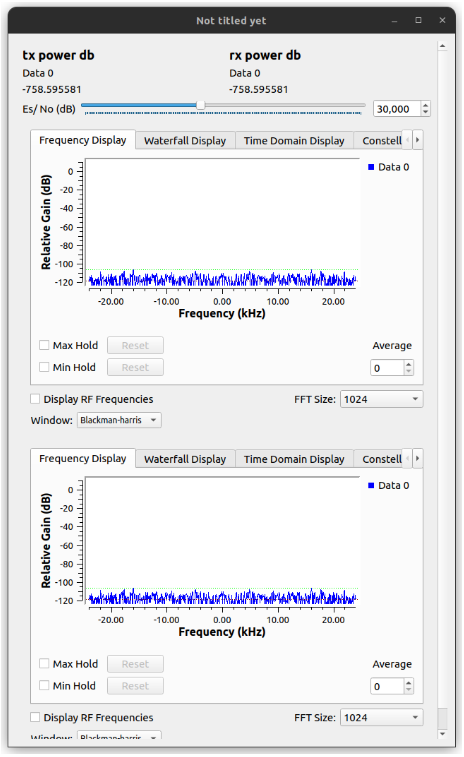
Divide

Log10
k: 0

QT GUI Number Sink
ID: Rx_power_db
Name: rx power db
Autoscale: No
Average: 0
Graph Type: None

Figure 2.1: Mercury channel test.

By starting the GNU Radio project, both the spectrum analyzer and the waterfall can be accessed.


Not titled yet

tx power db
Data 0
-758.595581

rx power db
Data 0
-758.595581

Es/ No (dB) 30,000

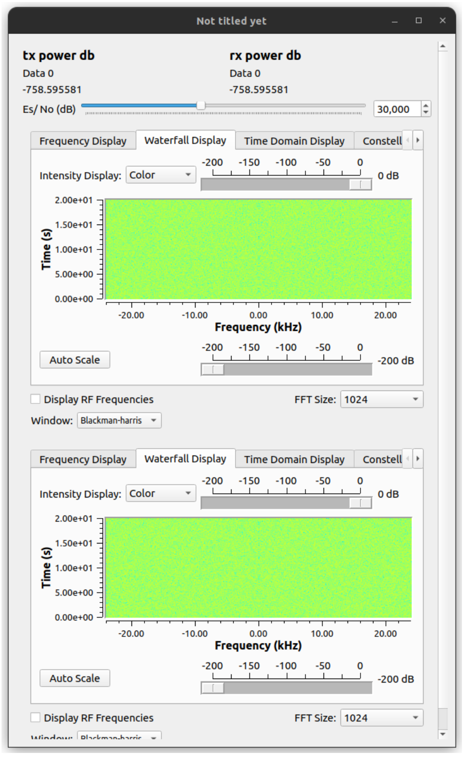
Max Hold
Min Hold
Average 0

Display RF Frequencies
FFT Size: 1024
Window: Blackman-harris

Max Hold
Min Hold
Average 0

Display RF Frequencies
FFT Size: 1024

Figure 2.2: Spectrum analyzer.


