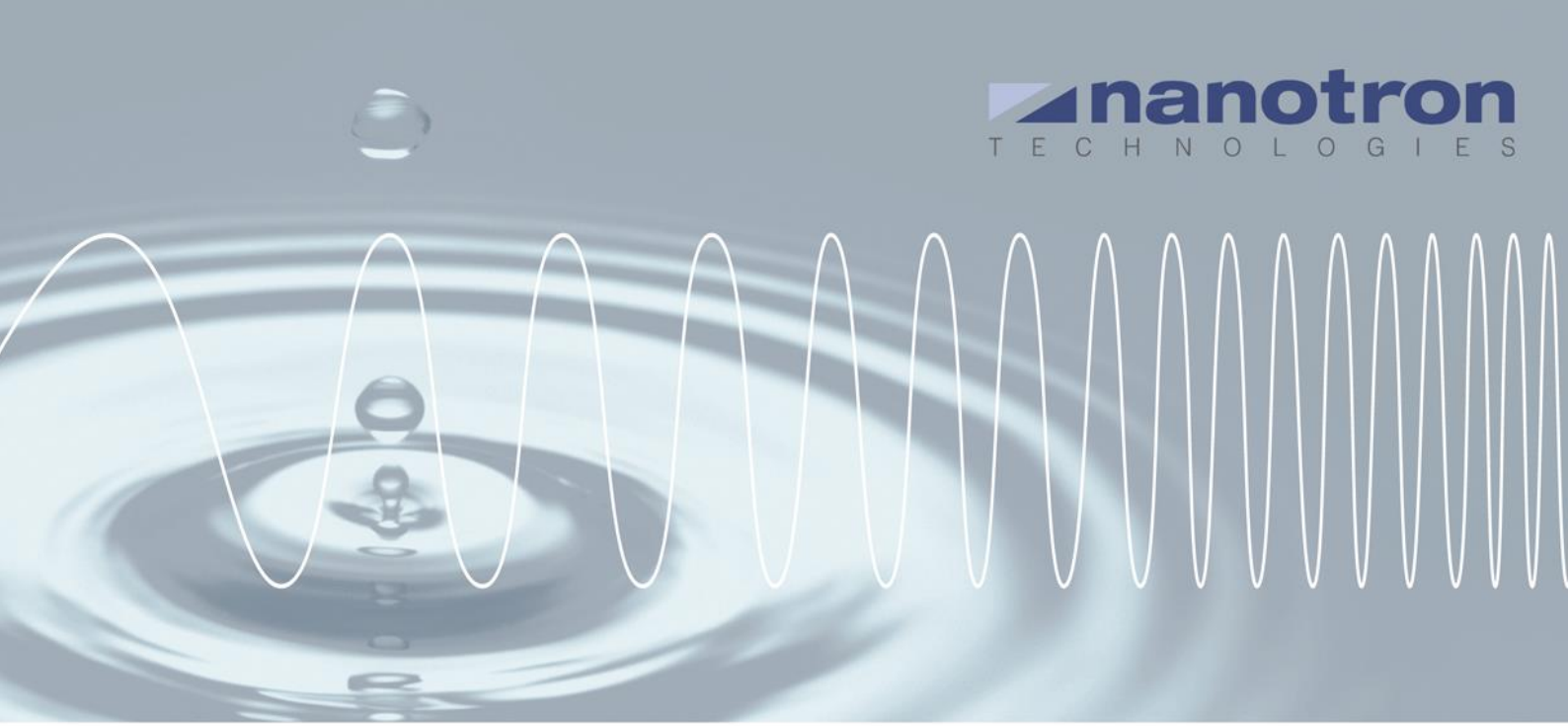


swarm API 3.0

3.0.11

NA-13-0267-0003

Document Information

Document Title:	swarm API 3.0
Document Version:	3.0.11
Current Date:	2019-04-08
Print Date:	2019-04-08
Document ID:	NA-13-0267-0003
Document Author:	MBO

Disclaimer

Nanotron Technologies GmbH believes the information contained herein is correct and accurate at the time of release. Nanotron Technologies GmbH reserves the right to make changes without further notice to the product to improve reliability, function or design. Nanotron Technologies GmbH does not assume any liability or responsibility arising out of this product, as well as any application or circuits described herein, neither does it convey any license under its patent rights.

