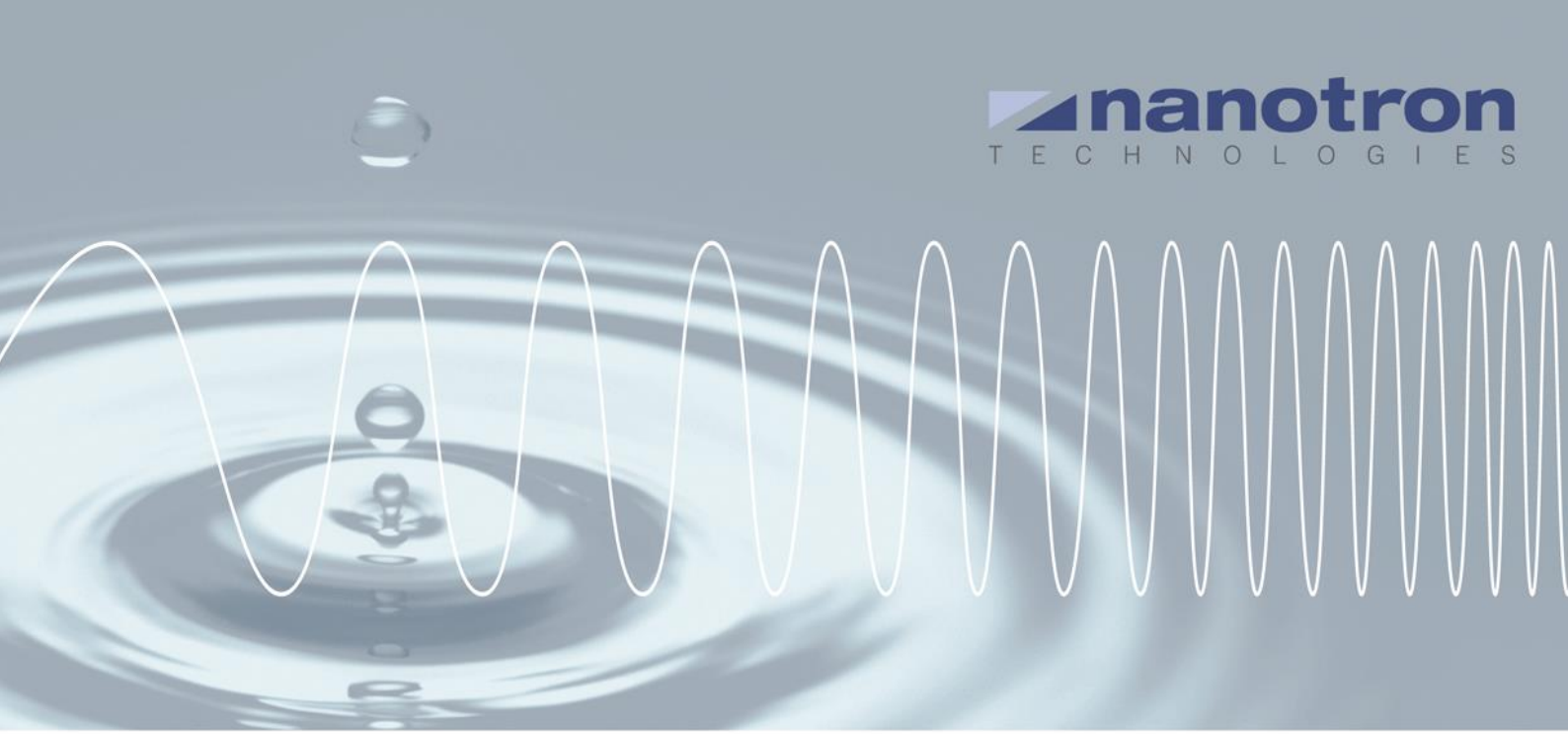

swarm API

2.03

NA-13-0267-0003-2.03

Chirp it.

Document Information

Document Title:	*swarm* API
Document Version:	2.03
Current Date:	2014-11-20
Print Date:	2014-11-20
Document ID:	NA-13-0267-0003-2.03
Document Author:	nanotron

Contents

1. Scope

The Scope of this document is to define a hardware independent Application Programming Interface (API) to realize the ranging functionality for a *swarm* radio described in chapter 2. A swarm is defined as a congregation of independent radios or nodes which share a common interest in their relative positioning and can communicate together, and detect newcomers and departures from the local swarm area.

The main goal of the API is to support and simplify the development of ranging applications and allow swarm mobility. For these purposes, it provides a list of commands which act as interface with the hardware and can be used to build more complex applications in an easy way.

From API 2.0 onward power management for mobile devices is supported. The API 2.0.x is not backward compatible with previous releases (V1.6.X).

2. Functionality Overview

The intended application is a multi-node peer-to-peer ranging solution in which every node shall be independent from each other, capable of ranging and communicating with other nodes and battery operated to allow fully mobile applications. For this purpose the *swarm* radio shall be able to perform two tasks:

1. Send out ID broadcast information. Swarm radio nodes shall periodically broadcast their own node ID together with their node information data, so that other swarm radio nodes can be aware of the presence of the first one and can decide to interact with it. This feature can be deactivated.

2. Listening to other broadcast IDs and react to them. When a swarm radio receives a broadcast ID packet, it shall store the node's ID (and if relevant the information data received) and initiate a ranging operation with the swarm radio owning this broadcast ID.

A ranging operation consists of 3 packets. The first packet is a ranging request sent by the node initiating the operation; the other two packets are the automatic response of the node receiving the request. The automatic ranging response for packets can be also deactivated by "privacy mode".

The received sensor data consist of sensors data (temperature acceleration), battery level, node class and node power management mode. The node class is set by the user and it is used to classify the nodes and configure their automatic ranging feature. All this node information data is stored in the receiving swarm node and can be accessed via the APIs well as the battery status of the node itself.

The node class is an application dependent parameter; it is used to define different groups of swarms. They can be classified according to their power mode, the kind of device they are placed on, how fast they move... The user is free to use its own criteria and whatever better fits with the application. Each node can then be given a list of classes so that whenever it receives a node-id blink, it checks whether that class is in the list. If yes, it immediately starts ranging with that node.

To implement the described functionalities the *swarm* radio shall either be connected to a host platform or work as an autonomous node. The node is controlled through the USART interface connected to a host platform. This interface is described in chapter 4. Furthermore, the swarm radio nodes that are controlled by the host should be capable of communicating data packets of variable length to another *swarm* radio B, which can be host-controlled or autonomous.

The higher level application layer is not part of the API specification and must be programmed by the user.

