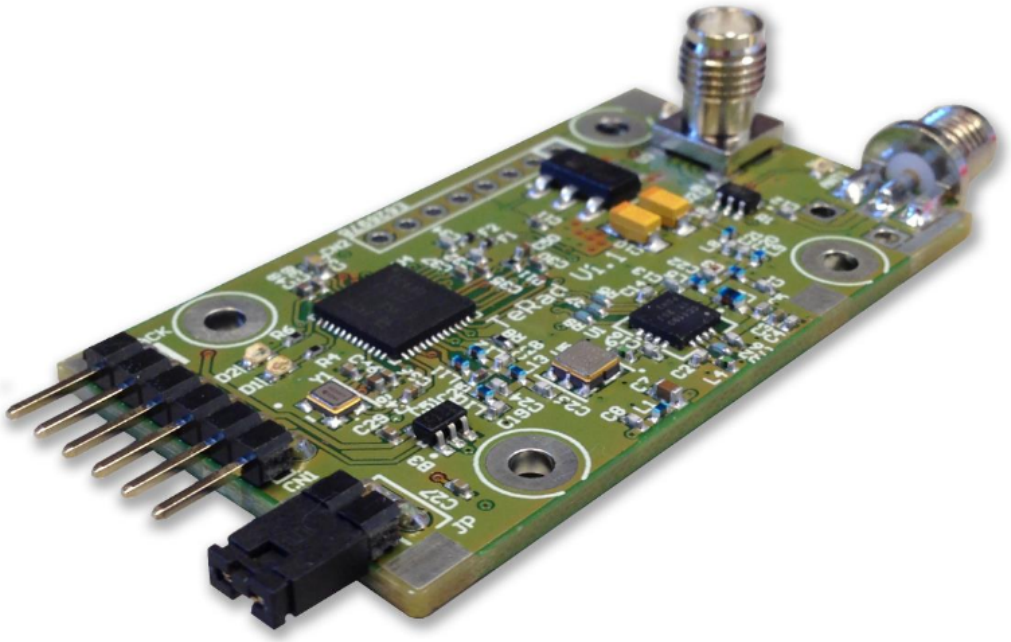
AMORES Project Consortium

http://www.amores-robotics.hu

AMORES Robotics Telemetry Radio

Handbook



Author:

Tom Szilagyi

Principal software developer

Published: 1st June, 2015.

This publication is up-to-date with TeRad board version 1.1 and software version 1.00.

Please check our website for product upgrades.

Contents

1 Product description

The **AMORES Robotics Telemetry Radio** (TeRad) is a state-of-the-art, high speed telemetry radio intended for professional UAV applications. The module meets the relevant European regulations.

1.1 Features

- Test-proven, reliable operation onboard UAV aircrafts
- 869 MHz ISM band
- Long range, low bit error rate
- Controllable output power for decreased current draw in short range applications
- Point-to-point operation
- High RF power for long range applications
- Antenna diversity for optimal signal quality
- Buffer alignment algorithm for MAVLink and AirGuardian protocols
- Hardware flow control for reliable UART data transmission
- Conforming to EU R&TTE directive EN 300 220

1.2 Specifications

Supply voltage	5 V_{DC} ± 5%
Maximal current consumption	365 mA
Maximal output power	26 dBm (400 mW)
Modulation	MSK, GFSK
Payload data transmission rate	17-185 kbit/sec
Nominal frequency	869 MHz
Antenna connectors	Standard 50Ω SMA connector
Suggested antenna	Amores Robotics $\lambda/2$ dipole
Operating temperature	from -40°C to +85°C
Dimensions	54 x 30 mm
Weight	12 g

Expected range: several (10-15) kilometers with maximal output power, dependent on modulation, RF data rate, use of ECC, antennas and terrain. See the companion document **AMORES Robotics Telemetry Radio – In-Field Performance Evaluation** for details and tips on reaching optimal in-field performance.

1.2.1 Hardware

- lightweight, compact device
- uses 868 MHz ISM band with a maximum RF output power of 26 dBm (400 mW)
- dual antenna connectors support antenna diversity
- 6-pin FTDI-compatible interface for power supply and data transfer
- indicator LEDs show RX/TX activity as well as active antenna
- setup jumper for configuration

1.2.2 Software & digital radio

- provides a transparent, virtually full duplex link between two UART interfaces
- efficient selective acknowledge & retransmit scheme adds robustness against packet loss or corruption
- automatic division of channel capacity between transmit directions based on actual demand
- four alternative radio modes allow trading payload capacity for available range
- optional antenna diversity algorithm effectively mitigates fading-induced link dropouts
- optional forward error correction algorithm adds even more resiliency to long range links
- optional buffer alignment algorithm to align radio packets with user datagrams
- diagnostic mode for link capacity measurement and link quality assessment
- highly configurable UART interface allows connecting with any UART-enabled host device
- textual menu-driven configuration interface for setting up the device via a serial console
- user upgradable software – no proprietary programming device required

1.2.3 Host connection

Data transfer and power supply is provided via a single 6-pin FTDI-compatible connector header. The UART connection supports:

- full duplex operation: independent receiving and sending of frames
- configurable speed: 9600, 19200, 38400, 57600, 115200, 230400 Baud
- configurable parity (None, Even, Odd)
- configurable number of stopbits (1 or 2)
- optional RTS/CTS flow control

1.2.4 User configurability and upgradability

The software of TeRad allows end users to configure the operation of their device according to several options, in order to create a radio link that suits their specific transmission requirements. For example, it is possible to trade payload channel capacity for useful range and link robustness by changing the RF data rate and usage of ECC. Other features (antenna diversity, datagram framing etc.) can also be setup as desired.

Configuration is possible via a convenient, easy to use textual menu interface accessible via the normal UART connection (no special host software needed). Setup mode is activated by the presence of the setup jumper when the device is powered on. Configuration is persistent (stored in flash memory).

The software itself is easily upgradable via the TeRad software updater, a PC-based application that allows the end user to re-flash the device with the newest (or any supported) version of the TeRad software. This software updater is freely available for download from our product website. This future-proofing solution ensures customers purchasing TeRad will always be able to get the benefits of the latest features and bugfixes!

2 Terminology

The following terms are used in this document according to these definitions:

- **host**: any system connected to the telemetry radio, either on the ground (PC or laptop running ground control station software, etc.) or onboard (autopilot, etc.)
- **datagram** (aka. **logical packet**): a chunk of data taken from the user and transferred to the other endpoint in an atomic manner. The software guarantees atomic delivery of datagrams in correct order. Some datagrams may, however, be lost.
- **fragment**: datagrams are split into fragments. Each fragment is sent in a separate radio packet. In case of error (or timeout), missing fragments may be individually retransmitted without the need to resend the whole datagram. The receiver collects and verifies incoming fragments, eventually re-assembling and outputting the received datagram to the user.

3 Software updates

Users of TeRad have the ability to update the software in their board at their discretion via the freely downloadable **AMORES Telemetry Radio Software Updater**. Published versions of the TeRad software are flashed into the device via this tool.

