nanoANQ User Guide

1.5

NA-13-0275-0025

Document Information

Document Title:	nanoANQ User Guide
Document Version:	1.5
Current Date:	2020-12-07
Print Date:	2020-12-07
Document ID:	NA-13-0275-0025
Document Author:	MBOR

Contents

1. Scope

This document provides information about the configuration and operation of the nanoANQ RTLS anchors implemented as nanoANQ EM (CSS) [1], nanoANQ EA [2] nanoANQ-c / pcbANQ-c [3] and nanoANQ EM ER (UWB) [4].

This document considers that all anchor firmware revisions are not older than **2.0.1** for CSS and **1.2.0** for UWB.

As all anchors share the majority of the commands, nanoANQ will stand for all above mentioned anchor types. Otherwise, it will be mentioned separately.

2. RF Interfaces

2.1. Overview

Each nanoANQ has two RF interfaces (channels) according to the definitions in ISO/IEC 24730-5.

The air interface is used to exchange information between different anchors and between tags and anchors. This section focuses on the tag/anchor interface, to allow to integrate the individual tag.

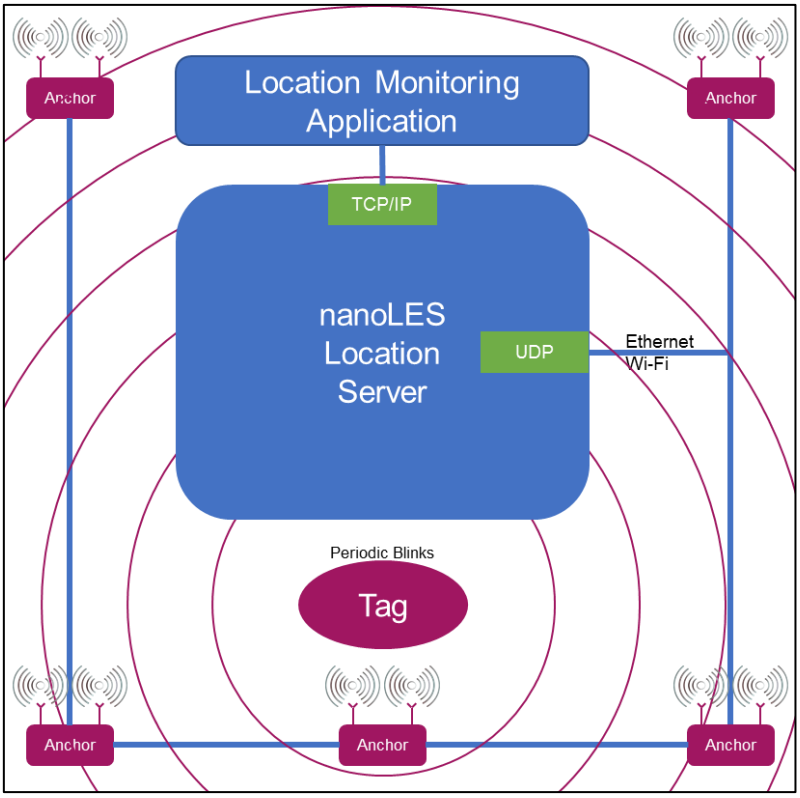


Figure 2-1 RTLS System Overview

The tags are sending broadcast messages to all anchors within the reachable range. These messages must match a certain structure to be treated correctly by the TDOA system. Each message may contain an optional user specific payload which is transparently forwarded to the application layer. The details are described in the following sections.

2.2. Message Structure

Each tag broadcast message must match the structure as shown in Figure 2-2 below.

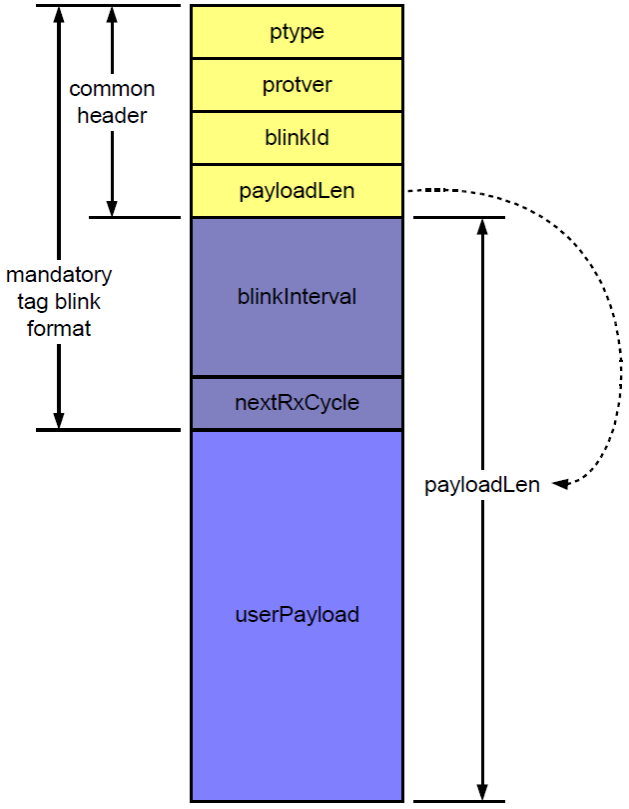


Figure 2-2 Generic Message Format

Each message used within the TDOA air interface uses the same basic structure consisting of the first fields of the message.

Filename	Size	Value	Function
ptype	1 Byte	0x02	Protocol Type (0x02 = Tag Blink)
protver	1 Byte	0x11	
blincId	1 Byte		Incrementing value, must be different for consecutive tag blinks
payloadLen	1 Byte		Number of bytes following the payloadLen field
blincinterval	3 Byte	0x000000	Interval between consecutive blinks in milliseconds. MSB first. Used for tag backchannel.
nextRxCycle	1 Byte	0xff (Note)	Number of blinks until tag is capable to receive the next tag backchannel message (feature currently not implemented)

Filename	Size	Value	Function
userPlayload	n Byte		Arbitrary payload, up to 64 Byte

Note: The current versions of the anchor firmware and the location server software do not support this feature. To simplify, the tag firmware when not using the backchannel shall set the blinkInterval to 0 and the nextRxCycle to 0xff.