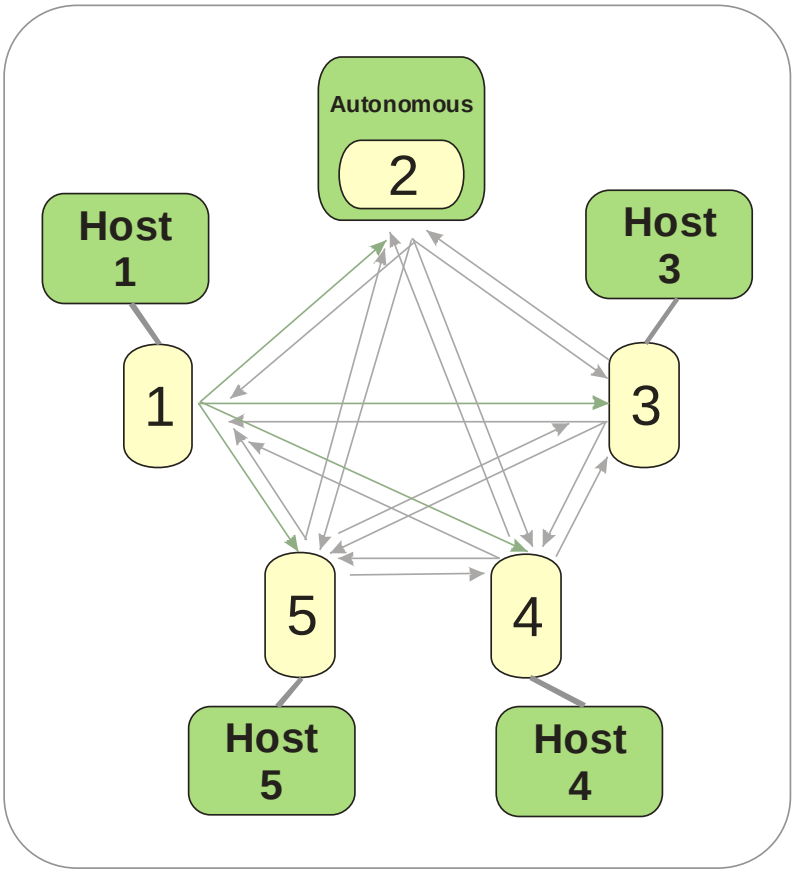

Fig. 1 Example of *swarm* bee nodes interacting with each other. Nodes 1,3,4,5 are host-controlled and node 2 works autonomously

3. Locating Methods

Swarm radios can implement location by using either collaborative or fixed location methods. The selection of one or the other is influenced by the application requirements.

When working in a collaborative mode the swarms are aware of the presence of other swarms in their neighborhood and are capable of estimating their range or 1-D position relative to its own position. They can then inform other nodes in the area and send them the estimated positions (as well as the received ones), so that all swarms can use that information to make an estimation of the position of other swarms that are not in range.

Collaborative Location

Collaborative location uses relative positions to provide location-awareness. Radio nodes determine the distance to neighbors by exchanging packets and measuring their time of flight (TOF) at the speed of light. This method is called ranging. Radios are autonomous, location infrastructure is not required.

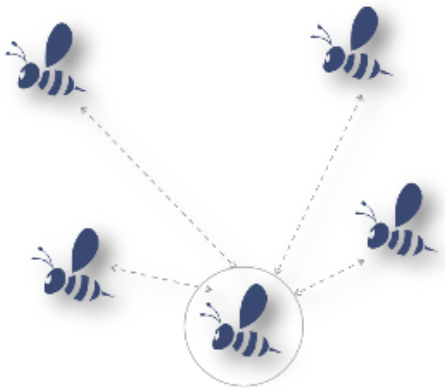

Fig. 2 Collaborative location for mobile nodes using nanotron's swarm concept.

Fixed Location

This uses fixed reference points or 'Anchors' to provide location awareness. Anchors are connected to a standard network, and a central computer or server tracks the positions of the tags. Because this system is based on time difference of arrival (TDOA), only one data packet sent from the tag is required to get a position fix in 1D, 2D or 3D. The need to only transmit a single packet reduces power consumption of the tag significantly. Less packets in the air per position fix allow for a larger number of objects to be tracked.

swarm bee LE module appears as a tag on the Fixed Location System. It is a modular product with a compact footprint suitable for integration into a customized tag

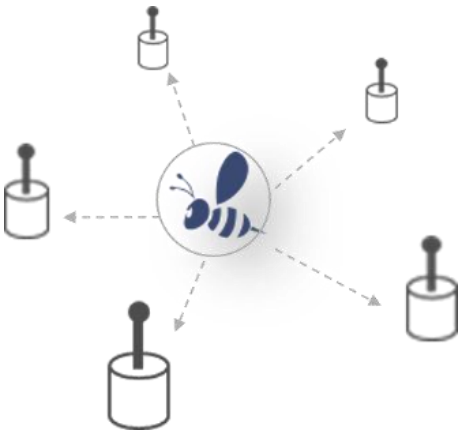

Fig. 3 Fixed location utilizing location infrastructure with fixedradio nodes as anchors

4. Application Programming Interface

4.1. General Requirements

This application programming interface has been implemented so that it allows users to create their own applications if necessary. It consists of a low level API based on SDS-TWR ranging and other commands supporting data communication. The embedded software can be used to implement the following functionalities:

- Peer-to-peer ranging to any node ID
- Low data rate communication
- Serial interface to a host
- Power management features

4.2. General Communication Protocols

Communication between the host controller and the swarm radio is supported by an ASCII protocol. This defines a set of commands with its corresponding parameters and its return values. The return values can be sent by the swarm radio immediately after the command is received or later once the action indicated by the command is performed, asynchronous return values.

4.2.1. ASCII Protocol

For the general communication protocol the following conventions apply:

1. All communication via the interface is done by ASCII characters. This implies that e.g. a 6 byte node ID (hexadecimal) will be transmitted in the following format:

Node ID (hex)												
0000BF260468	0	0	0	0	B	F	2	6	0	4	6	8
ASCII (hex)	30	30	30	30	42	46	32	36	30	34	36	38

2. All command communication ends with carriage return / line feed:

Command termination			
… \r\n	…	\r	\n
ASCII (hex)	…	0D	0A

3. All command codes and their respective parameters are separated by one space character (ASCII 20)

   Example (String): `"RATO 0000BF260468\r\n"`

   Example (Bytes): `{0x52,0x41,0x4F,0x30,0x30,0x30,0x30,0xBF,0x26,0x04, 0x68,0x0D,0x0A }`

4. All commands are transmitted MSB first, LSB last.

5. Return code for unknown or erroneous command is „=ERR\r\n“

   Example:
   ```
   WrongCommand xyz
   =ERR<CR, LF>
   ```

6. To all commands which return one single line, the reply begins with ‘=’ (ASCII Hex 3D).

   Example:
   ```
   GNID
   =001122334455
   ```

7. To all commands which return multiple lines, the reply begins with '#' (ASCII Hex 23), followed by the number of lines.

   Example:
   ```
   GIDL 240 011
   #007
   240
   011
   004
   009:1F3C26041968,1,-19366,+01203,+28214,+26
   026:1F31051999C3,1,+26468,-31599,+18501,+24
   098:1F318052001A,2,+08221,-21266,+12305,+22
   239:1F3CDD3221FA,1,+20122,+32471,-10122,+18
   ```

8. All asynchronous lines start with '*' (ASCII Hex 2A).

   Example:
   ```
   *RRN:001122334455,0,0010.3,-37
   ```

4.3. Power Management

Swarm radios supports three different power modes, which can be selected with the API or on the hardware. The three modes are:

- Receiver always active (no power management). Can be set with the API command SPSA 0.
- Node sleeping, receiver only active after node ID blink for the specified RxWindow. UART is still active. Can be set with the API command SPSA 1. The RxWindow can be changed with the command SRXW.
- Deep sleep, receiver only active after node ID blink for the specified RxWindow. USART is not active. This mode is set in the hardware and it overwrites the two previous modes.

Note: When power saving is active, the *swarm* radio is **only** able to receive ranging requests, data packets and NodeIDs during the time span specified by the SetRxWindow SRXW command following immediately after its own NodeID broadcast. Otherwise any communication over the air will be ignored.

The default configuration receiver always active, which is compatible to previous API command descriptions:

When a swarm radio A node wants to communicate to another swarm radio B in sleep mode it first stores the information until it receives the swarm radio B's NodeID.