The following table lists published TeRad software versions. Items of this list are available through the Software Updater. Builds are identified as calendar dates.

Versions up to 0.99 are not for general use; they were only used for pre-release product development and testing. Versions starting with 1.00 and upwards are those that shipped devices are programmed with. The column 'Keep' indicates whether the version can be loaded onto the device using the 'Keep existing device setup' option of the Software Updater (see section 3.3.2), *with the assumption that the device contains the preceding version.* In case the device contains an older version, the compatibility of all interim versions must be observed. If any of them are indicated as not keep-able, the option cannot be used. In such cases the new configuration must be set up manually.

Version	Build	Keep	Notes
1.00	2015-04-29	yes	ADD: Freq Synth recalibration only if temperature has drifted more than +/-3°C.
0.99	2014-04-14	yes	ADD: Trigger RF core Frequency Synthesizer recalibration once per minute.
0.99	2014-03-10	yes	FIX: Distance calibration.
0.99	2014-02-28	yes	ADD: Link distance measurement.
0.99	2014-02-25	**NO**	ADD: True diversity and ECC support.
0.99	2014-01-21	yes	FIX: Regression of Operating mode: RF test.
0.99	2014-01-17	yes	Baseline for field-upgradable telemetry radio software.

3.1 Using the Software Updater

The PC-based software application **AMORES Telemetry Radio Software Updater** is used for flashing the TeRad board with its embedded software. It handles factory (first) programming as well as in-field updating of already programmed units.

The device must be connected to the host running the Software Updater via its regularly used serial interface (the main 6-pin connector labeled CN1). Note: when the Entry method `pulse` is used (this generally only happens at factory programming – see below), additional two connections are required to the pins `RST` and `TEST`. These pins are located on another (unpopulated) connector (CN2, 8-pin) alongside the longer edge of the board.

3.2 Obtaining the latest build of the Software Updater

Before using it, please make sure that you have the latest release of the Software Updater. When a new TeRad software release is published, a corresponding version of the Software Updater containing the new release is made available for download. Since each release of the Software Updater cumulatively contains all versions of the TeRad software published up to the date of the particular Software Updater release, older versions should not be kept around.

The latest release of the Software Updater can always be obtained from the AMORES Project Consortium website: `http://amores-robotics.hu`. After locating the TeRad product subpage, select one of the below editions for download:

- `atrswu-win32.exe`: Windows (32 and 64 bit; Windows XP or later)
- `atrswu-Linux-i686`: Linux (32 bit)
- `atrswu-Linux-x86_64`: Linux (64 bit)

Download and run the executable suitable for your environment. Each of the above editions is a standalone self-extracting executable. There is no installation procedure involved. Apart from decompressing to the system's temporary folder (automatically deleted on application exit), they do not install anything to the filesystem or operating system registry.

3.3 Running the Software Updater

The Software Updater application's user interface is structured as a simple step-by-step wizard offering three steps.

The steps may be sequenced through in order via the **Next >** and **< Prev** buttons, or may be visited in any order via clicking on the notebook page labels at the top.

3.3.1 Select build

As the first step the TeRad software build to be downloaded into the device must be selected. Builds are presented in an ordered list (latest at the top). The first list item, which corresponds to the latest build, is selected as a default. When selecting a build, its changelog appears to the right of the list selector. This is a description of changes that went into the software compared to the previous version. User visible changes to and incompatibilities with earlier versions will be explicitly noted here.

3.3.2 Setup options

You may setup or change the configuration of the device you are about to flash.

You may also opt to keep the existing configuration of your device, in which case the device area where the setup information is stored won't be overwritten. You may only do this if the device already contains a valid configuration (that is, it is already programmed and functioning well). A further requirement of this option is that the newer version must not introduce any incompatible changes to the device setup, as compared to the existing version. You are informed about any such incompatible changes via each build's changelog at the first step.

Note: due to the fact that the Software Updater does not know the build version already in the device (if any), you must check this condition manually yourself.

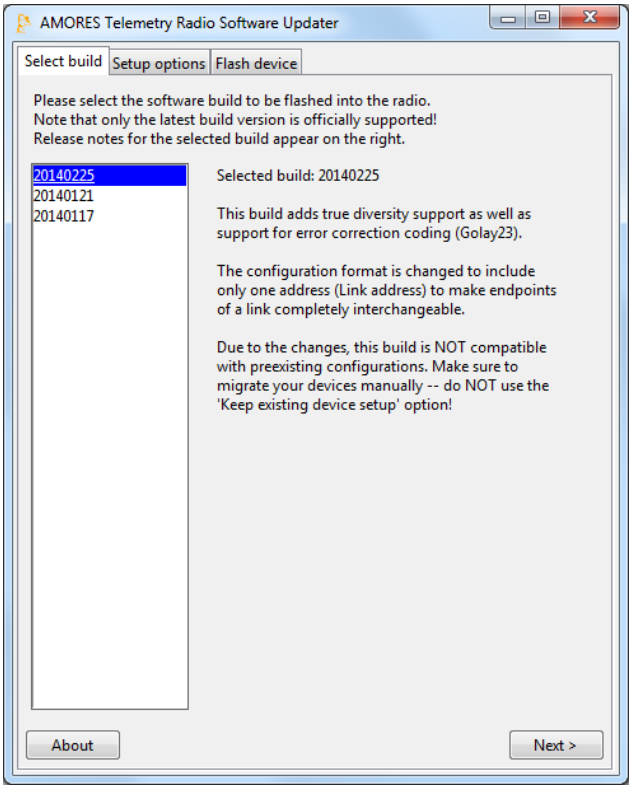


Figure 1: Software Updater – Select build

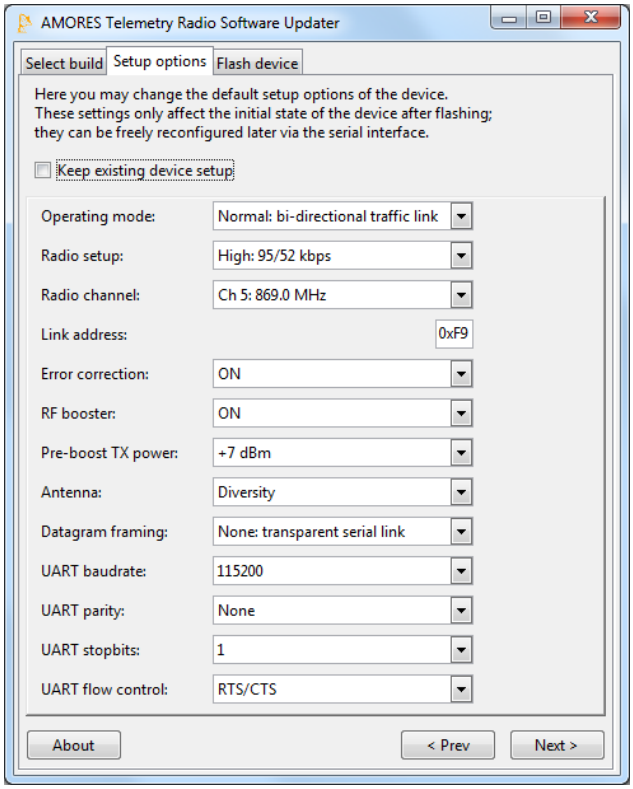


Figure 2: Software Updater – Setup options

3.3.3 Flash device

To be able to flash the device, you must first connect it to an available serial port of your computer. To enumerate serial ports on Windows, open the Device manager:

As you can see in the Windows Device Manager, in this case we are using a USB-serial converter that provides us with a virtual serial port named COM17. Please make sure that the serial port is not opened by any other program (eg. terminal emulator). Now we may proceed to the actual flashing of the device.