As far as possible, significant changes to product specifications and functionality will be provided in product specific Errata sheets, or in new versions of this document. Customers are encouraged to check the Nanotron website for the most recent updates on products.

Trademarks

All trademarks, registered trademarks, and product names are the sole property of their respective owners.

This document and the information contained herein is the subject of copyright and intellectual property rights under international convention. All rights reserved. No part of this document may be reproduced, stored in a retrieval system, or transmitted in any form by any means, electronic, mechanical or optical, in whole or in part, without the prior written permission of nanotron Technologies GmbH.

Copyright © 2017 nanotron Technologies GmbH.

 © 2019 All Rights Reserved

Contents

© 2019 All Rights Reserved

 © 2019 All Rights Reserved

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.

© 2019 All Rights Reserved

2. Functionality Overview

2.1. Purpose

A swarm is a collection of devices, swarm bees, that can listen to its peers and talk to them. They can monitor their own position and that of other devices in the swarm and can also report their condition. However, the most important feature of a swarm bee is to announce its own presence to allow automatic node discovery and a flexible swarm structures. In a swarm each node operates independent from the others and dynamically adapts to the changing environment.

This chapter explains the behavior of the swarm bee module, how to manage the different power modes and how to enhance its behavior with an external host microcontroller. In addition, it gives brief recommendations on how to operate it in coexistence with other wireless devices.

2.2. Behavior of *swarm* bee

Presence announcement
Every swarm bee can periodically announce itself with a broadcast message also known as nodeID blink and thus enables the automatic discovery. The nodeID blink contains the module ID and its condition.

In order to receive the broadcast, swarm devices in the area should be listening to the same 'channel' the announcing device is talking to. This 'channel' is defined by the transmission parameters which are different for swarm bee LE (CSS) or for swarm bee ER (UWB).

Communication is only possible among swarm devices with the same transmission parameters as well as technology (LE vs. ER).

swarm bee LE transmission parameters
Table 2-1 shows the swarm bee LE parameters and the recommended use of each of those and their default setting.

Table 2-1: Transmission parameters of swarm bee LE

Parameter	Purpose	Default
Synchronization word	Separation of different populations of nodes	1
80/1 transmission mode	Highest throughput	selected
80/4 transmission mode	Longest range	-
Forward error correction (FEC)	Range extension and/or link robustness	not applied

© 2019 All Rights Reserved

The 80/1 and 80/4 transmission modes refer to the 80MHz bandwidth and the symbol duration of 1 µs or 4 µs. With 80/1 a throughput of 1Mbps can be reached. When using the mode 80/4, the maximum throughput is 250 kbps, but the achieved range is longer. The FEC option boosts both the achieved range and the link robustness, but it requires more overhead, which implies lower throughput.

swarm bee ER transmission parameters

Table 2-2: Transmission parameters of swarm bee ER

Parameter	Purpose	Default
Mode	Frequency band, data mode symbol duration	3
PAN Id		0x6e6e

Monitor its position and that of the neighbors
When a swarm device detects the blink from a second swarm device with the ID and status it reacts to it by sending an automatic ranging request. If the second device acknowledges and accepts the ranging request, both will proceed with the ranging operation. Once the device has gathered all the required information, it estimates the distance (or range) between the two of them and broadcasts it to its neighbors.

The swarm bees have control over which peer they range by using a class mechanism. Every swarm device belongs to a certain class, and can decide to which classes it sends the automatic ranging, or even if it sends ranging requests at all. To allow exceptions the swarm devices can keep a white list including the IDs of the devices to whom they will always send ranging requests, independently of the class they belong to.