Swarm radios store one ranging request or data packet per node ID. The number of node IDs that can be stored may vary between swarm radio types. Swarm bee LE for instance supports 20. When a second ranging or data operation is signaled to the same node before the previous communication has been successfully established the first communication will be canceled and replaced by the new request. If the maximum number of ranging requests or data packets is reached an error will occur.

4.4. UART API Command Set Overview

This chapter summarizes and categorizes the API Command Set.

4.4.1. *swarm* bee radio Setup Commands

SNID	**S**ets the **N**ode **ID** of *swarm* node
GNID	**G**et **N**ode **ID** of the node connected to host
SSET	**S**ave **SET**tings; saves all settings including Node ID permanently to EEPROM
RSET	**R**estore **SET**tings from EEPROM (node configuration)
GSET	**G**et current **SET**tings (node configuration)
SFAC	**S**et node configuration to **FAC**tory default settings
SPSA	**S**et **P**ower **S**aving **A**ctive; sets power management mode on/off
STXP	**S**ets transmission (**TX**) **P**ower of the node
SSYC	**S**et the PHY **SY**n**C**word of *swarm* node
BLDR	**B**oot**L**oa**D**e**R**, switch to bootloader.

4.4.2. Ranging Commands

EPRI	**E**nable **PRI**vacy mode**;** Enables and disables automatic response to received ranging requests.
SPBL	**S**ets **P**rivacy **B**lack**L**ist; handles the privacy black list used to block ranging operation with specified node IDs; maximum number of entries is 20.
GPBL	**G**et **P**rivacy **B**lack**L**ist.
RATO	**RA**nge **TO**; initiates an elementary ranging cycle to another *swarm* node
BRAR	**B**roadcast **RA**nging **R**esults; enabled (or disabled) the broadcasting of ranging **r**esults after each successful ranging operation.
SROB	**S**elects **R**anging **O**peration **B**links; sets which classes of devices the node will initiate a ranging operation with upon reception of a node blink ID packet.
SRWL	**S**ets **R**ange **W**hite **L**ist; Handles the list of node IDs the node should range to. Maximum number of entries is 20.
GRWL	**G**et **R**anging **W**hite **L**ist.
ERRN	**E**nables (or disables) **R**anging **R**esult **N**otification
SROF	**S**ets **R**ange **OF**fset (in ms), this is fixed delay before ranging to autonomous devices

4.4.3. Data Communication Commands

EDAN	**E**nables and disables **DA**ta **N**otification
SDAT	**S**ends **DAT**a to node ID
GDAT	**G**ets received **DAT**a
BDAT	**B**roadcasts **DAT**a
FNIN	**F**ill data into **N**ode **I**D **N**otification packets.
ERAD	**E**nables (or disables) **R**anging **D**ata transmission along with a ranging operation initiated by RATO.
FRAD	**F**ills the **RA**nging data buffer. This data will be transmitted with the next RATO operation if EnableRangingData is enabled.
EIDN	**E**nables and disables Node **ID** Broadcast **N**otification

4.4.4. *swarm* radio Node Identification

EBID	**E**nable **B**roadcast **ID**;enables and disables broadcast of Node ID blink packets
SBIV	**S**ets the **B**roadcast **I**nterval **V**alue (or blinking rate)

NCFG	**N**otification **C**on**F**i**G**uration is used to define which information is visible after receiving *RRN or *NIN type of notifications.

4.4.5. Medium Access Commands

SRXW	**S**ets reception, **RX, W**indow during which he receiver listens after its Broadcast ID.
SRXO	**S**ets **RX W**indow **O**ccurrence; sets whether the receives listens after every blink, after every 2 blink,.. It will listen after every SRXO
SDCL	**S**ets the **D**evice **CL**ass of the node (1…8).
SFEC	**S**witches **F**orward **E**rror **C**orrection (FEC) on and off.
SDAM	**S**ets **DA**ta **M**ode and air communication speed (80/1 and 80/4 mode).

4.4.6. Sensor Commands

EMSS	**E**nables the **M**EMS **S**ensor
EBMS	**E**nable **B**roadcast **M**EM**S** within Node ID Blink packets.
SMRA	**S**ets **M**EMs' **RA**nge, i.e. +-2,4,8 or 16g.
SMTH	**S**ets the **M**EMS **TH**reshold for the slope interrupt.
SMBW	**S**ets **M**EMs' filter **B**and**W**idth of the MEMS.
SMSL	**S**ets **M**EMs' **SL**eep time of the sensor (0.5 … 1000ms)
SMDT	**S**ets **M**EMs' **D**ead**T**ime. Its the minimum time between two possible interrupts.
GMYA	**G**ets **MY A**cceleration; acceleration value of the node connected to the host
GMYT	**G**ets **MY T**emperature; temperature of the node connected to the host
GBAT	**G**ets **BAT**tery status of the node connected to the host
GPIO	Configure **GPIO** pins.
SPIN	**S**ets GPIO **PIN**s.
GPIN	**G**ets GPIO **PIN** status.
ICFG	**I**nterrupt **C**on**F**i**G**uration**.**

4.5. UART API Command Set

In this part the API commands are described and examples are shown how to use them.

4.5.1. *swarm* radio Setup Commands

SNID <ID>:

Description: Sets the Node ID of *swarm* node to <ID>
Parameters: <ID>
Format: 12 bytes (hex)
Range: 000000000000 … FFFFFFFFFFFE
000000000000 is not a valid address but resets the original unique Node ID which was set during production if supported by µC, otherwise: 000000000001

Example: `SNID 0000BF260468`

Return value: **=<ID>**
Format: 12 bytes (hex)
Range: 000000000000 … FFFFFFFFFFFE
Description: configured 6 byte Node ID of *swarm* node
if set ID = 000000000000 then default ID is returned

GNID <void>:

Description: Readback of configured Node ID of node connected to host
Parameters: void

Example: `GNID`

Return value: **=<ID>**
Format: 12 bytes (hex)
Range: 000000000001 … FFFFFFFFFFFE
Description: configured 6 byte Node ID of *swarm* node

SSET<void>:

Description: Saves all settings including Node ID permanently to EEPROM
Parameters: void

Example: `SSET`

Return value: **=<ErrorCode>**
Format: 1 byte (dec)
Range: 0 … 1
Description: Result of saving operation
errorcode = 0 Saving of all parameters successfully verified
errorcode = 1 Saving of parameters not successful; verification failed

RSET<void>:

Description: Restores all parameter settings from EEPROM
Parameters: void

Example: `RSET`

Return value: **=<ErrorCode>**
Format: 1 byte (dec)
Range: 0 … 1
Description: Result of restoring operation
errorcode = 0 Restoring of all parameters successful
errorcode = 1 Restoring parameters from EEPROM failed

GSET<void>:

Description: Reads current device configuration. First line is the number of following lines.
All others state the name of parameter separated with ':' and value. The value depends on parameter.

```
#<NumLines>
<ParameterName>:<Value>
<ParameterName>:<Value>
…
```

Parameters: void

Example:
```
GSET
```

Return values:

<NumLines>
Format: 4 bytes (dec), first byte fixed „#"
Range: 000 … 255
Description: Number of lines after this line

<ParameterName>
Format: ASCII
Description: Name of following parameter value.

<Value>
Format: Depends on parameter.

Example:
```
#030
BRAR:1
CSMA:4,255,18
EBID:1
EBMS:1
EDAN:1
EIDN:0
EMSS:1
EPRI:0
ERAD:0
ERRN:1
FW_VER: …
GPIO:0F,0,3,0,1
ICFG:0
NCFG:0004
SBIV:30000
SDAM:1
SDCL:1
SFEC:0
SMBW:6
SMDT:01000
SMRA:2
SMSL:01
SMTH:30
SNID:XXXXXXXXXXXX
SPSA:0
SROB:001
SROF:000
SRXW:00010
SSYC:1
STXP:63
```

SFAC<void>:

Description: Resets device configuration to factory settings (MEMS cf. 4.11).
Default configuration is described in chapter 4.5 Default Settings.