An tag must fill the first 4 fields as shown in the table above.

- The fields '**ptype**' and 'protver' must use the specified values.
- The field **blinkId** contains a value that is incremented by each blink. It is used within the location server to separate consecutive blinks.
- The fields '**blinkInterval**' and '**nextRxCycle**' are specified for a tag backchannel to allow messages to be sent from the TDOA system to the tags. To save power on the tag side, the receiver shall be enabled for a short time frame only.
  The specified scheme opens the receiver after each n-th tag blink for a certain time. The TDOA system uses the two fields to schedule the next message for a certain tag (the n-th blink, n=nextRxCycle, with a spacing of blinkInterval each).
- The user **payload** may contain any data (e.g. tag sensors, pushbuttons, etc.) to be sent from the tag to the user application. The payload is forwarded transparently by the entire system.

2.3. Example

For a better understanding an example of a typical message is given in Figure **2-3** below.

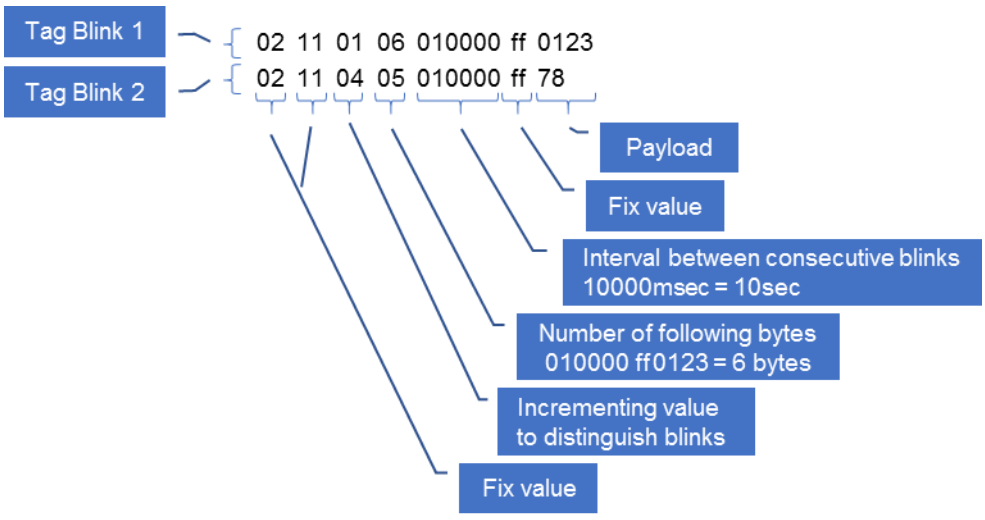


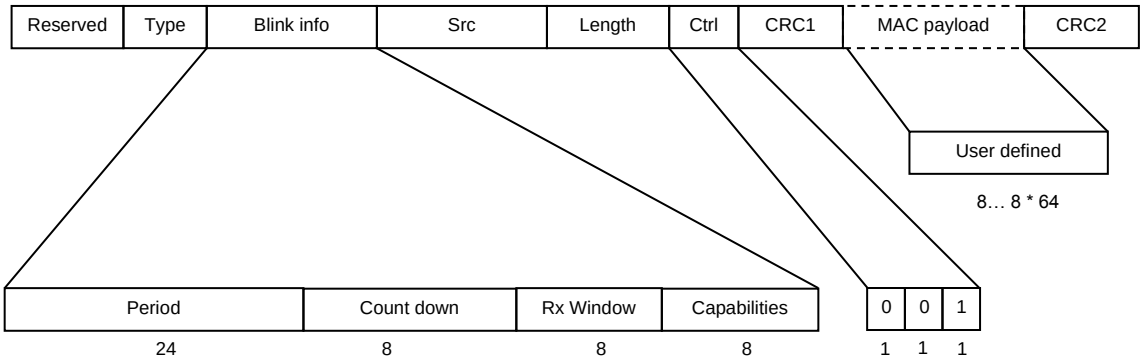
Figure 2-3 Example of Tag Message

2.4. Tag Air Interface Formats

In accordance to section 2.2, nanoANQ accepts the following tag blink formats:

2.4.1. ISO Format

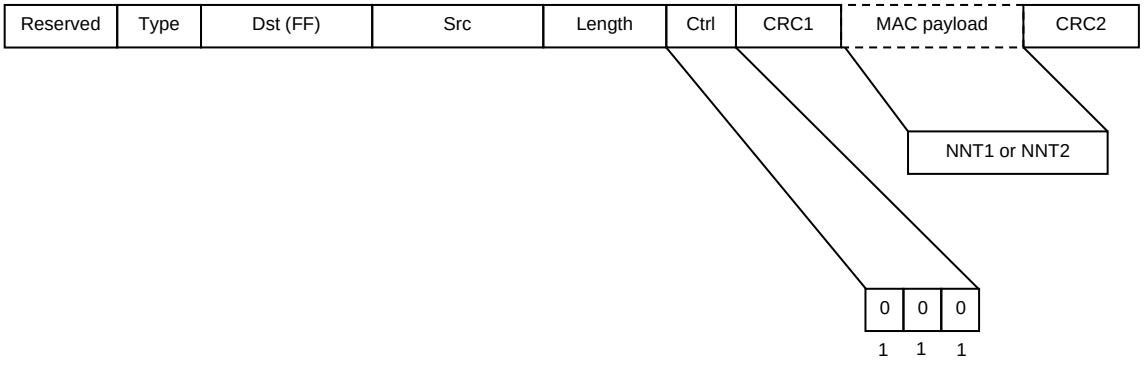
ISO/IEC 24730-5 defines an air interface based on CSS for RTLS. It contains blink packets which are broadcasted by a tag. UWB doesn't' support this format. For ISO/IEC tag blinks, the destination address field is used to transmit blink info, see the following frame structure:



Field	length	Description
Period	3 byte	Blink interval in [ms]
Count down	1 byte	Indicating the remaining blinks for the next RX slot (tag backchannel). 0 indicates that the receiver is immediately open after transmitting the blink.
RxWindow	1 byte	Width (duration) of the tag RX window in [ms].
Capabilities		Not used
Src	6 byte	Only lower 32 bits are used, upper 16 bits are ignored.
Ctrl		"001": ISO Tag blink

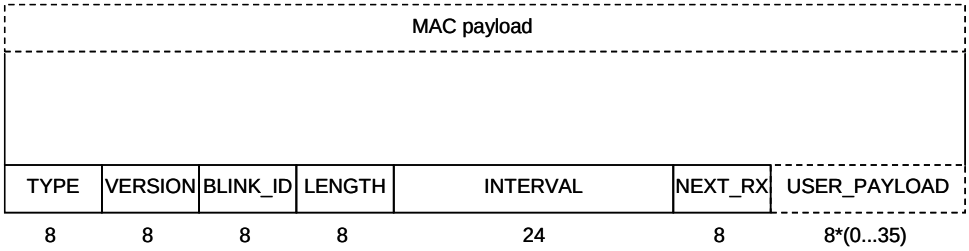
2.4.2. NNT Format

Two nanotron blink formats are defined: NNT1 and NNT2. Both formats use the broadcast blink format, where all tag blink related information is placed into the MAC payload section, as illustrated in the frame format below. The 48-bit Src field has the same restrictions as defined for ISO blinks: Only the lower 32 bits are used.



2.4.3. NNT1 Format

(aka Version 2.12)

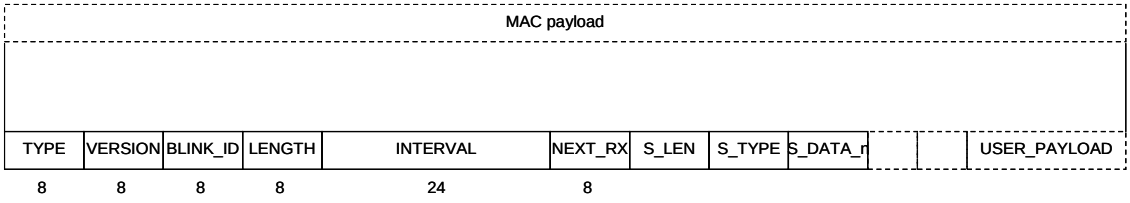


Field	length	Description
TYPE	1 byte	0x02 indicating a tag blink
VERSION	1 byte	0x11 (obsolete) or 0x12 (new), indicating NNT1
BLINK_ID	1 byte	Shall be different for subsequent packets. Recommended to use incremental or decremental counter 0..255 (255 ..0)
LENGTH	1 byte	Number of octets following after the LENGTH field
INTERVAL	3 byte	Tag blink interval in [ms]
NEXT_RX	1 byte	Indicating the remaining blinks for the next RX slot (tag backchannel). 0 indicates that the receiver is immediately open after transmitting the blink.

Note: No support for UWB

2.4.4. NNT2 Format

This format supports CSS as well as UWB. This is the recommended one.



Field	length	Description
TYPE	1 byte	0x02 indicating a tag blink
VERSION	1 byte	0x13 indicating NNT2 (2.13), 0x20 assetTAG
BLINK_ID	1 byte	Shall be different for subsequent packets. Recommended to use incremental or decremental counter 0..255 (255 ..0)
LENGTH	1 byte	Number of octets following after the LENGTH field
INTERVAL	3 bytes	Tag blink interval in [ms]
NEXT_RX	1 byte	Indicating the remaining blinks for the next RX slot (tag backchannel). 0 indicates that the receiver is immediately open after transmitting the blink.
S_LEN		Specifies the length of all S_TYPE and S_DATA together. At least one byte
S_TYPE		0x00: NO SENSOR (ESC) 0x01: Battery 0x02: reserved .. 0xFF: reserved
S_DATA_n		Sensor data according to the structure defined via S_TYPE for S_TYPE 0x00: no data (ESC) for S_TYPE 0x01: 2 Bytes uint16_t, 100mV

3. Communication Interfaces

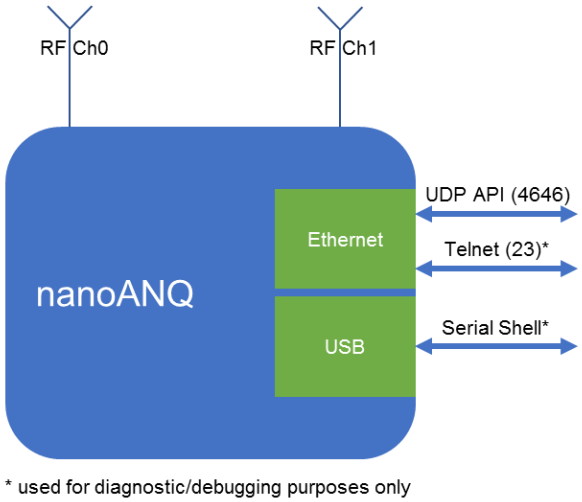


Figure 3-1 Communication Interfaces

The nanoANQ is accessible via three programming interfaces

- UDP API via Ethernet (UDP port 4646)
  This is the only interface to be used for anchor management and TOA measurement data access

- TCP Telnet via Ethernet (TCP port 23), Serial Shell via USB
  These interfaces provide low-level management functions and are intended for diagnostic and debugging purposes only
  The TCP and USB commands which are accessible by the user are given in section 7

Software Interfaces

3.1. UDP API

General UDP Message Format

Field	length	Description
UDP_SOURCE_PORT	2 Bytes	16-bit port number of the UDP message's origin
UDP_DESTINATION_PORT	2 Bytes	16-bit port number of the UDP message's destination
UDP_LENGTH	2 Bytes	length of the UDP datagram, including both header and data fields
UDP_CHECKSUM	2 Bytes	UDP checksum
UDP_DATA	variable	The encapsulated higher-layer message: - TDOA Blink - Anchor Management Message

3.1.1. TDOA Blink Format

This defines the UDP blink format (UDP_DATA) sent by anchors to the server when they received tag or synchronization blinks via the air interface. By default, UDP packets are send to UDP port 5555.