Not titled yet

tx power db
Data 0
-758.595581

rx power db
Data 0
-758.595581

Es/ No (dB) 30,000

Intensity Display: Color
0 dB
-200 dB

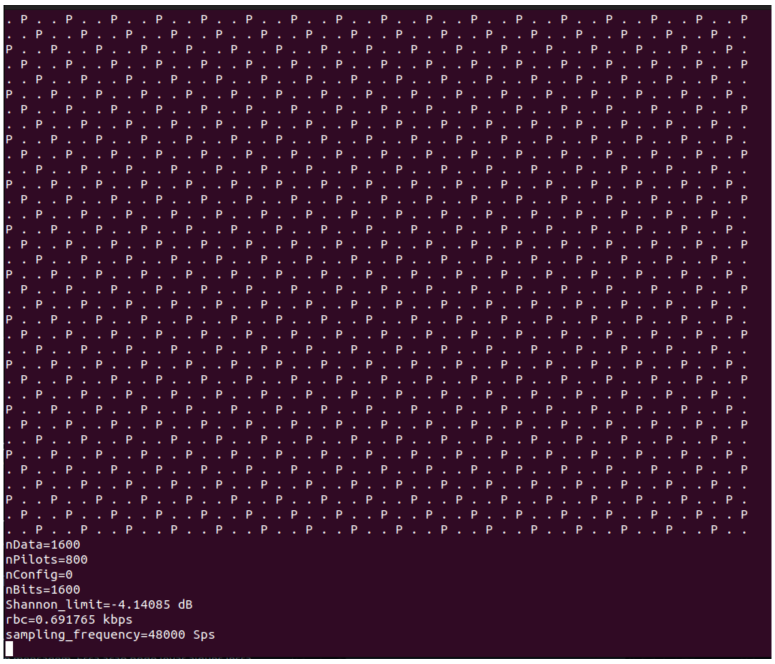
Display RF Frequencies
FFT Size: 1024
Window: Blackman-harris

Intensity Display: Color
0 dB
-200 dB

Display RF Frequencies
FFT Size: 1024

Figure 2.3: Waterfall.

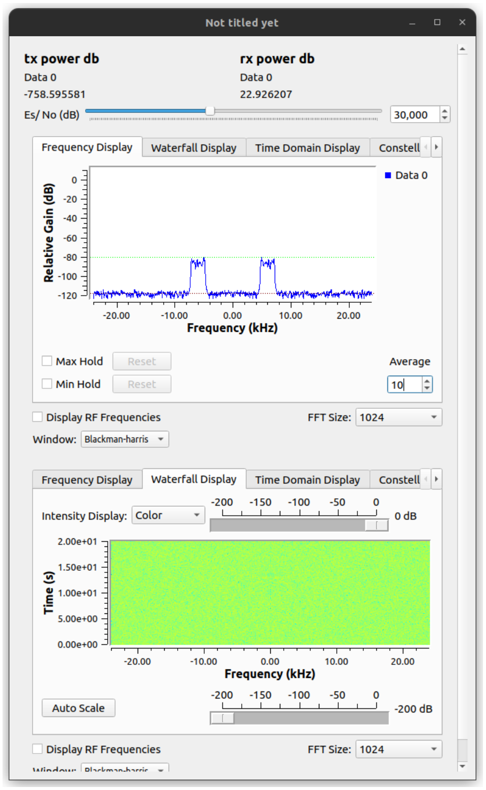
By configuring Mercury 1 to TX_TEST mode and starting the program, the pilot pattern and current configurations are printed in the terminal.



```
. P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P
. . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . .
P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P .
. P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P
. . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . .
P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P .
. P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P
. . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . .
P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P .
. P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P
. . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . .
P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P .
. P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P
. . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . .
P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P .
. P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P
. . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . .
P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P .
. P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P
. . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . .
P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P .
. P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P
. . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . .
P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P .
. P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P
. . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . .
P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P .
. P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P
. . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . .
P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P .
. P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P
. . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . .
P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P .
. P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P
. . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . . P . .
nData=1600
nPilots=800
nConfig=0
nBits=1600
Shannon_limit=-4.14085 dB
rbc=0.691765 kbps
sampling_frequency=48000 Sps
```

Figure 2.4: Mercury pilot pattern and current configurations.

The transmitted signal appears on both the spectrum analyzer and the waterfall.