Parameters: void

Example: `SFAC`

Return value: **=<ErrorCode>**
Format: 1 byte (dec)
Range: 0…1
Description: Result of writing operation to EEPROM
errorcode = 0 EEPROM successfully written
errorcode = 1 Writing parameters to EEPROM failed

SPSA <Type>
Description: Sets power management mode for *swarm* radio
Parameters: <Type>
TYPE = 0 Power saving off, device is continuously receiving
TYPE = 1 Power saving on, wake up on any interrupt
Format: 1 byte (dec)
Range: 0 … 1

Return value: **=<Type>**

STXP <Power>:
Description: Sets the node's transmission power (0=minimum … 63=maximum).
Parameters: <Power>
Format: 2 bytes (dec)
Range: 00 … 63

Example: `STXP 63`

Return value: **=<Power>**

SSYC <SyncWord>:
Description: Sets the node's PHY syncword (0 … 8). The node will only listen to messages with its same syncword.
Parameters: <SyncWord>
Format: 1 byte (dec)
Range: 0 … 8

Example: `SSYC 1`

Return value: **=<SyncWord>**

BLDR <void>:
Description: Restarts device and start bootloader.
Parameters: <void>

Example: `BLDR`

Return value: **=**0

4.5.2. Ranging Commands

EPRI <Enable>:

Description: Enable Privacy mode: Enables and disables response to a received ranging request.
Parameters: <Enable> = 0 Node will respond to ranging requests
<Enable> = 1 Node will not respond to ranging requests
Format: 1 byte (dec)
Range: 0 … 1

Example: `EPRI 1`

Return value: **=<Enable>**

SPBL <Option> <ID>:

Description: Sets privacy blacklist: Handles the privacy black list used to block ranging operation with specified node IDs; maximum number of entries is 20.
Parameters: <Option> = 0 Clear black list
<Option> = 1 Add ID to the black list
<Option> = 2 Remove ID from the black list
Format: 1 byte (dec)
Range: 0 … 2

Parameters: <ID> 6 byte Node ID of ranging partner node
Format: 12 bytes (hex)
Range: 000000000001 … FFFFFFFFFFFE

Example: `SPBL 1 0000BF260468`

Return value: **=<ErrorCode>**
Format: 1 byte (dec)
Range: 0…2
Description: indicating status of operation
<ErrorCode> = 0 success, value stored/removed or list cleared
<ErrorCode> = 1 list full, not possible to add more ID
<ErrorCode> = 2 ID to be removed is not in the list

GPBL <void>:

Description: Gets privacy blacklist.
Parameters: void

Example: `GPBL`

Return values: **<NumLines>**
Format: 4 bytes (dec), first byte fixed „#“
Range: 000 … 255
Description: Number of lines after this line

<NodeID>
Format: 12 bytes (hex)
Range: 000000000001 … FFFFFFFFFFFE

<Value>
Format: Depends on parameter.

Example:
```
#003
DDF451534C23
134683567ABC
33A441FFB311
```

RATO <Option=0> <ID>:
RATO <Option=1> <ID> <TimeOut>:

Description: Range to: Initiates an elementary ranging cycle to node with node ID <ID>
Parameters: <Option> = 0 Immediate start ranging operation
<Option> = 1 Wait for node <ID> blink before starting ranging operation
Format: 1 bytes (dec)
Range: 0 … 1

<ID> 6 byte Node ID of ranging partner node
Format: 12 bytes (hex)
Range: 000000000001 … FFFFFFFFFFFE

<TimeOut> maximum time in ms to wait for the node <ID> blink.
Format: 5 bytes (dec)
Range: 0 … 65000

Example: `RATO 1 0000BF260468 1000`

Return value with **<Option=0>:** **=<ErrorCode>,<RangingResult>,<RSSI>**

<ErrorCode>
Format: 1 byte (dec)
Range: 0 … 5

Description: indicating status of ranging operation
Errorcode = 0: success ranging result valid
Errorcode = 1: ranging to own ID
Errorcode = 2: ID out of range, no ACK
Errorcode = 3: ranging unsuccessful, ACK OK, then timeout
Errorcode = 5: Timeout <DURATION> has been reached

<RangingResult>
Format: 6 bytes (float)
Range: 0000.0 … 99999 ranging distance in meters
Description: returning the measured ranging distance in meters

<RSSI>
Format: 4 byte (dec)
Range: -128…-35
Description: RSSI value of remote node.

Return value with **<Option=1>:** **=<Error>**

<Error>
Format: 1 byte (dec)
Range: 0 … 1
Description: Indicate status of operation.
Error = 0: success, request accepted
Error = 1: error, ranging to own ID is not allowed

BRAR <Enable>:

Description: Broadcast ranging results: Enables/Disables the broadcast transmission of ranging results after each successful ranging.

Parameters: ENABLE = 0 broadcast disabled
ENABLE = 1 broadcast enabled
Format: 1 byte (dec)
Range: 0 … 1

Example: `BRAR 1`

Return value: **=<Enable>**

SROB <ClassMask>:

Description: Sets range on broadcast: Selects if and to which class of devices <ClassMask> the Node will initiate a ranging operation upon reception of a node blink ID.

Parameters:
ClassMask = 0 (=bin 0000 0000) Range on Broadcast disabled
ClassMask = 1 (=bin 0000 0001) Range on Broadcast to device class 1
ClassMask = 2 (=bin 0000 0010) Range on Broadcast to device class 2
ClassMask = 4 (=bin 0000 0100) Range on Broadcast to device class 3
ClassMask = 8 (=bin 0000 1000) Range on Broadcast to device class 4
ClassMask = 16 (=bin 0001 0000) Range on Broadcast to device class 5
ClassMask = 32 (=bin 0010 0000) Range on Broadcast to device class 6
ClassMask = 64 (=bin 0100 0000) Range on Broadcast to device class 7
ClassMask = 128 (=bin 1000 0000) Range on Broadcast to device class 8

Device classes can be combined by performing a logical OR of ClassMask, e.g. ClassMask=162 (=bin 1010 0010) Range on Broadcast to classes 2,6,8

Format: 3 bytes (dec)
Range: 000 … 255

Example: `SROB 162`

Return value: **=<ClassMask>**

SRWL <Option> <ID>:

Description: Sets range white list: Handles the list of node IDs the node should range to. Maximum number of entries is 20.

Parameters:
<Option> = 0 Clear list
<Option> = 1 Add ID to the list
<Option> = 2 Remove ID from the list

Format: 1 byte (dec)
Range: 0 … 2

<ID> 6 byte Node ID of ranging partner node
Format: 12 bytes (hex)
Range: 000000000001 … FFFFFFFFFFFE

Example: `SRWL 1 0000BF260468`

Return value: **=<Error>**
Format: 1 byte (dec)
Range: 0…2
Description: indicating status of operation
<Errorode> = 0: success; value stored/removed or list cleared
<Errorode> = 1: list full, not possible to add more ID
<Errorode> = 2: ID to be removed is not in the list

GRWL <void>:

Description: Gets ranging whitelist.
Parameters: void

Example: `GRWL`

Return values: **<NumLines>**
Format: 4 bytes (dec), first byte fixed „#“
Range: 000 … 255
Description: Number of lines after this line

<NodeID>
Format: 12 bytes (hex)
Range: 000000000001 … FFFFFFFFFFFE

<Value>
Format: Depends on parameter.