Field	Parameter	value	Description
UDP_TYPE	uint16_t type	0x0101	UDP_TDOA_BLINK
UDP_VERSION	uint16_t version	0x0002 0x0003	UDP_TDOA_BLINK_VERSION UDP_TDOA_BLINK_VERSION2
UDP_BLINK_RAW_DATA	uint8_t raw_data		raw blink data
UDP_BLINK_PAYLOAD	uint8_t blink_payload		blink payload

4. Command overview

The Management Interface is used to change settings and the behavior of anchors. The following sections describe the protocol and its types of packets. By default, the anchor is listening to port 4646 for incoming UDP packets.

NEMONIC	OPCODE_CMD	RW	Short description
AGC	0x00	RW	Enables/Disables the automatic gain control.
ANCHORID	0x01	R	Reads the anchor ID.
BANDWIDTH	0x02	RW	Selects transmission modes such as 80/1, 80/4.
ETHMAC	0x03	R	Reads the Ethernet MAC address.
FLASHLOAD	0x04	W	Reboots anchor into the bootloader. The bootloader will wait for a new FW update.
FWREV	0x06	R	Reads the firmware version.
HWREV	0x07	R	Reads the hardware version.
RESET	0x09	W	Restarts the anchor.
RXENABLE	0x0A	RW	Enables/Disables the receiver of both modules individually.
SERVER	0x0B	RW	Configures the server and port to which the anchor will send the raw data UDP packets.
SYNCWORD	0x0D	RW	Selects a syncword (used to separate networks)
TXENABLE	0x0F	RW	Enables/Disables the transmission of pacer blinks from both modules individually.
TXPWR	0x10	RW	Sets/Gets the output power register value for both modules at once.
EEDUMP	0x12	R	Reads the configuration from EEPROM.
INTERVAL	0x14	RW	Sets/Gets the pacer blink interval in ms.
CSMATO	0x16	RW	Sets/Gets the CSMA timeout in ms.
CSMACFG	0x29	RW	Sets/Gets the CSMA configuration.
GPIO	0x17	RW	Sets/Gets GPIOs configuration. This includes blink patterns.
LEDMASK	0x18	RW	Allows to mask LEDs and prevents/enables them to light up.
FEC	0x1a	RW	Enables/Disables forward error correction.
TXPWR_EXT	0x1b	RW	Sets/Gets the output power register value for both modules individually.
TEST	0x2c	W	Production tests.
DEFAULT	0x2b	W	Resets the device to factory default settings.
UID	0x2e	R	Reads the unique id.
IFCONFIG	0x2d	RW	Ethernet configuration. Settings are applied after reset.

5. Protocol description

5.1. Basics

TYPE (2 Bytes)	VERSION (2 Bytes)	OPCODE_TYPE (2 Bit)	OPCODE_CMD (14 Bit)	LEN (1 Byte)

5.1.1. Type

5.1.2. Version

- 0x0001 (Version 1.0)

5.1.3. OPCODE_TYPE

- GET (0)
- SET (1)
- RESP (2)
- UNKNOWN (3)

5.1.4. OPCODE_CMD

This are the command opcodes from the table in section 4.

5.1.5. LEN

The length depends on the OPCODE and is described for each individual command. It is related to the field “DATA” which follows.

5.2. Commands

Every set and get command is acknowledged by sending back a response type packet. The response packet holds information related to the originating command (similar to the SET request). If the opcode is unknown, it will send back the received packet with the opcode unknown response. A get command usually has 0 byte in length. If there is an exception, it will be mentioned in the GET request section of the command description.

5.2.1. Unknown OPCODE

In case the opcode used is unknown to the anchor, it will echo the received packet back to the sender and set the OPCODE_TYPE to UNKNOWN (3).

Example for an unknown opcode 0x3FF as a set command:

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (N Byte)
0x01050001	0x43FF	0x03	0x010203

This packet will not be understood by the anchor since the OPCODE_CMD 0x3FF does not exist. Therefore, the following response will be generated:

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**C**3FF	0x01	identical to request

5.2.2. AGC

Enables/Disables the automatic gain control. AGC must be enabled to determine RSSI values and proper CSMA with signal detection operation.

Range: 0 … 1

- 0 = Disable
- 1 = Enable

Default: 1 (Enabled)
Recommended: 1 (Enabled)

SET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x4000	0x01	0x00 … 0x01

SET response

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**8**000	0x01	identical to request

The OPCODE_TYPE changes to response (RESP = 2)

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x0000	0x00

GET response

In this example the AGC is enabled (1) and the anchor returns 0x01.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**8**000	0x01	0x01

The OPCODE_TYPE changes to response (RESP = 2).

5.2.3. ANCHORID

Reads the anchor id. The anchor id is generated from the Ethernet mac address, which is set during production. It cannot be changed. The anchor id uses the lower 3 bytes of the mac address as its ID.

Range: 0x00000000 … 0x00FFFFFF

Default: The anchor gets its ID from lower 3 bytes of the Ethernet mac address.
Recommended: -

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x0001	0x00

No parameters necessary.

GET response

The example anchor has a mac address 0x180b52010203 which results in an anchor id 0x00010203.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (4 Bytes)
0x01050001	0x**8**001	0x04	0x00010203

The OPCODE_TYPE changes to response (RESP = 2).