Not titled yet

tx power db
Data 0
-758.595581

rx power db
Data 0
22.926207

Es/ No (dB) 30,000

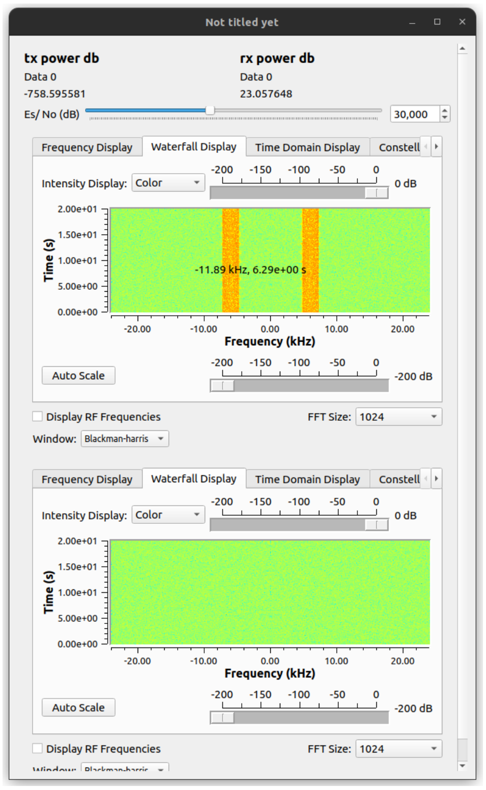
Max Hold
Min Hold
Average 10

Display RF Frequencies
FFT Size: 1024
Window: Blackman-harris

Intensity Display: Color
0 dB
-200 dB

Display RF Frequencies
FFT Size: 1024

Figure 2.5: Spectrum analyzer.


Not titled yet

tx power db
Data 0
-758.595581

rx power db
Data 0
23.057648

Es/ No (dB) 30,000

Intensity Display: Color
0 dB
-11.89 kHz, 6.29e+00 s
-200 dB

Display RF Frequencies
FFT Size: 1024
Window: Blackman-harris

Intensity Display: Color
0 dB
-200 dB

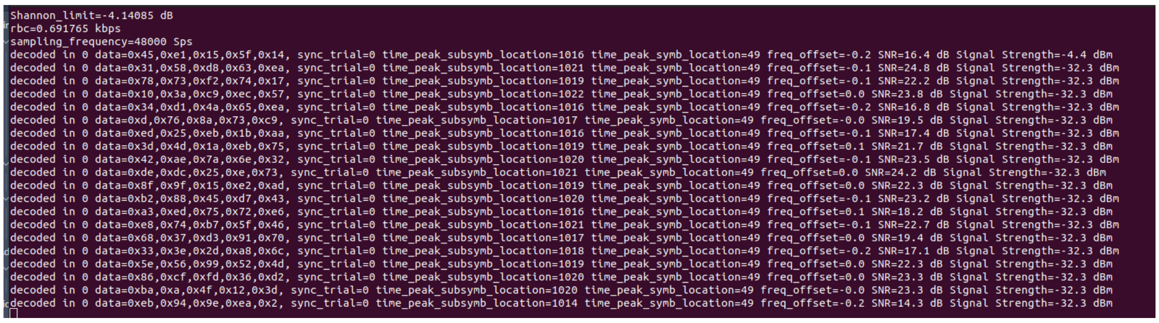
Display RF Frequencies
FFT Size: 1024

Figure 2.6: Waterfall.

By configuring Mercury r to RX_TEST mode and starting the program, the pilot pattern and current configurations are printed in the terminal.

It is important to verify that the plotting folder exists and Mercury has access to it, or that the plotting is inactive. This can be done from the physical layer configuration file.