Example:
```
#003
DDF451534C23
134683567ABC
33A441FFB311
```

ERRN <Notify>:

Description: Enables and disables ranging result notification
Parameters: <Notify> = 0 Node will not trigger host when data packet has been received
<Notify> = 1 Node will trigger host when data packet has been received
Format: 1 byte (dec)
Range: 0 … 1

Example:
```
ERRN 1
```

Return value: **=<Notify>**

SROF <Duration>

Description: Sets ranging offset: fixed delay in ms before starting automatic ranging.
Parameters: <Duration>
Format: 5 bytes (dec)
Range: 0 … 65000

Example:
```
SROF 00901
```

Return value: **=<Duration>**

4.5.3. Data Communication Commands

EDAN <Notify>:

Description: Enables and disables data notification
Parameters: <Notify> = 0 Node will not trigger host when data packet has been received
<Notify> = 1 Node will trigger host when data packet has been received
Format: 1 byte (dec)
Range: 0 … 1

Example: `EDAN 1`

Return value: **=<Notify>**

SDAT <Option=0> <ID> <Len> <Data>:
SDAT <Option=1> <ID> <Len> <Data> <TimeOut>:

Description: Sends <Data> of length <Len> to node <ID>.Depending on <OPTION> it transmits immediately or after the node <ID> blink.
Parameters: <Option> = 0 Immediate transmission
<Option> = 1 Wait for node <ID> blink before transmitting
Format: 1 bytes (dec)
Range: 0 … 1

<ID> 6 byte Node ID of ranging partner node
Format: 12 bytes (hex)
Range: 000000000001 … FFFFFFFFFFFE

Note: If ID is all FF than it send always after receiving a node ID notification until <TimeOut>

<Len> length of payload in bytes (hex)
Format: 2 bytes (hex)
Range: 01 … 80

<Data> payload to be transmitted
Format: 2 bytes (hex) <LEN> times 2 bytes of payload
Range: 00 … FF

<TimeOut> maximum time in ms to wait for the node <ID> blink.
Format: 5 bytes (dec)
Range: 0 … 65000

Example: `SDAT 1 1F318052001A 02 FA13`

Return value with **<Option=0>**: **=<Error>**
Description: Status of transmission.
Format: 1 byte (dec)
Range: 0 ... 1
<Error> = 0 Transmission successful
<Error> = 1 Transmission failed

Return value with **<Option=1>**: **=<PayloadID>**
Description: ID of <Data>, used to identify the asynchronous return value (*SDAT)
Format: 8 byte (hex)
Range: 00000000 … FFFFFFFF

Note: After issuing a SDAT command and asynchronous response will be generated when the transmission is processed. The response is described in chapter 4.6.4.

GDAT <void>:

Description: Gets data: Reads out transmitted data
Parameters: void

Example: `GDAT`

Return values: **=<Number of bytes>,<ID>,<Payload>**

<Number of bytes>
Format: 2 bytes (hex)
Range: 00…80
Description: returns the number of bytes in pending message
Number of bytes = 00: no pending message available
Number of bytes = 01…80 number of bytes in message

<ID>
Format: 12 bytes (hex)
Range: 000000000001 … FFFFFFFFFFFF
Description: returns ID of node which sent message

<Payload>
Format: 2 bytes (hex) <Number of bytes> times 2 bytes of payload
Range: 00 … FF
Description: payload received

BDAT <Option=0> <Len> <Data>:
BDAT <Option=1> <Len> <Data> <TimeOut>:
BDAT <Option=2>:

Description: Broadcasts <Data> of length <Len> to all nodes
Parameters: <Option> = 0 Immediate broadcast transmission
<Option> = 1 Send the packet after any node ID blink.
<Option> = 2 Cancel broadcasting
Format: 1 bytes (dec)
Range: 0 … 1

<Len> length of payload in bytes (hex)
Format: 2 bytes (hex)
Range: 01 … 71

<Data> payload to be transmitted
Format: 2 bytes (hex) <len> times 2 bytes of payload
Range: 00 … FF

<TimeOut> = 0 timeout disabled, broadcast will always occur.
<TimeOut> > 0 time in ms during which the node transmits after a node ID blink.
Format: 5 bytes (dec)
Range: 0 … 65000

Example: `BDAT 0 2 FA13`

Return value with **<Option=0>**: **=<Error>**
Format: 1 byte (dec)
Range: 0 ... 1
<Error> = 0 Transmission successful
<Error> = 1 Transmission failed
Description: Status of transmission.

Return value with **<Option=1>**: **=<PayloadID>**

Format: 8 byte (hex)
Range: 00000000 … FFFFFFFF
Description: ID of <Data>, used to identify the asynchronous return value (*SDAT)

Note: After issuing a SDAT command and asynchronous response will be generated when the transmission is processed. The response is described in chapter 4.6.4.

Return value with **<Option=2>**: **=0**

ERAD <Enable>:

Description: Enable ranging data: Enables and disables the transmission of data from the ranging data buffer along with a ranging operation initiated by RATO.

Parameters:
<Enable> = 0 Ranging data will not be transmitted
<Enable> = 1 Ranging data will be transmitted with RATO command

Format: 1 byte (dec)
Range: 0 … 1

Example: `ERAD 1`

Return value: **=<Enable>**

FRAD <Len> <Data>:

Description: Fills the ranging data buffer with <Data> of length <Len>. This data will be transmitted with the next RATO operation if ERAD is <On>. The ranging data <Data> is contained within the ranging packet itself.

Parameters:
<Len> length of ranging data payload in bytes (hex)
Format: 2 bytes (hex)
Range: 01 … 74

<Data> payload to be transmitted
Format: 2 bytes (hex) <Len> times 2 bytes of payload
Range: 00 … FF

Example: `FRAD 0A FA13192F680426AE2345`

Return value: **=<ErrorCode>**
Format: 1 byte (dec)
Range: 0 … 1
Description: Status on ranging data buffer fill operation
Errorcode = 0: successful
Errorcode = 1: not successful

EIDN <Notify>:

Description: Enables and disables Node ID broadcast notification

Parameters:
<Notify> = 0 Node will not trigger host when Node ID Broadcast has been received
<Notify> = 1 Node will trigger host when Node ID Broadcast has been received

Format: 1 byte (dec)
Range: 0 … 1

Example: `EIDN 1`

Return value: **=<Notify>**
Format: 1 byte (dec)
Range: 0 …1
Description: returning parameter which has been set NOTIFY
Node ID Broadcast enabled/disabled

EBMS <Notify>:

Description: Enables and disables broadcasting of MEMS and TEMP values in blink packet

Parameters:
<Notify> = 0 Node will not broadcast MEMS & TEMP values in blink packet
<Notify> = 1 Node will broadcast MEMS &TEMP values in blink packet

Format: 1 byte (dec)
Range: 0 … 1

Example: `EBMS 1`

Return value: **=<Notify>**

FNIN <Len=0>:
FNIN <Len>0> <Data>:

Description: Fills the data buffer of each node ID notification with <Data> of length <Len>. This data will be transmitted with every node ID notification until deleted by host application.