In reciprocity, the swarm device receiving a ranging request may accept or refuse it. Any swarm device can be set in privacy mode and deny such requests. It may also deny such a request when it comes from devices present in its blacklist.

Table 2-3 shows the parameters determining to which other devices a swarm bee will accept or deny a ranging request.

Table 2-3: Ranging parameters

Parameter	Purpose	Default
Class	Classification in different functional groups	1
Ranging class	Classes to which the devices sends a range request	1
White list	Devices to which a ranging request is always sent	-
Black list	Devices from which a ranging request is never accepted	-
Privacy mode	None range requests is accepted	-
Range result broadcast	Inform the neighbors of the estimate distance between itself and another swarm device	enabled

Report condition
When swarm bee listens to the channel, it detects the messages broadcasted by the neighbors: blink, containing its ID, and the broadcast message containing information related to an estimated distance. Moreover, it may also contain other useful information like: sensors data, GPIO, battery status, class, power mode, etc. The amount can be set by the command NCFG. Table 2-4 shows some of the condition parameters.

Table 2-4: Condition parameters

Parameter	Purpose
Acceleration	Acceleration values in x, y and z
Status of GPIOs	Level of the 4 GPIOs
Temperature	Temperature inside de swarm bee
Battery status	Battery level at the Vcc pin
Power mode	Necessary now the communication strategy to follow
Timestamp	Instant when the blink was transmitted according to the internal clock
Class	To what class the swarm bee belongs

© 2019 All Rights Reserved

In addition, when a swarm device detects critical values of its sensors or GPIOs, it can generate an interrupt and trigger a broadcast message, similar to the blink, indicating the cause of the transmission. All the devices receiving this blink, can become aware of such a condition and act according to it.

Data transmission
A swarm bee can also communicate to other swarm devices in its area. That is, transmit or receive other data coming from a host controller connected to the swarm bee. The data transmissions can be included together with a blink, a ranging request message or can be carried out independently as a data transmission specific message, either unicast or broadcast.

The strategy used for the data transmission is important to ensure the arrival of the information to its destination, and it will depend on the condition (specially the power mode) of the recipient. For instance, if the sender (host) knows that the recipient is not active at that moment it shall store the message and delay the transmission until the recipient confirms that it is listening. The swarm bee can store only one message per recipient ID in Tx direction while it can store up to 10 messages with different IDs. Rx messages are not stored and sent immediately after reception to the host. When a message has to be sent, immediate or delayed is an option depending on the packet type which is shown in Table 2-5 .

Table 2-5: Data transmission options

Packet type		Immediate	Delayed
Data transmission packet	unicast	yes	yes
	broadcast	yes	yes
Other packets	nodeID blink	yes	no
	automatic range request	yes	no
	range request	yes	yes

When a swarm bee has to send a delayed unicast packet it waits until it receives a nodeID blink from the recipient, checks its condition to know when the device is listening and sends the message according to it. The same concept is applied to a delayed broadcast data message. The swarm bee listens for a defined period of time in which it will sent the message to each device from which it has received a nodeID blink. This is not a guaranty that all nodes have been reached. The reason can be to short time frame or nodes which are not reachable.

2.3. Operating *swarm* bee

With host control
The communication between a host controller and a swarm bee is only possible when the serial interface is enabled. This can be set by hardware with the A_MODE pin, or by the radio status of the swarm device. All possible combinations are reflected in Table 2-6. When the serial interface is active communication in both directions is possible.

Table 2-6: Combinations to enable the serial interface

A_MODE pin	Radio status	Serial interface status	Meaning
Low	Active	Enabled	Communication possible
Low	Disabled	Disable	Communication possible
High	Active	Enable	Communication possible
High	Disabled	Disable	Communication NOT possible

When a swarm bee is connected to a host controller, it has total control on all the settings and the behavior of the swarm bee. It can modify the settings of the swarm bee and monitor its condition. In addition it can request the swarm bee to notify it with all the information it gathers from the neighbors. The swarm bee will notify the host controller each time it receives a nodeID blink from another device and pass all the information contained in the message. It will also inform the host when it has a new estimated distance between two swarm devices, independently of whether it estimated it itself or received the information over the air. In both cases the status of the device that triggered the operation will also be passed to the host. Moreover, the host will be notified upon reception of a data message including its content for further processing.