Once the signal is received and decoded, the decoded message's initial bytes and synchronization information, and signal strength (dbm) are printed and the received constellation is plotted.



```
Shannon_limit=-4.14085 dB
rbc=0.691765 kbps
sampling_frequency=48000 Sps
decoded in 0 data=0x45,0xe1,0x15,0x5f,0x14, sync_trial=0 time_peak_subsymb_location=1016 time_peak_symb_location=49 freq_offset=-0.2 SNR=16.4 dB Signal Strength=-4.4 dBm
decoded in 0 data=0x31,0x58,0xd8,0x63,0xea, sync_trial=0 time_peak_subsymb_location=1021 time_peak_symb_location=49 freq_offset=-0.1 SNR=24.8 dB Signal Strength=-32.3 dBm
decoded in 0 data=0x78,0x73,0xf2,0x74,0x17, sync_trial=0 time_peak_subsymb_location=1019 time_peak_symb_location=49 freq_offset=-0.1 SNR=22.2 dB Signal Strength=-32.3 dBm
decoded in 0 data=0x10,0x3a,0xc9,0xec,0x57, sync_trial=0 time_peak_subsymb_location=1022 time_peak_symb_location=49 freq_offset=0.0 SNR=23.8 dB Signal Strength=-32.3 dBm
decoded in 0 data=0x34,0xd1,0x4a,0x65,0xea, sync_trial=0 time_peak_subsymb_location=1016 time_peak_symb_location=49 freq_offset=-0.2 SNR=16.8 dB Signal Strength=-32.3 dBm
decoded in 0 data=0xd,0x76,0x8a,0x73,0xc9, sync_trial=0 time_peak_subsymb_location=1017 time_peak_symb_location=49 freq_offset=-0.0 SNR=19.5 dB Signal Strength=-32.3 dBm
decoded in 0 data=0xed,0x25,0xeb,0x1b,0xaa, sync_trial=0 time_peak_subsymb_location=1016 time_peak_symb_location=49 freq_offset=-0.1 SNR=17.4 dB Signal Strength=-32.3 dBm
decoded in 0 data=0x3d,0x4d,0x1a,0xeb,0x75, sync_trial=0 time_peak_subsymb_location=1019 time_peak_symb_location=49 freq_offset=0.1 SNR=21.7 dB Signal Strength=-32.3 dBm
decoded in 0 data=0x42,0xae,0x7a,0x6e,0x32, sync_trial=0 time_peak_subsymb_location=1020 time_peak_symb_location=49 freq_offset=-0.1 SNR=23.5 dB Signal Strength=-32.3 dBm
decoded in 0 data=0xde,0xdc,0x25,0xe,0x73, sync_trial=0 time_peak_subsymb_location=1021 time_peak_symb_location=49 freq_offset=0.0 SNR=24.2 dB Signal Strength=-32.3 dBm
decoded in 0 data=0x8f,0x9f,0x15,0xe2,0xad, sync_trial=0 time_peak_subsymb_location=1019 time_peak_symb_location=49 freq_offset=0.0 SNR=22.3 dB Signal Strength=-32.3 dBm
decoded in 0 data=0xb2,0x88,0x45,0xd7,0x43, sync_trial=0 time_peak_subsymb_location=1020 time_peak_symb_location=49 freq_offset=-0.1 SNR=23.2 dB Signal Strength=-32.3 dBm
decoded in 0 data=0xa3,0xed,0x75,0x72,0xe6, sync_trial=0 time_peak_subsymb_location=1016 time_peak_symb_location=49 freq_offset=0.1 SNR=18.2 dB Signal Strength=-32.3 dBm
decoded in 0 data=0xe8,0x74,0xb7,0x5f,0x46, sync_trial=0 time_peak_subsymb_location=1021 time_peak_symb_location=49 freq_offset=-0.1 SNR=22.7 dB Signal Strength=-32.3 dBm
decoded in 0 data=0x68,0x37,0xd3,0x91,0x70, sync_trial=0 time_peak_subsymb_location=1017 time_peak_symb_location=49 freq_offset=0.0 SNR=19.4 dB Signal Strength=-32.3 dBm
decoded in 0 data=0x33,0x3e,0x2d,0xa8,0x6c, sync_trial=0 time_peak_subsymb_location=1018 time_peak_symb_location=49 freq_offset=-0.2 SNR=17.1 dB Signal Strength=-32.3 dBm
decoded in 0 data=0x5e,0x56,0x99,0x52,0x4d, sync_trial=0 time_peak_subsymb_location=1019 time_peak_symb_location=49 freq_offset=0.0 SNR=22.3 dB Signal Strength=-32.3 dBm
decoded in 0 data=0x86,0xcf,0xfd,0x36,0xd2, sync_trial=0 time_peak_subsymb_location=1020 time_peak_symb_location=49 freq_offset=0.0 SNR=23.3 dB Signal Strength=-32.3 dBm
decoded in 0 data=0xba,0xa,0x4f,0x12,0x3d, sync_trial=0 time_peak_subsymb_location=1020 time_peak_symb_location=49 freq_offset=-0.0 SNR=23.3 dB Signal Strength=-32.3 dBm
decoded in 0 data=0xeb,0x94,0x9e,0xea,0x2, sync_trial=0 time_peak_subsymb_location=1014 time_peak_symb_location=49 freq_offset=-0.2 SNR=14.3 dB Signal Strength=-32.3 dBm
```