5.2.4. BANDWIDTH

Configures the chirp transmission mode of the transceiver.

Range:

- 0 = 80/1
- 1 = not used, legacy
- 2 = 80/4

Note: The first number stands for bandwidth which is 80MHz. The second number is the symbol duration in µs. 80/1 80MHz stands for a bandwidth and 1µs symbol duration. A symbol is a digital 1 or 0.

Default: 0 (80/1)
Recommended: For high throughput '0' (80/1) for max. operational distance '2' (80/4).

SET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x4002	0x01	0x00 … 0x02

SET response

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**8**002	0x01	identical to request

The OPCODE_TYPE changes to response (RESP = 2).

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x0002	0x00

GET response

The anchor of this example has 80/4 configured as bandwidth and returns 0x02.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**8**002	0x01	0x02

The OPCODE_TYPE changes to response (RESP = 2).

5.2.5. ETHMAC

Reads the Ethernet mac address. The Ethernet mac address is a 6 bytes value starting with the Nanotron identifier followed by a 3 bytes unique number.

Range: 0x180b52000001 … 0x180b52ffffff

Default: -
Recommended: -

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x0003	0x00

GET response

The anchor of this example has the mac address 0x180b52010203.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (6 Byte)
0x01050001	0x**8**003	0x06	0x180b52010203

The OPCODE_TYPE changes to response (RESP = 2).

5.2.6. FLASHLOAD

Reboots the device into the bootloader. The bootloader will expect a new firmware and therefore will wait until a new FW is transmitted via TFTP or 10min are elapsed without a FW update. If the current image is unchanged (valid), the current runtime image is booted again.

Range: -
Default: -
Recommended: -

SET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (0 Byte)
0x01050001	0x4004	0x00	-

SET response

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (0 Byte)
0x01050001	0x**8**004	0x00	-

The OPCODE_TYPE changes to response (RESP = 2).

5.2.7. FWREV

Reads the firmware version. The firmware version is a 4 Bytes value which is decoded as follows:

0xAABBCCDE

AA = Major
BB = Minor
CC = Sub
D = Release candidates (valid release candidates are 0x0 … 0xE, the value 0xF is the final release version)
E = Changes since last tag

Range: -
Default: -
Recommended: -

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x0006	0x00

GET response

The anchor of this example has the FW version ver3.0.10-rc3.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (4 Bytes)
0x01050001	0x**8**006	0x04	0x03000A30

The OPCODE_TYPE changes to response (RESP = 2).

5.2.8. HWREV

Reads the hardware. The hardware version is fixed for each individual product and product iteration. The firmware checks this version during boot and initializes the hardware accordingly. The hardware version is a 2 Bytes value and is decoded as follows:

0xAABB

AA = Major
BB = Minor

Hardware	HW Rev
nanoANQ EA v3 Chirp (Ethernet)	9.0
nanoANQ EA v3 Chrip (WIFI)	10.0

Range: -
Default: -
Recommended: -

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x0007	0x00

GET response

The anchor of this example has the HW version 9.0 (nanoANQ-c / pcbANQ-c (Ethernet))

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (2 Bytes)
0x01050001	0x**8**007	0x04	0x0900

The OPCODE_TYPE changes to response (RESP = 2).

5.2.9. RESET

Reboots the anchor.

Note: In the case a device is connected to the USB port, the anchor will wait for 10 seconds before booting the runtime image.

Range: -
Default: -
Recommended: -

SET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (0 Byte)
0x01050001	0x4009	0x00	-

SET response

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (0 Byte)
0x01050001	0x**8**009	0x00	-

The OPCODE_TYPE changes to response (RESP = 2).

5.2.10. RXENABLE

Enables/Disables the receivers of the anchor.
Range: 0 … 3

- 0 = Disable receivers
- 1 = Enable receiver channel 0, disable receiver channel 1
- 2 = Enable receiver channel 1, disable receiver channel 0
- 3 = Enable both receivers

Default: 3 (Both receivers enabled)
Recommended: 3 (Both receivers enabled)

SET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x400A	0x01	0x00 … 0x03

SET response

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**8**00A	0x01	identical to request

The OPCODE_TYPE changes to response (RESP = 2).

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x000A	0x00

GET response

In this example both receivers are enabled (3).

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**8**00A	0x01	0x03

The OPCODE_TYPE changes to response (RESP = 2).

5.2.11. SERVER

Configures the nanoLES address and port to which the UDP packets have to be sent to. The anchor will send the raw data of blinks received via the air interface to this address. The nanoLES will use this raw data to calculate a position.

Range: [IP address (4 Bytes)] [Port (2 Bytes)]

- IP address (4 Bytes)
- Port (2 Bytes)

Default: 0.0.0.0:0 (No server configured)
Recommended: The nanoLES server address and port

SET request

This example sets the IP 192.168.202.72 and port 5555 as destination.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (6 Byte)
0x01050001	0x400B	0x06	0xC0A8CA4815b3

SET response

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**8**00B	0x06	identical to request

The OPCODE_TYPE changes to response (RESP = 2).

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x000B	0x00

GET response

In this example the anchor has following IP and port configured: 192.168.202.72:5555.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (6 Byte)
0x01050001	0x**8**00B	0x06	0xC0A8CA4815b3

The OPCODE_TYPE changes to response (RESP = 2).

5.2.12. SYNCWORD

Selects a syncword. A syncword is used to differentiate multiple CSS (**C**hrip **S**pread **S**pectrum) based systems on the air interface which are in the operational range. It allows to create individual networks. Only devices with the same syncword will process the associated packets. Otherwise, they are ignored. This permits the receivers to drop the packets in an early stage of the reception, allowing to re-enable the receiver faster.