Parameters:	<Len>	length of ranging data payload in bytes (hex)
Format:	2 bytes (hex)	
Range:	0 … 60	
	0 = delete data	
	<Data>	payload to be transmitted
Format:	2 bytes (hex)	<Len> times 2 bytes of payload
Range:	00 … FF	

Example: `FNIN 0A FA13192F680426AE2345`

Return value: **=<ErrorCode>**

Format:	1 byte (dec)
Range:	0 … 1
Description:	Status on buffer fill operation
Errorcode = 0:	successful
Errorcode = 1:	not successful

4.5.4. *swarm* radio Node Identification

EBID <Enable>:

Description: Enable/Disable broadcast node ID: Enables/disables broadcast of Node ID blink packets.
Parameters: <Enable> = 0 Broadcast of Node ID blink packets disabled
<Enable> = 1 Broadcast of Node ID blink packets enabled
Format: 1 byte (dec)
Range: 0 … 1

Example: `EBID 1`

Return value: **=<Enable>**
Format: 1 byte (dec)
Range: 0 …1
Description: returning parameter which has been set
ENABLE = 0 Broadcast of Node ID blink packets disabled
ENABLE = 1 Broadcast of Node ID blink packets enabled

SBIV <Time>:

Description: Sets the broadcast interval in which the Node ID will be sent
Parameters: <Time> Blink interval in milliseconds [ms]
Format: 5 bytes (dec)
Range: 00000 … 65000

Example: `SBIV 10000`

Return value: **=<Time>**

NCFG <Mask>

Description: **<Mask>** sets visible information in ranging result notifications and node ID notifications. This allows control over information displayed each time a "*RRN:" or "*NIN:" is generated.
Parameters: **<Mask>**
Format: 4 bytes (hex)
Range: 0 ...FFFF

Bit 0 = Device Class
Format: 1 byte (dec)
Range: 1 …8
Description: Device class of remote node

Bit 1 = MEMS values consists **of** x,y,z acceleration.
<XACC>
Format: 6 bytes (dec), signed INT16, first byte "+" or "-"
Range: -32768 … +32767
Description: Acceleration value.
<YACC> See <XACC>
<ZACC> See <XACC>

Bit 2 = RSSI values
Format: 4 byte (dec)
Range: -128 … -35
Description: RSSI value of remote node.

Bit 3 = TEMP values
Format: 3 byte (dec), first byte "+" or "-"
Range: -99… +99
Description: Temperature value of remote node.

Bit 4 = Power saving mode
Format: 1 byte (dec)
Range: 0 … 2
0 = receiver of remote node is always active
1 = remote node is in sleep mode

2 = remote node is in deep sleep
Description: Power saving mode of remote node.

Bit 5 = Battery level
Format: 3 bytes (dec)
Range: 0 … 255
Description: Node's own battery level is in dV.
(Example: 32 = 3.2V)

Bit 6 = GPIO Status
Format: 2 bytes (hex)
Range: 00 … FF
Bit 0 = PA0 = DIO_0
Bit 1 = PA2 = DIO_1
Bit 2 = PA3 = DIO_2
Bit 3 = PA8 = DIO_3
Description: GPIO Pin status.

Bit 7 = Wakeup reason
Format: 2 bytes (hex)
Range: 00 … FF
Bit 0 = DIO_0
Bit 1 = DIO_1
Bit 2 = DIO_2
Bit 3 = DIO_3
Bit 4 = MEMS
Description: Wake up reason of remote device. If no bit is set, it's the normal blink wake up for node ID notification.

Bit 8 = BlinkID
Format: 3 bytes (dec)
Range: 0 … 255
Description: Blink ID.

Bit 9 = RX Backchannel counter.
Format: 3 bytes (dec)
Range: 0 … 255
Description: RX Backchannel counter.

Example: `NCFG FF`

Return value: =<Mask>

4.5.5. Medium Access Commands

SRXW <Time>:

Description: Sets reception window, time span the receiver listens after sending its broadcast ID in power saving mode.
Mnemonic: **SRXW**
Parameters:

<Time> = 0 Receiver is continuously disabled
<Time> = 1-65000 Time in ms that the receiver stays on after its Node ID broadcast. If TIME> NodeID broadcast intervall the receiver is always active

Format: 5 bytes (dec)
Range: 0 … 65000

Example: `SRXW 00010`

Return value: **=<Time>**

SRXO <value>:

Description: Sets reception window occurrence.
Mnemonic: **SRXO**
Parameters:

<value> = 0 Reception window disabled.
<value> = 1-255 Number of Node ID Blinks with a RxWindow. If 1, every Node ID Blink a RxWindow, if 2, every second Node ID Blink a RxWindow.

Format: 3 bytes (dec)
Range: 0 … 255

Example: SRXO 1

Return value: **=<value>**

SDCL <Class>:

Description: Sets the node's device class which will be broadcasted in its blink ID packets.
Parameters: <Class>
Format: 1 byte (dec)
Range: 1 … 8

Example: `SDCL 1`

Return value: **=<Class>**

SFEC <Enable>:

Description: Enables forward error correction (FEC)
Parameters: <Enable> = 0 FEC off
<Enable> = 1 FEC on
Format: 1 byte (dec)
Range: 0 … 1

Example: `SFEC 1`

Return value: **=<Enable>**

SDAM <Mode>:

Description: Sets the node's data mode
Parameters: <Mode> = 1 80MHz, 1µs per bit mode
<Mode> = 2 80MHz, 4µs per bit mode
Format: 1 byte (dec)
Range: 1 … 2

Example: `SDAM 2`

Return value: **=<Mode>**

4.5.6. Sensor Commands

EMSS <Enable>:

Description: Enables and disables the MEMS sensor on the *swarm* radio.
Parameters: <Enable> = 0 MEMS sensor will be disabled on node
<Enable> = 1 MEMS sensor will be enabled on Node
Format: 1 byte (dec)
Range: 0 … 1

Example: `EMSS 1`

Return value: **=<Enable>**

SMRA <Grange>:

Description: Sets the MEMS range that is the sensor's acceleration scale.
Parameters: <Grange> = 1 MEMS g range set to +/- 2g
<Grange> = 2 MEMS g range set to +/- 4g
<Grange> = 3 MEMS g range set to +/- 8g
<Grange> = 4 MEMS g range set to +/- 16g
Format: 1 byte (dec)
Range: 1 … 4

Example: `SMRA 2`

Return value: **=<Grange>**
Note:
ERR returned if MEMS is switched off

SMTH <Thres>:

Description: Sets the MEMS sensor's threshold definition for the slope interrupt. An LSB corresponds to an LSB of acceleration data and therefore depends on the selected g range (see above).
Parameters: <Thres>
For <Grange> = 1 Threshold = 3.91 mg/LSB * <Thres>
For <Grange> = 2 Threshold = 7.81 mg/LSB * <Thres>
For <Grange> = 3 Threshold = 15.62 mg/LSB * <Thres>
For <Grange> = 4 Threshold = 31.25 mg/LSB * <Thres>

Format: 3 bytes (dec)
Range: 0 … 255

Example: `SMTH 213`

Return value: **=<Thres>**
Note:
ERR returned if MEMS is switched off

SMBW <BandWidth>:

Description: Sets the MEMS sensor's filter bandwidth.
Parameters: <BandWidth>
<BandWidth> = 1 7.81 Hz
<BandWidth> = 2 15.63 Hz
<BandWidth> = 3 31.25 Hz
<BandWidth> = 4 62.50 Hz
<BandWidth> = 5 125.00 Hz
<BandWidth> = 6 250.00 Hz
<BandWidth> = 7 500.00 Hz
<BandWidth> = 8 1000.00 Hz
Format: 1 byte (dec)
Range: 1 … 8

Example: `SMBW 3`

Return value: **=<BandWidth>**
Note:
ERR returned if MEMS is switched off

SMSL <SleepTime>:

Description: Sets the MEMS sensor's sleep time
Parameters: <SleepTime>

<SleepTime> =	1	→	0.5 ms
<SleepTime> =	2	→	1 ms
<SleepTime> =	3	→	2 ms
<SleepTime> =	4	→	4 ms
<SleepTime> =	5	→	6 ms
<SleepTime> =	6	→	10 ms
<SleepTime> =	7	→	25 ms
<SleepTime> =	8	→	50 ms
<SleepTime> =	9	→	100 ms
<SleepTime> =	10	→	500 ms
<SleepTime> =	11	→	1000 ms

Format: 2 bytes (dec)
Range: 1 … 11

Example: `SMSL 11`

Return value: **=<SleepTime>**
Note:
ERR returned if MEMS is switched off

SMDT <DeadTime>:

Description: Sets the MEMS sensor's dead time.
Parameters: **<DeadTime>**
Format: 5 byte (dec)
Range: 0 … 65000
Description: Time in milliseconds with MEMS interrupt disabled, after MEMS interrupt occurred.