Figure 2.7: Decoded message information.

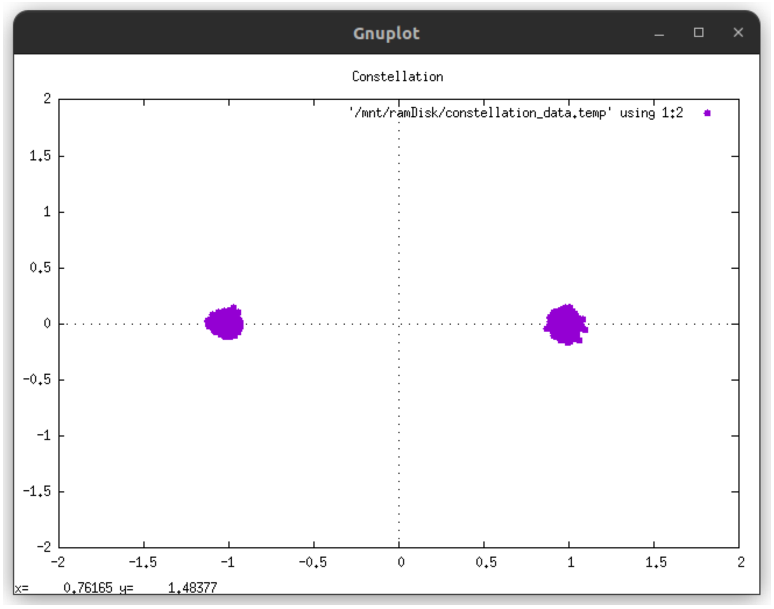


Figure 2.8: Received constellation.

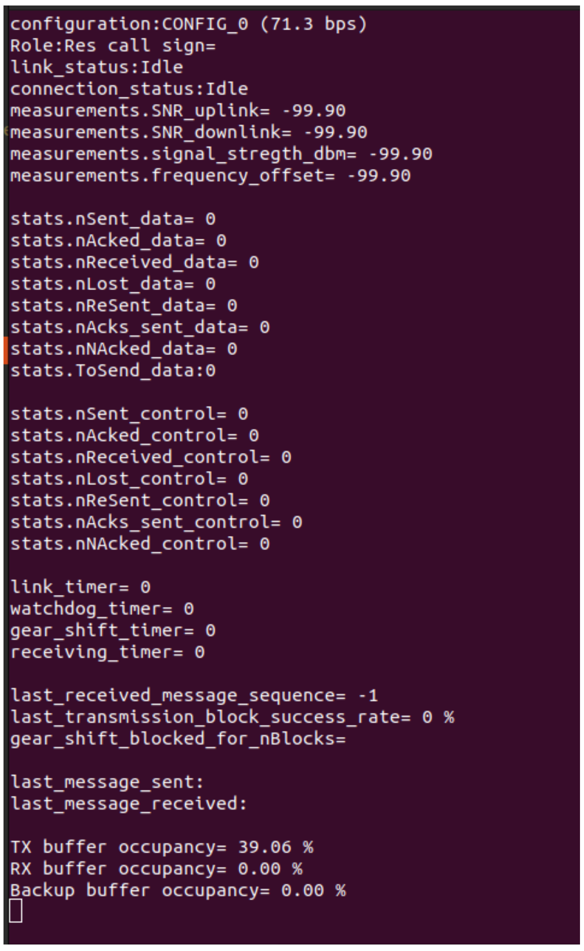
The same steps can be repeated to test the connection of Mercury 2 to Mercury 1.

in addition, the robustness configuration for both Mercury 1 and Mercury 2 can be changed from the physical layer configuration file to test any of the 17 pre-set configurations. This can be done by setting the variable “init_configuration” to the desired configuration.