Besides that the host receives information from the swarm bee, it can via commands request internal information about status, condition, mode and start a ranging or send a message to a particular swarm bee. With all those information the host can have a very complete overview of the network and adapt or change the settings of its swarm device to adjust its behavior.

© 2019 All Rights Reserved

Without host control
If no host controller is connected to a swarm bee its settings can still be changed over the air. At any time when the device is active, the swarm bee module can receive instructions; an instruction can come from any of its peers (we will call it controlling device), which does not need to be always the same. This controlling device can request the condition of the swarm device and modify its settings but it cannot request notifications about the network messages. To avoid communication loss certain instructions are locked and needs to an unlock procedure. This shall circumvent that those critical settings are done by mistake. Those parameters are to be found in Table 2-7.

Table 2-7: Locked parameters when controlling the device over the air

Parameter		Locking purpose
NodeID		The Id of the node may be lost or unknown
Transmission power		A reduction in power could mean that the device is no more in the reception area
Transmission parameters	Synchronization word (LE)	Change of those parameters can make the device inaccessible
	Transmission mode (LE)	
	FEC (LE)	
	Pan Id (ER)	
	Mode (ER)	
Parameters determining the active time of the device	Blinking period	It can cause the device to be never or very seldom in receive mode
	Reception window length	
	Reception window periodicity	
Coexistence parameters		It could make communication unfeasible

Note: For more details see section 4.4.

Mix and match
A swarm can consist of different kind of devices, those connected to a host and those working autonomously. Any host controlled swarm bee can, at a certain moment, receive instructions from its controller to temporarily become a ‘controlling’ device and send some instructions to any other swarm bee in the area.

On the other hand, any device can receive instructions from any neighbor at any moment and, in consequence, it will obey to instructions, so far not disabled, independently if it has its own host controller or not.

2.4. Power management

With regards to its power consumption a *swarm* bee module can be used either managed or un-managed mode. There are three different power modes which are controlled by hardware and software simultaneously. Table 2-8 shows the three power modes and how they should be selected.

Table 2-8: Power modes * variable depending on the configuration

Power mode	A_MODE pin	Power management	Meaning	Power-Down Mode	Consumption
0	Low	Disabled	Un-managed	None	Always active
1	Low	Enabled	Managed	Sleep	Reduced
2	High	-	Managed	Snooze	Lowest
3	Low	Enabled	Managed	Nap	Lowest*

The power mode determines whether the swarm bee can be in a low power state and in this case the state of the different components inside the swarm bee. Table 2-9 shows, for each power state, the behavior of the internal component of the swarm bee.

© 2019 All Rights Reserved

Table 2-9: Status of the component for each power state

Swarm bee power state	State of the components			
	Radio	**Serial interface**	**Sensors**	**GPIO**
Active	Active	Active	Continuously monitored	Controlled
Sleep	Disabled	Active	Continuously monitored	Controlled
Snooze	Disabled	Disabled	Not monitored	Not controlled
Nap	Disabled	Disabled	Optional	Optional

Let's explain each of the power modes individually.

Un-Managed Power Mode: power mode 0
To reach the un-managed mode also known as 'always on', the control signal A_Mode is pulled low and the software-defined power management is disabled. The device remains active with its receiver switched on. This mode of operation is useful for anchor devices or gateways with a permanent power supply for instance.

The fact that the swarm bee is always active means that: The serial interface of the swarm bee is active to allow continuous communication. Thus, in case a host controller is connected, communication is always possible. The sensors are continuously monitored and the GPIOs controlled, so that they can generate interrupts at any moment. The radio is always active; when it is not transmitting it is in receive mode, waiting to receive messages.