Range: 0 … 12
Default: 1
Recommended: any (All devices must use the same syncword to interact with each other)

SET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x400C	0x01	0x00 … 0x0C

SET response

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**8**00C	0x01	identical to request

The OPCODE_TYPE changes to response (RESP = 2).

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x000C	0x00

GET response

In this example the syncword is 3.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**8**00C	0x01	0x03

The OPCODE_TYPE changes to response (RESP = 2).

5.2.13. TXENABLE

Enables/Disables the sending of blinks with the specified interval (see section 5.2.16). These blinks must be sent to allow nanoLES to get knowledge about the clock. This process is integral part of the TDOA location system.

Range: 0 … 3

- 0 = Disable blinking
- 1 = Enable blinking of channel 0, disable channel 1
- 2 = Enable blinking of channel 1, disable channel 0
- 3 = Enable blinking on both channels

Default: 3 (Blinking on both channels enabled)
Recommended: 3 (Blinking on both channels enabled)

SET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x400F	0x01	0x00 … 0x03

SET response

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**8**00F	0x01	identical to request

The OPCODE_TYPE changes to response (RESP = 2).

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x000F	0x00

GET response

In this example both channels are configured to transmit blinks (3).

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**8**00F	0x01	0x03

The OPCODE_TYPE changes to response (RESP = 2).

5.2.14. TXPWR

[DEPRECATED] Do not use this command for future developments. Use TXPWR_EXT instead.

Sets the output power.

Range: 0 … 63
Default: 63
Recommended: 63

SET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x4010	0x01	0x00 … 0x3F

SET response

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**8**010	0x01	identical to request

The OPCODE_TYPE changes to response (RESP = 2).

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x0010	0x00

GET response

In this example the configured output power is 63.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**8**010	0x01	0x3F

The OPCODE_TYPE changes to response (RESP = 2).

5.2.15. EEDUMP

Reads the content of the EEPROM. This gives useful information that can be to added to an error report. It is a snapshot of all anchor settings.

The get command has parameters: [start_addr (1 Byte)] [length (1 Byte)]

Range:

- start_addr: 0 … 255
- length: 0 … 255

Default: -
Recommended: -

GET request

In this example the first 5 bytes of the eeprom are read out.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (2 Bytes)
0x01050001	0x0012	0x02	0x0005

GET response

The first 2 Bytes in the response are still the start_addr and length. The eeprom data following the length field: [start_addr (1 Byte)] [length (1 Byte)] [ee_data (n Bytes)]

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (7 Byte)
0x01050001	0x**8**012	0x07	0x0005AABBCCDDEE

The OPCODE_TYPE changes to response (RESP = 2).

5.2.16. INTERVAL

Configures the blink interval of the anchor in milliseconds. The channel from which the packet is transmitted alternates between both channels. This process is integral part of the TDOA location system.

Range: 0 … 1000

- 0 = Disable blinking
- 1 … 1000 = Interval in milliseconds

Default: 250ms
Recommended: 250ms

SET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (2 Bytes)
0x01050001	0x4014	0x02	0x0000 … 0x03E8

SET response

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (2 Bytes)
0x01050001	0x**8**014	0x02	identical to request

The OPCODE_TYPE changes to response (RESP = 2).

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x0014	0x00

GET response

In this example the interval is 250ms.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (2 Bytes)
0x01050001	0x**8**014	0x02	0x00FA

The OPCODE_TYPE changes to response (RESP = 2).

5.2.17. CSMATO

Configures the CSMA timeout. CSMA will block the transmission when another device is using the medium until it is free again or this timeout expires. If the timeout expires the packet is dropped.

Note 1: If the device has TX enabled, an interval greater than 0 and the it could not transmit for consecutive 10 seconds, the anchor will reboot in order to attempt to fix an unknown error.

Note 2: CSMA based on signal strength only works if the AGC is enabled.

Range: 0 … 65535ms

- 0 = Disable CSMA timeout (The device will try to send forever until the medium is free).
- 1 … 65535ms = Timeout: After this time CSMA stops and the packet transmission is canceled.

Default: 50ms
Recommended: 50ms

SET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (2 Bytes)
0x01050001	0x4016	0x02	0x0000 … 0xFFFF

SET response

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (2 Bytes)
0x01050001	0x**8**016	0x02	identical to request

The OPCODE_TYPE changes to response (RESP = 2).

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x0016	0x00

GET response

In this example the timeout is 50ms.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (2 Bytes)
0x01050001	0x**8**016	0x02	0x0032

The OPCODE_TYPE changes to response (RESP = 2).

5.2.18. CSMACFG

Configures the CSMA mode. Configures the CSMA detection method. This means under which conditions the medium is detected as free. CSMA will block the transmission when another device is using the medium until it is free again or this timeout expires. If the timeout expires the packet is dropped.

Note 1: If the device has TX enabled, an interval greater than 0 and it could not transmit for consecutive 10 seconds, the anchor will reboot in order to attempt to fix an unknown error.

Note 2: CSMA based on signal strength only works if the AGC is enabled.

Range: [mode (1 Byte)] [seed (1 Byte)] [threshold (1 Byte)]

- Mode
    - 0 = CSMA disabled
    - 1 = Symbol detection - fixed
    - 2 = Symbol detection - random
    - 3 = Signal strength detection - fixed
    - 4 = Signal strength detection – random
- Seed
    - 0 .. 255 =
        - in mode 1/3 (fixed) it is the (seed+1) * 24µs slots to be detected as free (total since start of tx) before TX.
        - in mode 2/4 (random) seed is ignored and a random number of 24µs slots is used.
- Threshold
    - 0 .. 63 =
        - in mode 3/4 (signal strength) it is the threshold to decide if the medium is blocked.
        - in mode 1/2 (Symbol) is ignored.

Default after execution of DEFAULT command: Not affected. Unchanged.

Default after production test: 4,0,20 => Signal detection, threshold 20 for nanoANQ-c and pcbANQ-c.

Default after production test: 1,0,0 => Symbol detection with 1x 24µs detection slot for all other anchors.