The robustness of each of those configurations can be tested by changing the value of E_s/N_0 in the GNU radio window.

2.0.3.2 Datalink layer

Once the physical layer connection is validated in both directions, the mode of both Mercury 1 and Mercury 2 can be switched back to ARQ_MODE and the data link layer API can be used to set a CALL for each of them, initiate the connection and transfer data from one to another.



```
configuration:CONFIG_0 (71.3 bps)
Role:Res call sign=
link_status:Idle
connection_status:Idle
measurements.SNR_uplink= -99.90
measurements.SNR_downlink= -99.90
measurements.signal_stregth_dbm= -99.90
measurements.frequency_offset= -99.90

stats.nSent_data= 0
stats.nAcked_data= 0
stats.nReceived_data= 0
stats.nLost_data= 0
stats.nReSent_data= 0
stats.nAcks_sent_data= 0
stats.nNAcked_data= 0
stats.ToSend_data:0

stats.nSent_control= 0
stats.nAcked_control= 0
stats.nReceived_control= 0
stats.nLost_control= 0
stats.nReSent_control= 0
stats.nAcks_sent_control= 0
stats.nNAcked_control= 0

link_timer= 0
watchdog_timer= 0
gear_shift_timer= 0
receiving_timer= 0

last_received_message_sequence= -1
last_transmission_block_success_rate= 0 %
gear_shift_blocked_for_nBlocks=

last_message_sent:
last_message_received:

TX buffer occupancy= 39.06 %
RX buffer occupancy= 0.00 %
Backup buffer occupancy= 0.00 %
```

Figure 2.9: Datalink.

The datalink layer can be tested with Gearshift active or inactive, the initial configuration and the ACK configuration along with other configurations such as the API ports and timeout values can be configured fro the datalink layer configuration file.

2.0.4 Configurations

The physical layer and the datalink layer of Mercury can be configured independently.

2.0.4.1 Physical layer

The physical layer parameters can be defined in “physical_layer/physical_config.cc”

The physical layer has the following parameters to be configured (more fine-tuning parameters are in telecom_system.cc/telecom_system.h):

Roubustness:

init_configuration: The initial configuration to be used for transmission (Modulation, code rate, etc.)

value can be set to (CONFIG_0, CONFIG_1 ... CONFIG_16).

Table 2.1: Roubustness configurations.

Config	Modulation	FEC code rate	EsN0 (FER<0.1)	Bit-rate (HD pilots)	Bit-rate (LD pilots)
0	BPSK	1/16	-10.0	71.3	84.2
1	BPSK	2/16	-7.5	156.1	184.5
2	BPSK	3/16	-6.0	241.0	284.8
3	BPSK	4/16	-4.5	325.8	385.0
4	BPSK	5/16	-3.5	410.6	485.3
5	BPSK	6/16	-2.5	495.5	585.6
6	BPSK	8/16	-1.5	665.2	786.1
7	QPSK	5/16	-0.5	762.6	889.7
8	QPSK	6/16	0.5	920.2	1073.5
9	QPSK	8/16	1.5	1235.3	1441.2
10	8PSK	6/16	3.0	1353.7	1353.7
11	8PSK	8/16	4.0	1818.1	1818.1
12	QPSK	14/16	6.5	2261.4	2654.7
13	16QAM	8/16	7.5	2470.6	2882.4
14	8PSK	14/16	9.0	3389.7	3389.7
15	16QAM	14/16	12.5	4361.3	5088.2
16	32QAM	14/16	13.5	5664.7	5664.7

To create a new configuration, the name of the configuration needs to be added to comon_defines.h and the configuration parameters need to be added to the load_configuration function in the telecom_system.cc file of the physical layer.