Important: Always place a setup jumper on the device before powering it with the intention of a software update. After powering the device, the green LED must be continuously lit, signalling that the jumper was detected and the device entered Setup mode.

There are two so-called 'entry methods' which allow the Software Updater application to interact with the device in Setup mode. They are useful in different situations and have different prerequisites.

Entry method	Prerequisites
`setup`	Device must be already programmed with some version of the software. Only the regular 6-wire FTDI-compatible connection is needed.
`pulse`	Device may be blank (or dysfunctional), but the serial interface must have RTS and DTR output pins (FTDI green and grey); connected to board pins TEST (CN2.6) and RST (CN2.7), respectively. The device setup area cannot be preserved in this mode.

Entry method `setup` is the default. It should be used by the vast majority of users; in fact it is not expected that regular users will ever need the `pulse` method. However, support for factory programming and recovering from the rare but possible case of a failed download (eg. caused by a power outage) necessitate its inclusion.

The `pulse` method does not require anything on the part of the device, but requires connections to board

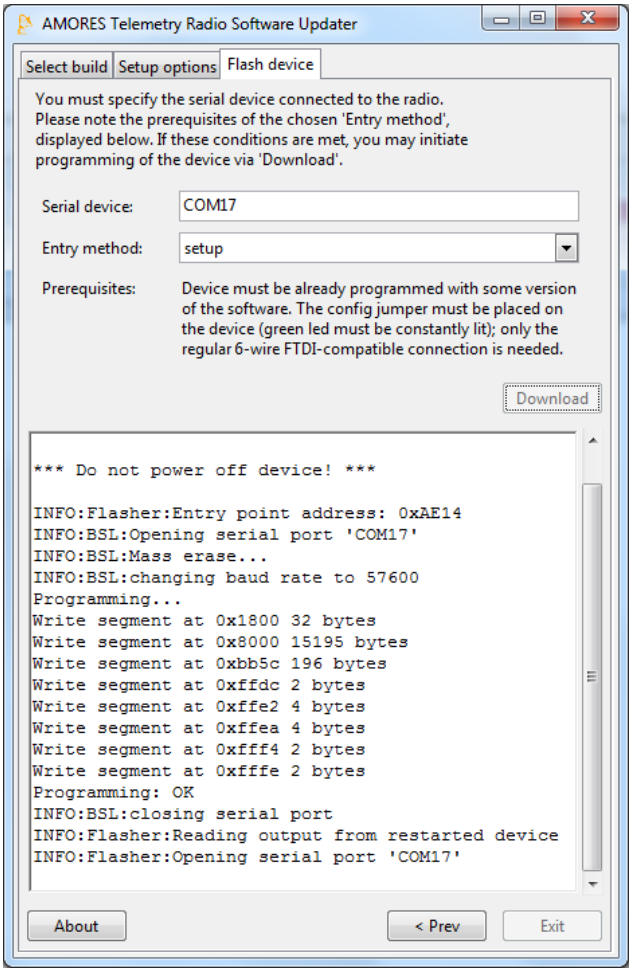


Figure 3: Software Updater – Flash device

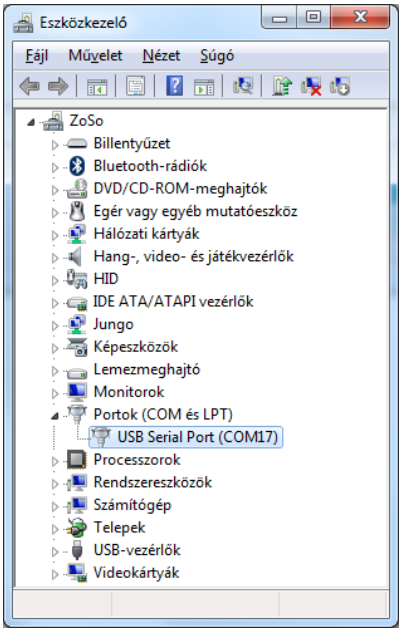


Figure 4: Windows Device Manager

pins `TEST` and `RST` via a serial adapter that supports not only an RTS, but also a DTR line. This method should be used only if (for example due to an earlier failed programming attempt) the device is in a state where the TeRad software does not correctly start on power up. It is also the method used for factory programming of a blank device.

Please remember that after a download, each device MUST be power-cycled (powered off before being powered on again) before any further use. However, the software updater application itself can be used to program several devices in a row without restarting it. A common use case is the flashing of a pair of devices to form a link:

- configure and flash the first device;
- disconnect the device from the 6-pin connector, and connect the second device;
- press the Download button again.

The download process itself takes about 10 seconds. Upon completion of a download, the device's setup mode is initiated again and its output is read back into the text area. You should at least skim through this output and verify that the device is now indeed running the selected software build with the configured setup options.

4 Host connection

Normally, TeRad provides a transparent serial link over the air: two devices forming a link are equivalent to a serial cable connecting two hosts (that is, two pieces of *data terminal equipment* in RS-232 parlance, with TeRad devices playing the role of *data communication equipment*) – only without the cable.

At both ends of this link, the host is connected to TeRad via UART through the 6-pin CN1 connector (see table). The host also powers TeRad via this connector. Please make sure that the host connector is capable of supplying the current necessary for the operation of TeRad (this depends on the configured transmit power).

Pin	Signal	Driver	FTDI color
CN1.1	GND	n/a	black
CN1.2	CTS/	device	brown
CN1.3	+5V	host	red
CN1.4	RXD	host	orange
CN1.5	TXD	device	yellow
CN1.6	RTS/	host	green

The UART connection implemented in the TeRad software has the following properties:

- Full duplex operation: independent receiving and sending of frames;
- Configurable speed: 9600, 19200, 38400, 57600, 115200, 230400 Baud;
- Configurable parity (None, Even, Odd);
- Configurable number of stopbits (1 or 2);
- Optional RTS/CTS flow control (strongly encouraged).

4.1 Flow control operation

When enabled, RTS/CTS flow control is used by TeRad. It must also be supported by the host. The flow control consists of two active-low signals `RTS/` and `CTS/`:

- `RTS/` must be driven by the host and must be low when the host is prepared to receive further data frames (this should be the default state of the `RTS/` line). It must be pulled high (inactive) when the host needs to temporarily inhibit TeRad from sending data. The software will then buffer data and resume sending frames over UART when `RTS/` is pulled low by the host. It is guaranteed that TeRad will send at most one extra UART frame after the host has pulled `RTS/` high.
- `CTS/` is driven by TeRad and is low when it is capable of receiving further data. It will be pulled high when the in-memory inbound buffer has less than 64 bytes of free space. The host must suspend sending of data within this amount to prevent data loss over the UART. TeRad will re-activate (pull low) `CTS/` when its inbound buffer free space exceeds 256 bytes.