Example: `SMDT 1000`

Return value: **=<DeadTime>**

GMYA <void>:

Description: Reports Node's own MEMS acceleration values in mg.
Parameters: void

Example: `GMYA`

Return values: **<XACC>**
Format: 6 bytes (dec), Signed INT16, first byte "+" or "-"
Range: -32768 … +32767
Description: Acceleration value for X coordinate (positive and negative)

<YACC>
Format: 6 bytes (dec), Signed INT16, first byte "+" or "-"
Range: -32768 … +32767
Description: Acceleration value for Y coordinate (positive and negative)

<ZACC>
Format: 6 bytes (dec), Signed INT16, first byte "+" or "-"
Range: -32768 … +32767
Description: Acceleration value for Z coordinate (positive and negative)

Example: `=+31599,+18501,-26468`

Note:

ERR returned if MEMS is switched off

GMYT <void>:

Description: Reports Node's own MEMS temperature values
Parameters: void

Example: `GMYT`

Return values: **=<Temperature>**
Format: 3 bytes (dec), first byte "+" or "-"
Range: -99 … +99
Description: Node's own current temperature (positive and negative) in °C

Example: `=+26`

Note:
ERR returned if MEMS is switched off

GBAT <void>:

Description: Reports battery level of the node connected to the host
Parameters: void

Example: `GBAT`

Return values: **=<BatteryStatus>**
Format: 3 bytes (dec)
Range: 000 … 255
Description: Node's own battery level in dV (example: =025 is 2.5V)

Example: `=025`

GPIO <pins> <mode> <speed> <otype> <pupd>:

Description: Configure GPIO's of µC.
Parameters: <pins>
Format: 2 bytes (hex)
Range: 00 …FF
Description: <pins> defines which pins affected by changes.
Bit 0 = PA0 = DIO_0
Bit 1 = PA2 = DIO_1
Bit 2 = PA3 = DIO_2
Bit 3 = PA8 = DIO_3

<mode>
Format: 1 byte (dec)
Range: 0 ... 3
Description: 0 = GPIO Input Mode
1 = GPIO Output Mode
2 = GPIO Alternate function Mode
3 = GPIO Analog Mode

<speed>
Format: 1 byte (dec)
Range: 0 ... 3
Description: 0 = Very Low Speed (400KHz)
1 = Low Speed (2MHz)
2 = Medium Speed (10MHz)
3 = High Speed (40MHz)

<otype>
Format: 1 byte (dec)
Range: 0 … 1
Description: 0 = Push Pull
1 = Open Drain

<pupd>
Format: 1 byte (dec)
Range: 0 … 2
Description: 0 = No pull
1 = Pull up
2 = Pull down

Example: GPIO 03 1 3 0 1

Return values: **=<pins>, <mode>, <speed>,<otype>,<pupd>**

Example: `=03,1,3,0,1`

SPIN <mask> <status>:

Description: Set new pin status.
Parameters: <mask>
Format: 2 bytes (hex)
Range: 00… FF
Description: <pins> defines which pins affected by changes.
Bit 0 = PA0 = DIO_0
Bit 1 = PA2 = DIO_1
Bit 2 = PA3 = DIO_2
Bit 3 = PA8 = DIO_3
<status>
Format: 2 byte (hex)
Range: 00 ... FF
Description: 0 = Bit reset
1 = Bit set

Example: `SPIN 03 01`

Return values: **=<mask>, <status>**

GPIN <void>:

Description: Read current pin status.
Parameters: void

Example: GPIN

Return values: **=<status>**
Format: 2 byte (hex)
Range: 0 … FF
Description: Each bit indicates a pin status 0 = reset, 1 = set.

Example: `=01`

ICFG <setting>:

Description: Configure and enable interrupt sources.
Parameters: <settings>
Format: 4 byte (hex)
Range: 0 … FFFF
Description: Bit 0 = PA0 = DIO_0 = Rising edge enabled/disabled
Bit 1 = PA0 = DIO_0 = falling edge enabled/disabled
Bit 2 = PA2 = DIO_1 = Rising edge enabled/disabled
Bit 3 = PA2 = DIO_1 = falling edge enabled/disabled
Bit 4 = PA3 = DIO_2 = Rising edge enabled/disabled
Bit 5 = PA3 = DIO_2 = falling edge enabled/disabled
Bit 6 = PA8 = DIO_3 = Rising edge enabled/disabled
Bit 7 = PA8 = DIO_3 = falling edge enabled/disabled
Bit 8 = MEMS interrupt enabled/disabled

Example: `SINT 0001`

Return values: **=<setting>**

4.6. Format for Notification Messages

This chapter describes the format of all asynchronous messages.

4.6.1. Format for Data Notification Messages

This chapter describes the communication structure for Data Notifications when data notification has been enabled.

Notification format: Data Notification Flag (fixed): <ID>

Data Notification Flag:
Format: 4 bytes starting with "*" to signal identification flag
Content: *DNO

ID:
Format: 12 bytes (hex)
Range: 000000000001 … FFFFFFFFFFFE
Description: returns ID of node which sent message

Example: `*DNO:1F3CFF322133`

4.6.2. Format for Node ID Notification Messages

This chapter describes the communication structure for Node ID Notifications when node ID notification has been enabled.

Notification format: NodeID Notification Flag (fixed): <ID>, <MORE>

Node ID Notification Flag:
Format: 4 bytes starting with "*" to signal identification flag
Content: *NIN

ID:
Format: 12 bytes (hex)
Range: 000000000001 … FFFFFFFFFFFE
Description: returns ID of node which sent message

MORE:

Description: Depends on command NCFG. This configures the appended data.

Example: `*NIN:1F3CFF322133,04,-35`

4.6.3. Format for Ranging Result Notification Messages

This chapter describes the communication structure for Ranging Results Notifications when Ranging Result Notification has been enabled. This is printed whenever a ranging result broadcast is received or an automatic ranging is finished as long as Ranging Result Notification has been enabled.

Notification format: NodeID Notification Flag: <src ID>,<dest ID>,<ERR>,<DISTANCE>,<MORE>

Ranging Notification Flag:
Format: 4 bytes starting with "*" to signal identification flag
Content: *RRN

src ID:
Format: 12 bytes (hex)
Range: 000000000001 … FFFFFFFFFFFE
Description: Returns ID of node which requested ranging.

dest ID:
Format: 12 bytes (hex)

Range: 000000000001 … FFFFFFFFFFFE
Description: Returns ID of node which received ranging request

ERR:
Format: 1 byte (dec)
Range: 0 … 5
Description: indicating status of ranging operation
Errorcode = 0: success ranging result valid
Errorcode = 1: ranging to own ID
Errorcode = 2: ID out of range, no ACK
Errorcode = 3: ranging unsuccessful, ACK OK, then timeout.
Errorcode = 5: timeout duration has been reached.
Errorcode = 6: medium blocked (CSMA give up)

DISTANCE:
Format: 6 bytes (float)
Range: 0000.0 … 9999.9 ranging distance in meters
Description: returning the measured ranging distance in meters

MORE:
Description: Depends on command NCFG. This configures the appended data.