Managed Power Mode

Power mode 1: Active and Sleep

When the level of the A_MODE pin is low and the software-defined power management active, the swarm bee is in managed power mode alternating between active and sleep mode. During sleep mode the radio is disabled, but the serial interface, sensors and GPIOS are still active, so they can generate interrupts and communication with a host is always possible. The difference with respect to power the un-managed power mode is the behavior of the radio, which can be in three different modes: transmit, receive and disabled. Periodically, the timer generates an interrupt to indicate the swarm bee that it has to transmit a blink. After the swarm device has transmitted its nodeID blink, it can go to receive mode for a short time window. When the reception window is closed, the radio is disabled and the device goes to sleep mode until next interrupt. The swarm bee adapts the time during which it is active to minimize the power consumption. Table 2-10 lists the parameters that determine how long the swarm bee is active.

Table 2-10: Parameters determining the active time of a swarm bee in managed power mode

Parameter	Purpose	Default
Blink rate	how often the swarm bee announces itself	30 s
Rx window length	length of the reception period after the nodeID blink	30 ms
Rx window periodicity	how often the Rx period is open after the nodeID blink	after every blink

As the serial interface, sensors and GPIOs are always active, interruptions and, in consequence, transmissions can happen at any moment.

Power mode 2: Active and Snooze

When the level of A_MODE pin is high, the node is in managed power mode, no software-defined parameter is required. In this case the swarm bee can be active or in snooze mode; which consumes 1000 times less than the sleep mode. In snooze mode, the radio is disabled, the serial interface is not active, the sensors are off and the GPIOs are not controlled. This means that no communication with a host controller is possible and no interrupt is allowed other than the periodic timer controlling the nodeID blinks transmission.

When the timer generates its periodic interrupt, it triggers the swarm bee to read the value of the sensors, activate the GPIOs, the UART and send a node ID blink. Thus, the swarm device transmits its nodeID blink, goes to the receive mode for a short time window. When the reception window is closed, the radio, GPIOs and UART are disabled, so that the device goes into snooze mode again until next interrupt. The parameters defining the time that the device is active are the same ones shown in Table 2-10.

Power mode 3: Active and Nap

This power mode minimizes the power consumption, to levels similar to power mode 2 but, at the same time allows to wake up the swarm device using the GPIOs, or even the sensors. The number of components that are capable of generating an interrupt can be set; we should take into account, however, that the higher the number of components, the higher the power consumption.

© 2019 All Rights Reserved

In this mode the swarm device can be active or in nap mode; which can consume down to the same power as in power mode two. During nap mode the radio and the serial interface are disabled, the GPIO pin DIO_0 serves as wake-up signal and the other GPIOs and sensors will be controlled/monitored depending on their configuration.

Similarly to power mode 2, when the timer generates its periodic interrupt, it triggers the swarm bee to read the value of the sensors, activate the GPIOs if applicable except DIO_0 and send a node ID blink. After the swarm device transmits its nodeID blink, it can go to the receive mode for a defined time window. When the reception window is closed, the radio is disabled and the GPIOs and MEMS are falling back to the state configured for power mode 3. The swarm bee goes into the nap mode until the next interrupt. The parameters defining the time that the device is active are the same ones shown in Table 2-10.

When the GPIO pin DIO_0, or any other configured as wake-up source, is asserted, it also generates an interrupt and the swarm device is kept active until it goes low. As after any interrupt, the radio will go to transmit mode, send a nodeID blink indicating the cause of the interrupt, open the reception window according to the parameters in Table 2-10 and immediately after the interrupt is disabled. The other components, such as the serial interface, sensors and other GPIOs will be active as long as the GPIO that caused the interrupt is left high. A host controller has the possibility then to change the power mode of the swarm bee.