4.2 Throughput and latency considerations

When a pair of radios is communicating in normal mode, they arbitrate the usage of the radio channel (transmission direction) between themselves. Even when there is no user data to be sent over the air, the two radios keep in touch with each other, regularly exchanging the right to immediately start sending data.

The rate at which this exchange is done (5 Hz) determines an initial worst-case latency on the start of transmission: in case the other end has just acquired the right to send when user data starts arriving over the serial link, the arriving data will need to be buffered for 0.2 seconds. The UART receive buffer is 1024 bytes, which at higher baudrates will have easily filled by this time, and some data sent by the host will eventually be dropped. Thus it is important to implement flow control between the radio and host.

Note: this initial latency is not equivalent with the general transmit latency of the data transmission, which – once a logical packet is scheduled for transmission – is much, much lower. As long as at

least one of the endpoints keeps sending payload, logical packets will be scheduled continuously and the aforementioned arbitration exchange mechanism will not come into play.

Also, as long as an endpoint has no payload to send, it will immediately give up its right to transmission at each round, so the sending endpoint will not experience the aforementioned startup latency even in case of long gaps between blocks of payload to send.

5 Configuration

TeRad contains a persistent (stored in Flash memory) end user configurable setup. This setup specifies all parameters not wired into the software.

Accessing the setup is possible via the **Setup mode** activated by placing the setup jumper on the device before powering it up. When the jumper is present on power up, the software of the board will enter Setup mode on startup. In this mode the radio hardware is configured to a safe (non-transmitting, powered down) setting. Thus it is safe to power on the device without any antennas connected, as long as the jumper is in place.

Note: the presence of the jumper is only checked once as part of the TeRad software's power up sequence. Connection (or removal) of the jumper does not make any difference during subsequent operation of the device.

Note: the method of Setup mode entry (power up with jumper on) is chosen to prevent accidental in-flight reconfiguration of the device (or even accidental entering of Setup mode, where the radio cannot function and which it cannot exit on its own). This is a conscious design decision for maximum operational reliability; addition of any features aiming for remote reconfiguration capabilities (eg. configuration via AT commands) are not considered.

5.1 Overview of Setup mode

On powering up the device with the setup jumper on, Setup mode is entered. The green LED is continuously lit while in this mode.

When in Setup mode, the device communicates via the UART connection with the following fixed parameters (regardless of the values of UART configuration items):

```
9600 baud - no parity - 1 stop bit - no flow control
```

Therefore, any computer with a serial port and ordinary terminal emulator software can be used to access Setup mode; no special client application needs to be installed.

Immediately after device power up, a banner is printed containing the current build version. A menu listing follows showing setup options with their currently active values. Altering these options is possible by pressing the designated keys in the left column.

```
************************************************
**     AMORES Telemetry Radio -- Setup Mode     **
**      Firmware v1.00   build Apr 29 2015      **
**           http://amores-robotics.hu          **
************************************************

Press key to change       [current setting]

0 - Operating mode        [Normal: bi-directional traffic link]
1 - Radio setup           [High: 95/52 kbps]
2 - Radio channel         [Ch 5: 869.0 MHz]
3 - Link address          [F9]
4 - Error correction      [OFF]
```

```
5 - RF booster            [ON]
6 - Pre-boost TX power    [+7 dBm]
7 - Antenna               [ANT 1]
8 - Datagram framing      [None: transparent serial link]
9 - UART baudrate         [115200]
A - UART parity           [None]
B - UART stopbits         [1]
C - UART flow control     [RTS/CTS]

W - Write configuration to FLASH and exit
X - Exit without committing changes

>
```

For example, to change the **Radio channel** setting, press `2`. A single keystroke is all that is needed; don't press Enter. A submenu is presented showing the setup item, its current value, and appropriate choices for the new value:

```
> 2
==[ Radio channel         [Ch 5: 869.0 MHz]
0 - Ch 0: 868.0 MHz
1 - Ch 1: 868.2 MHz
2 - Ch 2: 868.4 MHz
3 - Ch 3: 868.6 MHz
4 - Ch 4: 868.8 MHz
5 - Ch 5: 869.0 MHz
6 - Ch 6: 869.2 MHz
7 - Ch 7: 869.4 MHz
8 - Ch 8: 869.6 MHz
9 - Ch 9: 869.8 MHz

>
```

Again, pressing any of the offered numbers on the left will choose the indicated setting for `Radio channel` (the setup item we're now in). Let's set the channel to **Ch 3** by pressing `3`:

```
> 3
--> Radio channel         [Ch 3: 868.6 MHz]

Press key to change       [current setting]

0 - Operating mode        [Normal: bi-directional traffic link]
1 - Radio setup           [High: 95/52 kbps]
2 - Radio channel         [Ch 3: 868.6 MHz]
3 - Link address          [F9]
4 - Error correction      [OFF]
5 - RF booster            [ON]
6 - Pre-boost TX power    [+7 dBm]
7 - Antenna               [ANT 1]
8 - Datagram framing      [None: transparent serial link]
9 - UART baudrate         [115200]
A - UART parity           [None]
B - UART stopbits         [1]
C - UART flow control     [RTS/CTS]

W - Write configuration to FLASH and exit
X - Exit without committing changes

>
```

Subsequent output starts with acknowledging our change, followed by a full reprint of the main menu where the change to the current setting is now also reflected. Now we can change another item, or go on to save our changes to Flash (or abandon the changes we made).

When changing the `Link address` setting, the input method is different, but only slightly so. Let's change the address to `a5`. This begins by pressing `3`, since we need to select the setup item `Link address` first. Then just key in `a5` (press `a`, then press `5`) without hitting Enter:

```
> 3
==[ Link address          [F9]
Enter new value as a 2-digit hex number:
> a5
--> Link address          [A5]
```

You could have entered `A5` just as well - note that in the acknowledgment following your input, the value is already parsed and printed in upper case. Note that in case of a single-digit address, you must type a leading `0` so the input will be two digits.

Please note that individual changes are not automatically saved – you must explicitly press `w` or `W` in the main menu to overwrite the setup in permanent Flash storage, so it will be effective from the next power up. But only do this if you are satisfied with all settings – after writing the setup to Flash, the software will exit and you will need to power cycle the device to get into a functional Setup mode again. The red LED will light up for the duration (a brief moment) of writing the setup to Flash.

As you have already guessed, the menu is not case sensitive – for example, you can type either `a` or `A` to access the UART baudrate setting.

Non-parseable (garbage) input is handled gracefully; characters not allowed at the given point are rejected with messages

- `Selection out of range` (at menu choices);
- `Invalid character input` (at address entries);
- `Eh?` (in the main menu).

Now that we've covered general usage of Setup mode, let's dig deeper into each item.