Example: `*RRN:1F3123123133,1F3CFF322133,0,0010.3,-68`

4.6.4. Format for SDAT Notification Messages

This chapter describes the return message format for asynchronous SDAT Notifications after issuing a SDAT command.

Notification format: *SDAT:<ID>,<ERRORCODE>,<PAYLOAD_ID>

SDAT Notification Flag:
Format: 5 bytes starting with "*" to signal identification flag
Content: *SDAT

ID:
Format: 12 bytes (hex)
Range: 000000000001 … FFFFFFFFFFFE
Description: returns ID of node which sent message

ERRORCODE:
Format: 1 byte (dec)
Range: 0 … 1
Description: indicating status of ranging operation
Errorcode = 0: success data communication valid
Errorcode = 1: error: no hardware ack received.
Errorcode = 2: error: timeout; message could not be delivered
Errorcode = 3: error: medium blocked (CSMA give up)
Errorcode = 4: error: unknown.

PAYLOAD_ID:
Format: 8 bytes (hex)
Range: 00000000 … FFFFFFFF
Description: used to identify the payload

Example: `*SDAT:1F3CFF322133,0,45A6213F`

4.7. API Default Settings

Parameter/ Command	Default Value	Description
Node ID	000000000001	factory pre-configured MAC address
BRAR	1	Broadcast ranging results is enabled.
EBID	1	ID broadcast enabled
SBIV	30000 ms	ID broadcast interval
EPRI	0	Responds to ranging request is true.
EDAN	1	data notification enabled
EIDN	0	Node ID notification is disabled
ERRN	1	Ranging result notification is enabled
CSMA	4, 255, 18	enabled with mode 4, random seed 255 and threshold 18
ERAD	0	Rangin data is disabled.
EBMS	1	Broadcast MEMS data is enabled.
SFEC	0	FEC is disabled.
SDAM	1	80/1 mode enabled
SDCL	1	Device class is 1
SROB	1	range on broadcast to device class 1
SROF	0	Ranging offset is 0.
STXP	63	TX power is 63.
SSYC	1	Syncword is 1.
SRXW	10	RX window is 10 ms.
SRXO	0	RX occurrence is 0 (RX window disabled)
SPSA	0	Power saving is disabled.
SMBW	6	MEMS sensor's bandwidth is 250 Hz.
SMDT	1000	MEMS sensor'sdead time is 1000 ms.
SMSL	1	MEMS sensor's sleep time is 0.5 ms.
SMTH	30	MEMS sensor’s threshold is set to 30.
SMRA	2	MEMS acceleration scale (g range) is +/- 4g.
ICFG	0	All interrupts are disabled.
NCFG	2	Notification configuration is set to RSSI only.
GPIO	FF, 0, 3, 0, 1	All inputs are with PullUp.

4.8. Settings for Different Node Behaviours

The *swarm* bee settings are predefined with some default values such that the user only needs to connect power to start working with it. This default behavior is as follows:

Every 30s the *swarm* bee broadcasts a blink with its nodeID and sensor data in it. During the time between blinks the radio is in receiving mode, waiting for different kind of messages which are listed in the following:

- A blink from another *swarm* radio: Whenever it receives a blink from any other *swarm* radio, it initiates a ranging operation with that radio. The ranging operation consists of an exchange of packets in which both *swarm* radios participate. Once it has all the necessary data it estimates the distance to the other *swarm* radio and broadcast it.

- A ranging request: Any *swarm* in the neighborhood who receive its nodeID blink may start a ranging operation with the *swarm*. In this case, the *swarm* answers by sending all necessary packets.

- A range result broadcast: After a ranging operation the *swarms* broadcast the result so that all the neighbors have access to that information. The range result may indicate the distance between the receiving *swarm* itself and the sender or between the sender and any other *swarm* radio in the area. When a *swarm* which is connected to a host receives a range result broadcast packet, it passes it to the host as a range result notification (RRN).

- Any other data packet from any swarm in the neighborhood: In case the *swarm* is connected to a host it notifies the host that data was received. It is up to the host controller to read the data or not (API command: GDAT).

This 'default' behavior of the *swarm* can be modified using the API commands. Some of the possible modifications are:

- The blinking interval can be changed (API command SBIV) or even deactivated (API command EBID) so that the *swarm* will not blink.

- The *swarm* bee can be set in low power mode: The swarm can be put in sleep mode so that it wakes up only when it needs to send a blink; after the blink it waits for a while in receive mode in case it receives ranging requests or other packet and it goes to sleep again. When a *swarm* is in this mode a notification is send as part of each blink so that all the nodes around know that if they need to send something to this node they should do it immediately after receiving its blink. The time during which the *swarm* is listening after its blink and whether it listens after each blink or after every x^{th} blink can be set by the user (API commands SRXW and SRXO).

- A privacy mode can be used: When a swarm receives a blink it reacts, by default, by initiating a ranging operation with the blink originator. This behavior can be modified (API command SPRI) so that the *swarm* only reacts to certain devices: Devices of the same class (API command SROB) or devices with certain IDs (API command RATO). This allows the user to reduce the number of packets over the air by not sending unnecessary messages.

- The notifications that are passed to the host can be selected and customized: The swarm can also notify the host controller every time it receives a blink (API command EIDN). Moreover, in every blink the *swarms* can add payload data; the user can decide what data is passed to the host in this notifications (RSSI, sensor data, battery status …). This can be done with the API command NCFG. By default the sensor data is added to the blink payload.

- Send to an individual *swarm* or broadcast data: The command SDAT can be used to send data to any other *swarm* in range. For the two operations the transmitting *swarm* needs to take into account that the receiving *swarms* may be in sleep mode. If they are, they should send the message every time one of the receivers sends a blink.

5. Compliance with CE

In order to certify the swarm radio in compliance with the CE ETSI 300 328 v1.8.1, all nodes are set to transmit using the Carrier Sense Multiple Access (CSMA) mechanism, in energy detection mode and with a threshold equal to or lower than -70dBm.

6. Revision History

Date	Authors	Version	Description
2014-01-31	nanotron	2.0	Initial version. Completely revised and enhanced *swarm* API, introduction of Mnemonic code, no longer compatible with previous API versions
2014-02-13	nanotron	2.01	Implementation changes added, commands EnableRangingResultNotification and SetRangingOffset added, typos corrected
2014-03-18	nanotron	2.02	Commands SetDiversity and SetAntenna removed, commands SetRxWindow and SetDeviceClass moved from Setup to Air Interface, RSSI range corrected, reply prefix added.
2014-10-27	nanotron	2.03	Several command groups removed. New command groups added.

End of Document

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.

About nanotron Technologies GmbH

Today nanotron's ***embedded location platform*** delivers location-awareness for safety and productivity solutions across industrial and consumer markets. The platform consists of chips, modules and software that enable precise real-time positioning and concurrent wireless communication. The ubiquitous proliferation of such platforms is creating the location-aware Internet of Things.

Further Information

For more information about products from nanotron Technologies GmbH, contact a sales representative at the following address:

nanotron Technologies GmbH
Alt-Moabit 60
10555 Berlin, Germany
Phone: +49 30 399 954 – 0
Fax: +49 30 399 954 – 188
Email: sales@nanotron.com
Internet: www.nanotron.com