2.5. Coexistence

Coexistence with other swarm bees

When multiple swarm bees operate in the same area it is quite probable that at some point two of them will transmit a packet at the same time. In this case, the two packets will collide in the air and none of them will arrive to its destination. They will wait for an acknowledgment. Due to the lack of such an acknowledgment, they will retransmit their packets which may be successful or not. The more swarm devices are in the area, the more probabilities of collisions, which implies longer delays and reduce the channel capacity.

To avoid this kind of problem and to maximize the channel capacity, the swarm bee can access the channel using Carrier Sense Multiple Access with Collision Avoidance technique, CSMA/CA, in symbol detection mode. When this technique is enabled the swarm bee behaves in the following way: whenever it has to transmit, it first listens to the channel. If it detects that someone is transmitting using its same symbol (another swarm), it waits for a short predefined time and listens again. It keeps on doing so until it does not detect any other swarm device transmitting and it can proceed with its own transmission.

Coexistence with other systems

In previous subsections we considered the case in which only swarm devices are transmitting/receiving in a certain area. Nowadays, however, it would be more realistic to consider that any other wireless network can be present in the same area, for instance Wi-Fi. If such a wireless network is working in the same frequency band as the swarm bees, the 2.4 GHz ISM band, it will cause the same problem mentioned above. The higher the throughput of other systems the more collision our system will suffer. The difference with respect to the previous scenario will most probably be that other systems will use different symbols. Thus, CSMA in symbol detection will not be useful anymore.

The swarm bee can react to this scenario by using CSMA in energy detection mode. Similarly to the previous scenario, whenever the swarm bee is going to transmit, it will first listen to the channel. If the energy detected in the transmission frequency band, is higher than a predefined threshold, the swarm device will wait for a short predefined time and listen again. It will continue doing so until the energy that it detects in the transmission band is lower than the threshold. Note that, as long as it is in the same frequency band, the energy detected is independent of the kind of signal that generates it.

Note: CSMA is only applicable for swarm bee LE. Swarm bee ER has not such a mechanism.

© 2019 All Rights Reserved

3. Locating Methods

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

When working in a collaborative mode the swarm nodes are aware of the presence of other swarms devices 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, a location infrastructure is not required.

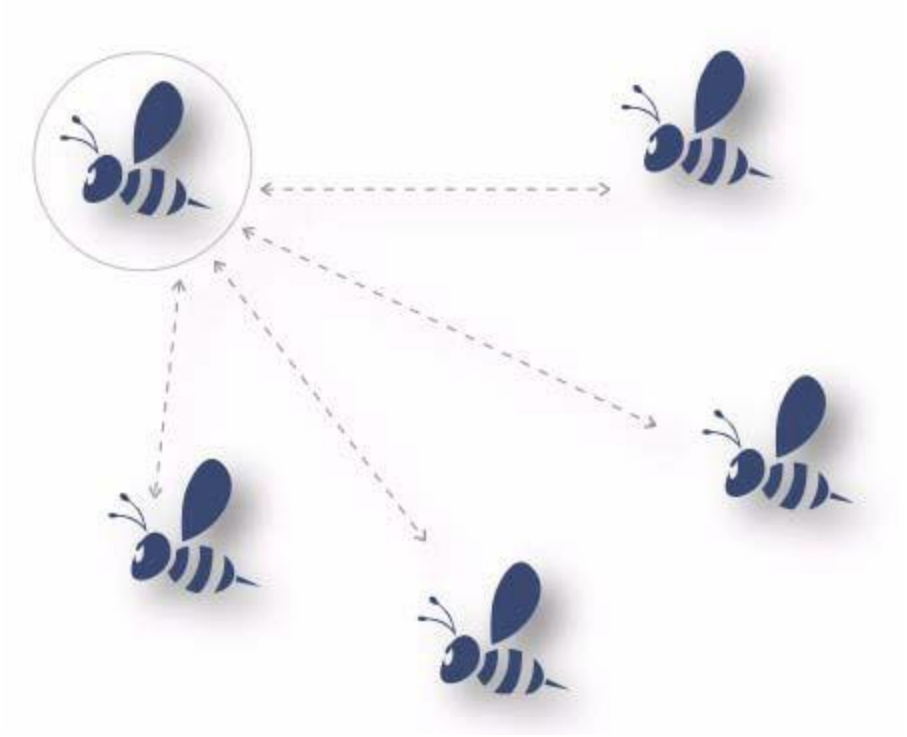


Figure 3-1 Collaborative location for mobile nodes using nanotron's swarm concept

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 a 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 the power consumption of the tag significantly. Less packets in the air per position fix allow for a larger number of objects to be tracked.