Important:

Configuration items below up to and including 'Pre-boost TX power'
MUST be the same on both endpoints in order to form an operational radio link!

5.2 Operating mode

```
0 - Normal: bi-directional traffic link
1 - Diagnostic: link quality assessment
2 - RF test: continuous transmission
```

The radio software may be operating in one of the above modes. To use the radio for payload transmission (normal usage; connecting two serial interfaces over the air) choose **Normal** mode.

The second setting may be used to run the radio in a special **Diagnostic** mode. In this mode, the two endpoints generate dummy traffic between each other. The UART interface is used to report statistics about several aspects of operation:

- received/transmitted bytes
- number of errors corrected by ECC
- fragment transmission retries

- dropped (lost) datagrams so far
- selected internal control variables
- antenna selection counters (diversity operation)
- RSSI statistics
- link distance (estimated from propagation delay)
- device temperature

Some of these values are reported for both endpoints (values of the remote device are transmitted and displayed as well as the local values). Other values are displayed for the local endpoint only. Please see section 6 for full reference.

Diagnostic mode may be used for link quality testing (measurement of signal levels, payload capacities, error rates) and general testing and debugging of device operation. Naturally, both endpoints must be in this mode.

RF test mode puts the radio into an RF measurement mode in which no data exchange takes place at all. The radio will be transmitting an infinite sequence of random bytes, effectively driving the RF circuits with a stationary signal. This is used for RF measurements only, when the antenna outputs are connected to RF equipment with proper impedance termination. Do not connect the radio to antennas (or leave the antenna ports unconnected) when running in this mode! The latter may actually cause permanent damage to the device. If you are not really sure what you are doing, **do not use RF test mode at all!**

5.3 Radio setup

```
0 - Extreme: 185/104 kbps
1 - High: 95/52 kbps
2 - Medium: 46/25 kbps
3 - Low: 32/17 kbps
```

This setting controls the radio modem: air speed, modulation format, and lots of accompanying internal settings. The most obvious implications of this setting are payload capacity and usable range. As a quick reference, approximate payload capacities (with Error correction OFF/ON respectively) are displayed as part of the option.

Available radio modes are summarized by the following table. For each setting, the table gives the approximate sustainable net payload capacity and the nominal UART baudrate matching that capacity with and without error correction (see section 5.6).

Designation	**Modulation**	**Air speed** [kBaud]	**Error correction**	**Payload capacity** [kbit/s]	**Matching UART speed** [kBaud]
Extreme	MSK	274.5	OFF	185	230.4
			ON	104	115.2
High	2-GFSK	136	OFF	95	115.2
			ON	52	57.6
Medium	2-GFSK	67.6	OFF	46	57.6
			ON	25	38.4
Low	2-GFSK	48	OFF	32	38.4
			ON	17	19.2

NB.: When matching payload capacity and UART speed setting, keep in mind that the actual UART capacity is at most 80% of its line speed (assuming no parity and 1 stop bit as usual) and may be as low as 67% (in case of 2 stopbits with parity enabled). Taking this into account, it is easy to see that the

available payload capacity over the radio closely matches the corresponding UART speed (or, in case ECC is used, even exceeds it). The only exception is Medium with ECC on - this is a bit too slow for a 38.4 kBaud UART (which can sustain 30 kbps of payload data) but too fast for a 19.2 kBaud UART.

All the above boils down to a very simple bottom line recommendation: *you can easily select the radio mode recommended for you based on the UART speed you demand* (per the above table). Note that in this case the UART speed that counts is that which your payload actually requires (nearly or completely saturates) to be transmitted.

Of course, you are not constrained to setting the UART speed as per the above; it is merely an indication of the setting that will provide an even payload capacity throughout your data path with no bottlenecks. You may have legitimate reasons to deviate from these settings; in particular, when interfacing the radio with other hardware (eg. an autopilot or other on-board equipment) you may be forced to use the UART setting of this other device. This sets an upper bound of data that the device is able to communicate to/from the radio; the actual effectively used rate may be much lower.

Please note, however, that in case the UART speed you select is lower than the one matching the radio setting, data may be correctly transmitted over the radio link only to be dropped at the receiving end. This is a problem especially if there is no RTS/CTS flow control over the serial line.

Naturally, higher air speed brings with it not only greater payload capacity, but also reduced receiver sensitivity and an increased susceptibility to noise and interference. The ultimate result is a shorter usable range for the same transmit power and antennas. As a rule of thumb, reverse the radio setting designation to get an indication of its robustness and range: go for a Low setting to obtain Extreme range/robustness, and so on.

As a corollary, it really pays to research the actual payload capacity requirement of your application before deciding upon the radio mode to use. You should always select the lowest setting that is able to accommodate the capacity requirement of your application, since this will give you the best available robustness of transmission and longest range.

In case extra link robustness is required, consider enabling error correction. Please see section 5.6 for further info.

5.4 Radio channel

```
0 - Ch 0: 868.0 MHz
1 - Ch 1: 868.2 MHz
2 - Ch 2: 868.4 MHz
3 - Ch 3: 868.6 MHz
4 - Ch 4: 868.8 MHz
5 - Ch 5: 869.0 MHz
6 - Ch 6: 869.2 MHz
7 - Ch 7: 869.4 MHz
8 - Ch 8: 869.6 MHz
9 - Ch 9: 869.8 MHz
```

Here you may choose which channel to use for the radio link. This specifies the actual center frequency of the channel – the bandwidth of the channel depends on the modulation and air speed (radio setup). Note that the bandwidth may be larger than the channel offset (200 kHz).

It is generally not advisable to use multiple TeRad links (multiple pairs of devices) at the same physical location, even with devices of different links configured to different channels. If you must absolutely do so, use a separation of at least 4 channels between the links, for example use channels 0, 4 and 9 for three independent point-to-point links. But beware of a loss of receiver sensitivity caused by masking from other links.

5.5 Link address

The radio software supports a point-to-point link protocol. The link address is an identifier number assigned to the link, and is included in the header of each radio packet exchanged by the endpoints. Each radio discriminates incoming packets based on this address. Only packets with a link address matching the device's own setting will be considered; all others will be dropped.

A pair of radios must be configured so both of them are set to the same link address.

Each address is a one-byte value. It is not recommended to use `0x00` and `0xFF` (255) due to their conventional use as broadcast addresses. (Technically these values *can* be used, but it is better to stay away from them).

5.6 Error correction

```
0 - OFF
1 - ON
```

The software supports a half-rate error correction coding (ECC) scheme to add extra robustness against random bit errors. The ECC used is the well known binary Golay code (famously used by the Voyager spacecrafts).

If enabled, all data of transmitted radio packets are protected by ECC (the headers as well as the payload fragments) resulting in a significant robustness improvement against random noise. Since there is less need for fragment retransmissions, the realised payload capacity tends to be more constant (sporadic retries do not eat into it).

The Golay encoder emits a 24 bit codeword for each 12 bits of input, thus halving the available payload capacity for each radio setup mode. Actually, it is a bit better than that (see values in section 5.3) for the reason that radio packet preambles and inter-packet guard times need not be duplicated.