SET request

In this example the mode 4, 0, 20 is configured.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (3 Byte)
0x01050001	0x4029	0x03	0x040014

SET response

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (3 Byte)
0x01050001	0x**8**029	0x03	identical to request

The OPCODE_TYPE changes to response (RESP = 2).

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x0029	0x00

GET response

In this example the mode is 4, 0, 20.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (3 Byte)
0x01050001	0x**8**029	0x03	0x040014

The OPCODE_TYPE changes to response (RESP = 2).

5.2.19. GPIO

Configures the LED#6 of the nanoANQ-c or pcbANQ-c. The purpose of this is determined by the application. This command allows to configure the LED#6 to blink in a pattern.

Protocol:

PORT (1 Byte)	MASK (1 Byte)	T_ON (2 Bytes)	T_OFF (2 Bytes)	REPS (2 Bytes)	STATE (1 Byte)

Range:

- PORT = 0
  - 0 = LED Array
  - 1 = RGB LED (not supported in nanoANQ-c or pcbANQ-c)
- MASK
  - Bit0 = LED#6
  - Bit1 … 7 = Reserved.
- T_ON
  - 0 … 65635 = pin active time in milliseconds
- T_OFF
  - 0 … 65635 = pin inactive time in milliseconds
- REPS
  - 0 … 65634 = repetitions
  - 65635 = repeat forever
- STATE
  - 0 = Starting repetitions with T_OFF
  - 1 = Starting repetitions with T_ON

Default: -
Recommended: -

SET request

In this example the LED#6 is blinking forever every 200ms with 50ms active time and 150ms inactive time.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (3 Byte)
0x01050001	0x4029	0x03	0x040014

SET response

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (3 Byte)
0x01050001	0x**8**029	0x03	identical to request

The OPCODE_TYPE changes to response (RESP = 2).

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x0029	0x00

GET response

In this example the mode is 4, 0, 20.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (3 Byte)
0x01050001	0x**8**029	0x03	0x040014

The OPCODE_TYPE changes to response (RESP = 2).

5.2.20. LEDMASK

Allows to mask all LEDs except the power LED and the LED Array configured through GPIO command.

Range: 0x00 … 0xFF

- Bit0 = rx channel 0 (blue)
- Bit1 = tx channel 0/1 (red)
- Bit2 = alive (green)
- Bit3 = rx channel 1 (blue)
- Bit4 = fault (red)

Default: 0xFF
Recommended: 0xFF (all LEDs enabled)

SET request

In this example the alive and fault LEDs are set to be enabled, while all other shall be muted.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x4018	0x01	0x14

SET response

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**8**018	0x01	identical to request

The OPCODE_TYPE changes to response (RESP = 2).

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x0018	0x00

GET response

In this example the mask is set to 0x1F.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**8**018	0x01	0x1F

The OPCODE_TYPE changes to response (RESP = 2).

5.2.21. FEC

Enables/Disables the FEC (**F**orward **E**rror **C**orrection). This improves the sensitivity and thus the max. operational communication range. The downside is the increased TX duration. In order to communicate amongst the involved devices, they must have the same setting. Either all involved devices as tags and anchors must have FEC enabled or disabled.

Range: 0 … 1

- 0 = Disable
- 1 = Enable

Default: 0 (Disabled)
Recommended: 0 (Disabled)

SET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x401A	0x01	0x00 … 0x01

SET response

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**8**01A	0x01	identical to request

The OPCODE_TYPE changes to response (RESP = 2).

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x001A	0x00

GET response

In this example the FEC is disabled (0) and the anchor returns 0x00.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**8**01A	0x01	0x00

The OPCODE_TYPE changes to response (RESP = 2).

5.2.22. TXPWR_EXT

Sets the TX output power.

Range: [txpwr channel 0 (1 Byte)] [txpwr channel 1 (1 Byte)]

- txpwr channel 0 = 0 … 63
- txpwr channel 1 = 0 … 63

Default: 63 (max. for both channels)
Recommended: 63 (max. for both channels)

SET request

In this example the TX output power of channel 0 is set to 63 and the output power of channel 1 is set to 50.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (2 Bytes)
0x01050001	0x401B	0x02	0x3F32

SET response

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (2 Bytes)
0x01050001	0x**8**01B	0x02	identical to request

The OPCODE_TYPE changes to response (RESP = 2).

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x001B	0x00

GET response

In this example the configured output power for both channels is 63.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (2 Bytes)
0x01050001	0x**8**01B	0x02	0x3F3F

The OPCODE_TYPE changes to response (RESP = 2).

5.2.23. DEFAULT

The default command resets all settings back to factory settings. See section6 for an overview of all commands and their default settings. To complete the process, a reset of the device must be executed.

Note: The IP address is reset to 192.168.202.79 and the DHCP is disabled. Before resetting the anchor, it is possible to reconfigure DHCP to enabled or to set the anchors IP address again. See section 5.2.25. The settings are effective after reboot.

Range: -
Default: -
Recommended: -

SET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (0 Byte)
0x01050001	0x402b	0x00	-

SET response

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (0 Byte)
0x01050001	0x**8**02b	0x00	-

The OPCODE_TYPE changes to response (RESP = 2).

5.2.24. UID

Reads the 96 bit unique ID of the STM32.

Range: -
Default: -
Recommended: -

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x002E	0x00

GET response

The 96 Bit unique ID is returned.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (12 Bytes)
0x01050001	0x**8**02E	0x02	…

The OPCODE_TYPE changes to response (RESP = 2).

5.2.25. IFCONFIG

Set the IP, netmask and gateway of the Ethernet interface. This setting is applied after resetting the anchor. The command only support IPv4 addresses.