Four essential parameters are needed for a new configuration, the modulation, ldpc_rate, ofdm_preamble_configurator_Nsymb and ofdm_channel_estimator.

OFDM:

The OFDM has the following main parameters (more fine-tuning parameters are in ofdm.cc/ofdm.h):

bandwidth: the used bandwidth for TX and RX.

frequency_interpolation_rate: sampling frequency interpolation rate to match the sound card sampling rate.

carrier_frequency: the OFDM signal carrier frequency in TX and RX.

output_power_Watt: the output signal power at TX.

Nfft: the Fast Fourier Transfer (FFT) size.

Nc: the number of used sub-channels.

Dx: frequency distance between pilots.

Dy: time distance between pilots.

Nsymb: number of symbols per OFDM frame.

gi: Guard interval to remove the multipath effect.

ofdm_start_shift: Number of non-used subcarriers starting at 0 Hz.

bit_energy_dispersal_seed: The seed used in the pseudorandom data generator of the bit energy dispersal.

Pilots:

pilot_boost: to define pilot carriers' power in relation to data carriers' power.

ofdm_pilot_configurator_first_row: Set the first row to data or pilot.

ofdm_pilot_configurator_last_row: Set the last row to data or pilot.

ofdm_pilot_configurator_first_col: Set the first column to data or pilot.

ofdm_pilot_configurator_second_col: Set the second column to data or pilot.

ofdm_pilot_configurator_last_col: Set the last column to data or pilot.

ofdm_pilot_configurator_seed: The seed used in the pseudorandom data generator of the pilot sequence.

ofdm_pilot_density: Pilot density (HIGH_DENSITY or LOW_DENSITY).

Channel Estmation:

ofdm_channel_estimator: The channel estimation Method (ZF or LS).

ofdm_channel_estimator_amplitude_restoration: Activate amplitude restoration (Yes, No). (only with M-Phase Shift Keying (PSK) modulation)

ofdm_LS_window_width: The LS window width.

ofdm_LS_window_hight: The LS window height.

Syncronization:

time_sync_trials_max: number of time synchronization detection peaks to be considered.

use_last_good_time_sync: whether to use the last-time synchronization result or try to synchronize again in each frame.

use_last_good_freq_offset: whether to use the last time synchronization result or try to synchronize again in each frame.

ofdm_time_sync_Nsymb: Number of OFDM symbols when the GI is used for time synchronization.

freq_offset_ignore_limit: the minimum value of sampling frequency and carrier frequency offset to be compensated.

Filters:

ofdm_FIR_rx_time_sync_filter_window: the FIR window at RX for the time sync part of the signal.

ofdm_FIR_rx_time_sync_filter_transition_bandwidth: the FIR transition width at RX for the time sync part of the signal.

ofdm_FIR_rx_time_sync_filter_cut_frequency: the FIR cut frequency at RX for the time sync part of the signal.

ofdm_FIR_rx_time_sync_filter_type: the FIR type (Low Pass Filter (LPF), High Pass Filter (HPF), etc..) for the time sync part of the signal.

ofdm_FIR_rx_data_filter_window: the FIR window at RX for the data part of the signal.

ofdm_FIR_rx_data_filter_transition_bandwidth: the FIR transition width at RX for the data part of the signal.

ofdm_FIR_rx_data_filter_cut_frequency: the FIR cut frequency at RX for the data part of the signal.

ofdm_FIR_rx_data_filter_type: the FIR type (LPF, HPF, etc..) for the data part of the signal.

ofdm_FIR_tx1_filter_window: the FIR window at TX.

ofdm_FIR_tx1_filter_transition_bandwidth: the FIR transition width at TX.

ofdm_FIR_tx1_filter_cut_frequency: the FIR cut frequency at TX.

ofdm_FIR_tx1_filter_type: the FIR type (LPF, HPF, etc..) at TX.

ofdm_FIR_tx2_filter_window: the FIR window at TX.

ofdm_FIR_tx2_filter_transition_bandwidth: the FIR transition width at TX.

ofdm_FIR_tx2_filter_cut_frequency: the FIR cut frequency at TX.

ofdm_FIR_tx2_filter_type: the FIR type (LPF, HPF, etc..).