The upside is that when decoding, up to three bit errors in each 24 bit codeword are corrected. This amounts to tolerance against a whopping 12.5% BER (bit error rate) value – provided, of course, that the bit errors are evenly distributed across the received data.

Regardless of whether ECC is applied to the link or not, the received data is still error checked by means of 16 bit CRC sums for each 64 bytes of payload. Any fragment with a CRC error is discarded by the receiver, and very likely (timeouts and retry counts permitting) re-transmitted by the sender. Thus, the chance of data corruption over the air is negligible (but beware of serial connections without RTS/CTS flow control, where – especially at higher baud rates – data loss happens easily and undetected).

5.7 RF booster

```
0 - OFF
1 - ON
```

This setting controls whether the CC1190 booster chip on the board is used. More precisely, this setting controls the HGM (High Gain Mode) input of the CC1190. If HGM is off, the booster outputs with the same power setting as received from the CC430 (see section 5.8).

5.8 Pre-boost TX power

```
0 - +10 dBm
1 - +7 dBm
2 - +5 dBm
3 - 0 dBm
4 - -10 dBm
```

```
5 - -15 dBm
6 - -20 dBm
7 - -30 dBm
```

This setting controls the transmit power of the CC430 RF core which drives the CC1190 booster chip. In case the RF booster is off, this is approximately equal to the transmit power on the antenna output.

In case the booster is on, the optimal setting for driving it to full transmit power is +7 dBm. This will result in full transmit power (+26 dBm / 400 mW) on the antenna output without overdriving the booster input.

Reminder:

All settings up to this point MUST be the same on both endpoints in order to form a usable link!

5.9 Antenna

```
0 - ANT 1
1 - ANT 2
2 - Diversity
```

This setting controls which antenna port to use. You may use either one of the antenna ports if you have only one antenna. However, for mobile over-the-air links between UAVs and their ground stations, it is strongly recommended to use diversity mode with two antennas connected. This way the radio will always be able to select the antenna with better momentary signal strength for receiving each radio packet. This is especially important for maintaining a stable link in case the endpoints are moving arbitrarily against each other and the terrain.

The diversity scheme employs received signal strength (RSSI) measurements on both antennas during the preamble of each incoming radio packet. Thus, each packet payload is received on the antenna with the stronger signal. Since radio packets are short, this provides a frequent re-evaluation of the selected antenna – effectively mitigating the deep signal dropouts that result from multipath fading.

5.10 Datagram framing

```
0 - None: transparent serial link
1 - AirGuardian
2 - MAVLink 0.9
3 - MAVLink 1.0
```

This setting specifies the algorithm for choosing datagram boundaries in the incoming data stream received from the UART interface. This may be important because datagrams are atomically handled by the TeRad software – they are either received successfully (permitting multiple fragment retries), re-assembled and output to the remote host intact, or eventually dropped completely. Thus, it might make sense to align datagrams to contain one or more complete user-level data packet. This way, most transmission errors will be visible to the user level as ordinary packet loss with no incompletely arriving (corrupt) user-level data packets.

With a setting of **None**, datagrams are assembled greedily as soon as the data is present in the inbound buffer and the endpoint receives the right to transmit. Datagram payload capacity permitting, available input data is always completely consumed by the datagram to be sent.

With a setting of **AirGuardian**, the software tries to send one or more AirGuardian sentences starting with the '$' or '#' character, until the next separator. This potentially leaves some data (unterminated sentence just being received via UART) in the input buffer.

With a setting of **MAVLink**, the software tries to send one or more MAVLink packets, potentially leaving some data (incomplete MAVLink packet just being received via UART) in the input buffer. Note that different MAVLink versions have different start-of-frame field values, so make sure to match the version your system is using.

The UART interface employs so-called *idle line detection*, which enables the selection of all available inbound data regardless of the chosen setting. The UART line is considered idle if there has been no incoming data for at least 10 bits time (measured according to the UART baudrate setting). This means that at least a whole UART frame's worth of time has passed without the host sending data. In such case the software assumes that whatever user packet the host has been sending is now completely received and ready to be transmitted as a datagram.

5.11 UART baudrate

```
0 - 9600
1 - 19200
2 - 38400
3 - 57600
4 - 115200
5 - 230400
```

Set the baudrate for data communication with the host. See the table in section 5.3 to match this to the payload capacity of your radio link.

5.12 UART parity

```
0 - None
1 - Even
2 - Odd
```

Set the parity of the UART interface. Inbound UART frames with a parity error are discarded.

5.13 UART stopbits

```
0 - 1
1 - 2
```

Set the number of stop bits in UART frames.

5.14 UART flow control

```
0 - None
1 - RTS/CTS
```

Enable or disable RTS/CTS flow control. It is strongly encouraged to use hardware flow control for reasons discussed in section 4.2.

6 Diagnostic mode

In case both endpoints are configured to be in diagnostic mode (see section 5.2) the devices do not carry payload traffic. Instead, they generate dummy test traffic to send. The RX/TX buffers are continuously being filled up just as though the user was submitting large amounts of data to both endpoints. In other words, whenever the endpoint's packet layer looks for a new user datagram to send, it will be able to fetch one (with the largest possible size).

Instead of providing the host data connection, the UART interface is expected to be connected to a text console (terminal emulator program). The serial connection is used to print human readable status information to this console. The UART settings of the device (baudrate, etc.) are also effective in this mode, so make sure to open the serial port with the same settings in the terminal emulator.

6.1 Regular status output

When the device is turned on, it will start with printing its version and configuration:

```
AMORES Telemetry Radio v1.00  build Apr 29 2015

Operating mode       Diagnostic: link quality assessment
Radio setup          High: 95/52 kbps
Radio channel        Ch 5: 869.0 MHz
Link address         0xF9
Error correction     ON
RF booster           OFF
Pre-boost TX power   0 dBm
Antenna              Diversity
Datagram framing     None: transparent serial link
UART baudrate        115200
UART parity          None
UART stopbits        1
UART flow control    RTS/CTS
```