Range: [mode (1 Byte)] [IP (4 Bytes)] [netmask (4 Bytes)] [gateway (4 Bytes)]

- mode = Fixed (0) or DHCP (1)
- IP
- netmask
- gateway

Default after execution of DEFAULT command:

- mode = Fixed (0)
- IP = 192.168.202.79
- Netmask = 255.255.255.0
- Gateway = 0.0.0.0

Default after production test:

- mode = DHCP (1)

Recommended:

- mode = DHCP (1)

SET request

In this example the anchor gets configured to be in Fixed (0) mode and IP 192.168.16.1, netmask 255.255.255.0, and gateway 192.168.16.3.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (13 Byte)
0x01050001	0x402E	0x0D	0x00C0A81001FFFFFF00C0A81003

SET response

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (1 Byte)
0x01050001	0x**8**02E	0x01	identical to request

The OPCODE_TYPE changes to response (RESP = 2).

GET request

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)
0x01050001	0x002E	0x00

GET response

In this example the anchor is configured in Fixed (0) mode and IP 192.168.16.1, netmask 255.255.255.0, and gateway 192.168.16.3.

TYPE + VERSION (4 Bytes)	OPCODE_TYPE + CMD (2 Bytes)	LEN (1 Byte)	DATA (13 Byte)
0x01050001	0x**8**02E	0x01	0x00C0A81001FFFFFF00C0A81003

6. Default Settings

There are 2 states an anchor can take.

a) After production test
b) After execution of DEFAULT command.

The settings are very similar. The following table shows the state of a) and highlights the differences in regard to b) in the column “Short description”. The column “Affected by DEFAULT CMD” states if this setting gets modified by the DEFAULT command. If it is not modified, it will be the same as it was before executing the DEFAULT command.

NEMONIC	OP	Affected by DEFAULT CMD	Short description
AGC	0x00	Yes	Enabled (1)
BANDWIDTH	0x02	Yes	80/1
RXENABLE	0x0A	Yes	Both channels enabled (3)
SERVER	0x0B	Yes	0.0.0.0:0
SYNCWORD	0x0D	Yes	1
TXENABLE	0x0F	Yes	Both channels enabled (3)
TXPWR	0x10	Yes	0x63 (max)
INTERVAL	0x14	Yes	250ms
CSMATO	0x16	Yes	20ms
CSMACFG	0x29	No	a) nanoANQ-c 4,0,20 (signal detection, threshold 20) b) All others 1,0,0 (symbol detection)
LEDMASK	0x18	Yes	0xFF (all LEDs allowed to blink - not masked)
FEC	0x1a	Yes	FEC disabled (0)
TXPWR_EXT	0x1b	Yes	Both channels to 0x3F (max) output power.
IFCONFIG	0x2d	Yes	a) DHCP enabled b) IP=192.168.202.79; MASK=255.255.255.0

7. Low Level Access

TCP Telnet via Ethernet (TCP port 23), Serial Shell via USB.
These commands shall be used if static IP addresses need to be set or reset from a fixed IP address to DHCP.

Changing IP Configuration

Command	Parameter	Description
default		restores default settings (DHCP=1), all other parameters to default values. Requires reset.
ifconfig	“dhcp” <ip> <ip> <mask> <ip> <mask> <gw>	dhcp: turns on DHCP based configuration <ip>: sets IP address manually <ip> <mask>: sets IP addr. and network mask manually <ip> <mask> <gw>: sets IP, mask, gateway manually
reset		reset the node. Required after each *bandwidth* and *ifconfig* command
server	<ip addr>:<port>	destination for live UDP blink packets default: 0.0.0.0:5555 Must be renewed after each ‘default’ command

8. References

[1] nanoANQ EM (Order No. MNACORUFL / MNACOR01),
Data Sheet Doc. Id. NA-13-0276-0028
[2] nanoANQ EA (Order No. BNAR02PYEA)
Data Sheet Doc. Id. NA-19-0352-0010
[3] nanoANQ-c (Order No. PN03ANQCS)
pcbANQ-c (Order No. BN03ANQCS)
Data Sheet Doc. Id. NA-20-0387-0009
[4] nanoANQ EM ER (Order No. MN01ANQEMER / BN01ANQEMPXER),
Data Sheet Doc. Id. NA-18-0371-0008
[5] eLibrary: https://nanotron.com/elibrary/

Document History

Date	Authors	Version	Description
15.11.2013	NNT	1.0	• 1st release for firmware versions 2.0.7 and above.
14.03.2014	NNT	1.1	• Added TX Power Curve
05.08.2014	NNT	1.2	• Common document for nanoANQ-EM and nanoANQ-V2 • Added Ranging API
19-02-2019	MBOR	1.3	• Added new commands and responses belonging to nanoANQ EM ER • Sect. 2.3 removed as buffering has become obsolete • The “Anchor Management Opcodes” table has been refurbished. The UWB management commands have been added. A technology column has been amended
11-12-2019	MBOR	1.4	• Added coding for the different channels in MGMNTIF_CMD_OPCODE_LOCHANNEL
04-11-2020	MBOR	1.5	• New version with explanations of the commands for a better comprehension of what they are doing and how to configure them

Life Support Policy

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. nanotron Technologies GmbH customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify nanotron Technologies GmbH for any damages resulting from such improper use or sale.

Sales Inquiries

nanotron Technologies GmbH
Alt-Moabit 60a
10555 Berlin, Germany

Europe/Asia/Africa: +49 (30) 399954-0
USA/Americas/Pacific: +1 (339) 999-2994
Mail: nanotronsales@inpixon.com
Web: www.nanotron.com, www.inpixon.com

About nanotron, An Inpixon Company

Nanotron Technologies GmbH, an Inpixon company (Nasdaq: INPX) is a leading provider of electronic location awareness solutions. If knowing what, where and when is mission-critical to your business, rely on nanotron with Location Running.

Nanotron's solutions deliver precise position data augmented by context information in real-time. Location Running means, reliably offering improved safety and increased productivity, 24 hours a day, 7 days per week: Location-Awareness for the Internet of Things (IoT).