Preamble:

ofdm_preamble_configurator_Nsymb: number of preamble symbols.

ofdm_preamble_configurator_nIdentical_sections: number of identical parts per preamble symbol.

ofdm_preamble_configurator_modulation: preamble modulation;

ofdm_preamble_configurator_boost: preamble boost.

ofdm_preamble_configurator_seed: preamble pseudo-random bits seed.

PAPR reduction:

ofdm_preamble_papr_cut: preamble PAPR cut limit (before filtering)

ofdm_data_papr_cut: data PAPR cut limit (before filtering)

FEC:

decoding algorithm: the LDPC decoding algorithm can be either the SPA or the Gradient Bit-Flipping Algorithm (GBF).

standard and framesize: the LDPC code deploys specially designed LDPC matrices of the size 1600 bits with several different code rates.

code rate: the code rate that defines the protection level vs data rate (1/16, 2/16, 3/16, 4/16, 5/16, 6/16 8/16, 14/16).

GBF_eta: the GBF LDPC decoder correction rate.

nIteration_max: the number of maximum decoding iterations.

The outer code has the following options:

outer_code: (CRC16_MODBUS_RTU or NONE)

ALSA:

microphone_dev_name: Alsa capture device name (”plughw:1,0” for example).

speaker_dev_name: Alsa play device name (”plughw:1,0” for example).

microphone_type: Capture channel.

microphone_channels: Capture channels (Mono, Stereo, Left, Right).

speaker_type: Play channel.

speaker_channels: Play channels (Mono, Stereo, Left, Right).

speaker_frames_to_leave_transmit_fct: Number of frames to be played before leaving the Alsa transmission function, to avoid Alsa underrun.

Plot:

plot_folder: Folder to be used as a temporary for the plot function.

plot_plot_active: Plot function activation (Yes, No).

2.0.4.2 Datalink layer

The datalink layer parameters can be defined in data_link_layer/datalink_config.cc

The datalink layer has the following parameters to be configured:

fifo_buffer_tx_size: The First In First Out (FIFO) transmission buffer size (data to be transmitted).

fifo_buffer_rx_size: The FIFO reception buffer size.

fifo_buffer_backup_size: Backup FIFO buffer for configuration switch.

link_timeout: Maximum time without any activity on the link (TX or RX) before link drops (ms).

tcp_socket_control_port: Control TCP/IP port.

tcp_socket_control_timeout_ms: Control TCP/IP timeout (ms).

tcp_socket_data_port: Data TCP/IP port.

tcp_socket_data_timeout_ms: Data TCP/IP timeout (ms).

init_configuration: The initial data configuration to be used.

ack_configuration: The configuration to be used for acknowledgment messages.

gear_shift_on: Gearshit setting (Yes, No).

gear_shift_algorithm: Gearshit algorithm (SNR-based, Ladder based)

gear_shift_up_success_rate_limit_precentage: Threshold for up-gearshifting.

gear_shift_down_success_rate_limit_precentage: Threshold for down-gearshifting.

gear_shift_block_for_nBlocks_total: Number of transmission blocks to be done without a gearshift try in case of an up-gearshift failure.

batch_size: Number of messages per data batch.

nMessages: Number of messages per block.

nResends: Number of trials for each data/control message.

ack_batch_size: Number of messages per acknowledgment batch.

control_batch_size: Number of messages per control batch.

ptt_on_delay_ms: time in milliseconds to wait after switching on the Push To Talk (PTT) before sending data.

ptt_off_delay_ms: time in milliseconds to wait after switching off the PTT after sending data.

switch_role_timeout_ms: time in milliseconds to wait before switching roles with the responder in case of an empty data buffer.

Chapter 3

License