These items represent the setup of the device as seen in Setup mode (see section 5). You should be familiar with them by now. Regular status output comes afterwards, with information arranged in columns. An example snippet is shown below:

```
========= Ack'd payld bytes | Local   Remote  | Dg lost |  Recvr  | Antenna | Local --- RSSI dBm --- Remote |  Link |  Tj *C
             RX    TX Total | ECC Rty ECC Rty |  RX  TX | Att H G | Use 1,2 |    Min    Avg    Max |    Avg |  D km | Loc Rem
[0:47:01]  3069  4092  7161 |   1   0   7   0 |   2   5 |   0 0 0 |  25  28 |  -99.5  -98.5  -97.0 |  -97.9 |   0.0 |  39  23
[0:47:02]  3069  3069  6138 |   9   0   8   0 |   2   5 |   0 0 0 |  42  15 | -100.0  -98.6  -97.5 |  -96.4 |   0.0 |  39  23
[0:47:03]  3069  3069  6138 |   3   0  12   0 |   2   5 |   0 0 0 |  46  10 | -100.0  -98.6  -97.5 |  -97.6 |   0.0 |  39  23
[0:47:04]  4092  3069  7161 |   0   0  10   0 |   2   5 |   0 0 0 |  38  16 |  -99.0  -98.0  -97.0 |  -97.4 |   0.0 |  40  23
[0:47:05]  3069  3069  6138 |   0   0   5   0 |   2   5 |   0 0 0 |  43   8 |  -99.0  -97.7  -97.0 |  -96.6 |   0.0 |  39  23
[0:47:06]  3069  3069  6138 |   1   0   1   0 |   2   5 |   0 0 0 |  44   7 |  -99.0  -97.7  -95.5 |  -96.8 |   0.0 |  40  23
[0:47:07]  3069  4092  7161 |   2   0   2   0 |   2   5 |   0 0 0 |  49   5 |  -99.0  -97.9  -97.0 |  -97.4 |   0.0 |  39  23
[0:47:08]  3069  3069  6138 |   0   0   6   0 |   2   5 |   0 0 0 |  52   5 | -100.0  -98.1  -97.0 |  -96.5 |   0.0 |  39  23
[0:47:09]  3069  3069  6138 |   3   0   7   0 |   2   5 |   0 0 0 |  52   5 |  -99.5  -97.9  -97.0 |  -97.0 |   0.0 |  39  23
[0:47:10]  4092  3069  7161 |   0   0   3   0 |   2   5 |   0 0 0 |  53   0 |  -98.5  -97.4  -96.0 |  -96.7 |   0.0 |  39  23
```

Each row corresponds to a time interval of one second; the beginning of the row contains the respective timestamp (measured in hours, minutes and seconds since power on). Header lines are reprinted every minute.

6.1.1 Traffic information

After the timestamp, the first three columns (under `Ack'd payld bytes`) contain the number of transferred bytes in each direction (received and transmitted) and their sum. These numbers represent bytes of *completely and correctly transferred payload*: they are only incremented when a transfer is acknowledged

to have completed without any remaining errors. The total figure indicates the payload capacity achieved with the current device setup, under the current conditions (as described by subsequent columns). Remember that under real use (**normal** mode), the subdivision of this total capacity will be automatic based on the actual momentary demand in each direction. The total, however, will be very close to the indicated figure.

The next four columns (in the next section between vertical lines) report information regarding transmission errors. Data is shown for both endpoints `Local` and `Remote`:

- `ECC`: number of codewords altered by error correction. This pertains to the inbound traffic of each endpoint.
- `Rty`: number of fragment retries done. This pertains to the outbound traffic of each endpoint.

In the next section, the numbers under `Dg lost` show *cumulative* counts of lost datagrams for the local endpoint, in both receive and transmit directions. These are the only printed statistics that are cumulative, collected since the device was turned on. By lost datagrams we refer to transmissions that did not reach their destination, most likely because the packet handler ran out of the maximum number of retries permitted for each datagram. Frequent dropping of packets is usually a good indication that the limit of the usable range is being approached.

The link – as perceived by the user – is primarily affected by dropped datagrams, which manifest themselves as missing blocks in the transferred data stream. On the other hand, corrections made by ECC (if error correction is used) do not have any effect as long as they can contain all bit errors. In case the number of bit errors is more than the ECC can handle (more than 3 in any 24-bit codeword), data will remain corrupt after having passed through the ECC decoder; this will eventually result in a CRC error for the respective fragment. The same result is reached when ECC is not used. In this case any bit errors in the transmission lead to a CRC error for the fragment. Such fragments are retransmitted within a (reasonably bounded) number of retry cycles, which usually allow these transfers to complete successfully (or the datagram is dropped). However, retries have a small effect on the link capacity, because they take time and thus 'steal' payload capacity from fresh (inbound buffered) data. This is nothing to worry about in case fragment retries are sporadic, but it bears a pronounced effect on the link when there are 10 or more retries each second, for multiple seconds. The net result is that in the face of these retries, the payload capacity will decrease, as indicated by the traffic count columns on the left.

At the end of each minute, a tally of local fragment retries is displayed in one line:

```
lp_retry_stats: 0:183 1:3 2:1 3:0 4:0 5:0 6:0 7:0 8:0
```

The above line says: there were 183 datagrams transmitted to the remote end with 0 fragment retries; 3 datagrams transmitted with 1 retry; 1 datagram took 2 retries; no datagram that was eventually transferred took more than that. (Only those datagrams that were successfully transferred are included in this tally; dropped datagrams are counted separately.)

The last value after `8:` is the number of datagrams that took 8 *or more* retries. The absolute maximum number of retries is 255. 8 retries are guaranteed for each datagram; if it is still not fully transferred at that point, further retries may be permitted depending on the load situation (whether or not the host has already submitted further data to transmit).

6.1.2 Status and sensory data

Under the heading `Recvr`, three values are shown. They are internal control variables of the radio receiver adjusted by the software to ensure optimal performance under all conditions. As such, they are not particularly relevant to the end user. For the sake of completeness, their meanings are listed below:

- `Att`: attenuation applied to the receiver input. This is a value in dBs which can be increased in steps of 6 to a maximum of 18 dB (displayed as a negative value signifying that it is attenuation). Indicated RSSI figures (see below) are affected by this setting, because the attenuator precedes the

AGC and demodulator. Its role is to protect the demodulator from such high signal levels that would cause saturation of the A/D and would thus prevent the receiver from locking on to the incoming signal. The displayed value is nominal (as dictated by the software), the real attenuation (as realised by the radio core) could be different. During real world usage, this number is almost always zero except when the two devices are brought very close (within a few tens of meters) to each other.

- `H`: setting related to the amount of AGC hysteresis.
- `G`: setting related to the AGC's digital variable gain amplifier.

The next section labeled `Antenna Use 1,2` displays diversity decision counts in favour of each antenna. The first number counts the times ANT1 was chosen, the second number counts the times ANT2 was chosen. Naturally, in case diversity is disabled, one of the numbers will always be 0. Diversity decisions are made at the start of each inbound radio packet, be it a fragment (carrying payload) or an ACK.

RSSI values are displayed as measured by the local as well as the remote endpoint. Values shown are the minimum, average and maximum RSSI readings during the past one second. (Readings are stored after the diversity decision is made, so the RSSI measured on the momentarily selected – better – antenna is used at all times.) The average of the remote RSSI is displayed as the fourth column, and should be close to the locally measured average (provided, of course, that the two devices are configured for equal transmit power, which should always be the case for an operational link).

The last-but-one section labeled `Link D km` displays an estimate of the link distance in kilometers, with a nominal precision of 100 meters. This estimate is derived from a measurement of the variation in the propagation delay of the signal. It is highly experimental; please **do not rely on this figure** for precise measurements. Its aim is not to be a source of telemetry data, rather it is implemented to provide an idea of whether the endpoints are 100m, or 1 km, or 10 km apart. This information may be used by the radio software to aid its own decision making. Note: the measurement may be subject to a constant offset error. Elimination of this error is, however, quite straightforward: just place the two radios close to each other (within a few meters) and observe the distance indication. The value should be within ±0.1 km (ideally zero). Any other reading should be noted and later subtracted from readings (preferably as a part of post-processing the recorded diagnostic mode output).

In the last section under `Tj *C` the junction temperatures of both local and remote devices are displayed in Celsius degrees. The measurements are made via the internal temperature sensor of the CC430 chip, and are not very accurate (they may be subject to substantial offset error). However, they provide good means to detect temperature drift, which necessitates recalibration of the on-chip digital frequency synthesizer. This recalibration enables continuous link operation over large temperature offsets affecting any endpoint device.

6.2 Event reporting

Besides the regular output described above, information about occasional events is printed. These events come from the packet layer and are usually meaningful to the developer only. Nevertheless, we list most of them here along with a brief description.

Collisions

```
COLLISION, aborting transmit lpseq_tx=<SEQ>
COLLISION, re-scheduling transmit lpseq_tx=<SEQ>
```

These messages are issued in situations when – after a de-synchronisation due to lost contact between the endpoints – the devices both transmit at once. When this condition is detected, the software immediately ceases transmission and issues one of the above messages.

- In case the datagram can be unread back into the inbound FIFO, the transmit is re-scheduled for later retry.

- If, since this datagram was read out, the FIFO has been filled up to such extent that the datagram cannot be unread, it is dropped. In this case the number of outbound datagrams lost (`Dg lost TX` in the traffic output) is incremented by one.

Aborted datagram transmissions

`TX ABORTED lpseq_tx=<SEQ> frag_map_tx=<FRAGMAP>`

This message is printed whenever a datagram transmission is aborted because its permitted number of retries have been used up and an ACK verifying reception of all fragments still has not been received. In such an event, the number of outbound datagrams lost (`Dg lost TX` in the traffic output) is incremented by one.

`<SEQ>` is the datagram sequence number (not really meaningful to users). `<FRAGMAP>` is a hex value with each bit encoding the reception of a fragment. It is taken from the last received ACK (if any) or zero (if no ACK has ever been received). Bits are LSB first, so `0xfffe` means only the first fragment is missing; `0x7fff` means only the last fragment is missing. (Normally, datagrams may consist of a maximum of 16 fragments, so `<FRAGMAP>` is a 16-bit value. In lower speed settings, the maximum number of fragments can be lower, and so the values corresponding to the above situations will be different.)

Missed datagrams

`LPSEQ MISMATCH got=<SEQ> exp=<SEQ>`

Datagrams have sequence numbers which allow the receiver to detect missed datagrams. In such cases the above message is printed with the actually received sequence number and the expected sequence number which should have been received. The difference of these values (in a modulo fashion) is used to increment the number of inbound datagrams lost (`Dg lost RX` in the traffic output).

RF core reinits

`CC1101 REINIT [<N1> <N2>] [<N3>]`

Service message notifying that the CC1101 idle timeout has elapsed. This is not necessarily a problem in itself; the software reinitializes the CC1101 radio core in case there has been no traffic for a certain length of time (about one second). This is to prevent 'stuck' conditions where the radio core is in a state when it is not able to receive packets even though it is put in the RX state. `<N1>`, `<N2>` and `<N3>` are hex numbers coding internal state variables of the software and the radio; they are meaningful to the developer only.

7 Appendix

7.1 Radio parameters

The below table shows receiver filter bandwidth and deviation (modulation depth) for each radio mode.

Designation	Modulation	Air speed [kBaud]	Receiver filter bandwidth [kHz]	Deviation [kHz]
Extreme	MSK	274.5	541.6	N/A
High	2-GFSK	136	203	76.2
Medium	2-GFSK	67.6	135	41.2
Low	2-GFSK	48	81	25.4

7.2 Propagation

7.2.1 Free space propagation loss

- with SI units (for physicists only ☺): $\mathsf{FSPL} = \left(4\pi d\frac{f}{c}\right)^2$
- with customary radio engineering units (dB, km, MHz): $\mathsf{FSPL_{dB}} = 20\log d + 20\log f + 32.45$, wherein substituting $f = 869$ MHz yields $\mathsf{FSPL_{dB}} = 20\log d + 91.23$. The following table covers the distance range we (as TeRad users) might be interested in.
- rule of thumb: a +20 dB increase in link margin is equivalent to a factor of 10 in distance; +6 dB equals a factor of 2.

d [km]	FSPL [dB]
0.1	71.2
0.2	77.3
0.3	80.8
0.5	85.2
1	91.2
2	97.3
3	100.8
5	105.2
10	111.2
20	117.3
30	120.8
50	125.2
100	131.2

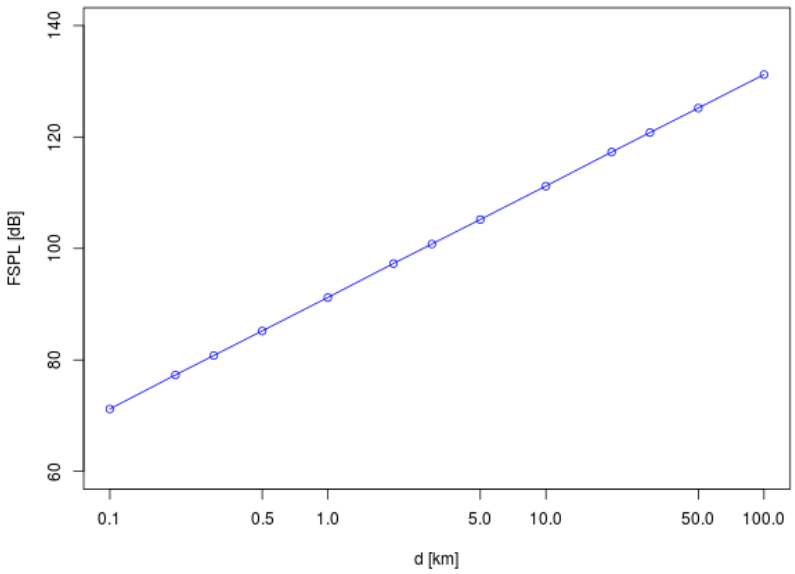


Figure 5: Free space propagation loss ($f = 869$ MHz)

7.2.2 Fresnel zone clearance

Radius of the first Fresnel zone in meters, for different link distances at $f = 869$ MHz, using the formula $r_{F_1} = 8.657\sqrt{\frac{D}{f}}$:

Distance [km]	r_{F_1} [m]
0.1	2.9
0.2	4.2
0.5	6.6
1	9.3
2	13.1
5	20.8
10	29.4
20	41.5
50	65.7
100	92.9

7.2.3 ISI-critical path divergence

Assuming a symbol time overlap of 10% (limit of ISI where symbols remain detectable) yields the following path divergences for individual radio speeds:

Radio mode	Air speed [kBaud]	Symbol length [μs]	Critical path divergence [m]
Extreme	274.5	3.6	108
High	136	7.4	222
Medium	67.6	14.8	444
Low	48	20.8	624