© 2019 All Rights Reserved

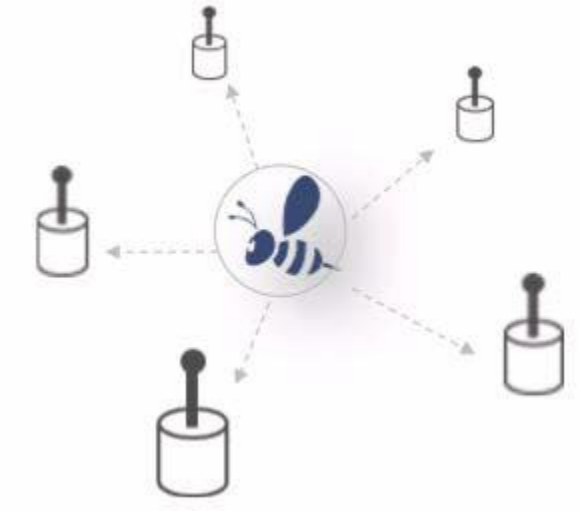


Figure 3-2 Fixed location utilizing location infrastructure with fixed radio nodes as anchors

© 2019 All Rights Reserved

4. Application Programming Interface

4.1. General Purpose

This application programming interface has been implemented so that it allows users to create their own applications, if desired. It consists of a low level API based on SDS-TWR (Symmetrical Double-Sided Two-Way Ranging) and other commands supporting data communication.

4.2. General Communication Protocols

Communication between the host controller and the swarm radio is supported by an ASCII and a BINARY protocol. These define a set of commands with corresponding parameters and 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
.
In order to reconfigure remote swarm devices over the air, an AIR protocol is also implemented. With the help of the AIR protocol a connected host can send commands to remote devices over the air.

4.2.1. ASCII protocol

For the general ASCII communication protocol the following conventions apply:

1. All communication via the interface is done by ASCII characters. This implies that, every numeric parameter, 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 0 0000BF260468\r\n"

Example (Bytes):       {0x52, 0x41, 0x54, 0x4F, 0x20, 0x30, 0x20, 0x30,
0x30, 0x30, 0x30, 0x42, 0x46, 0x32, 0x36, 0x30, 0x34, 0x36, 0x38, 0x0D,
0x0A}
```

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

```
Example: WrongCommand xyz

="ERR\r\n"
```

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

```
Example: GNID

=001122334455
```


© 2019 All Rights Reserved

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

```
Example: GRWL

#003

DDF451534C23

134683567ABC

33A441FFB311
```

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

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

4.2.2. BINARY protocol

This protocol describes the bidirectional binary interface between swarm- and host device. The information is provided in higher density format. Each transfer is done in a single frame which is described within general packet structure chapter.

4.2.2.1. *General Packet Structure*

The general packet structure is given by the following table.

Table 4-1: General packet structure

SYN	LEN	DATA	CRC_LOW	CRC_HIGH
1 byte	1 byte	1... 256 byte	1 byte	1 byte

The SYN field is used for the synchronization the data stream and by default is "0x7f". Next, the LEN field describes the length of the following DATA. The range is 0 .. 255, 0 is representing 256 byte of DATA. It is possible that the SYN value or "0x7F" occurs in the data, so in order to prevent false detection of packet starting with this value, it must be modified to an escape value. The DATA field contains the user data/payload. To escape SYN in data field use the Escape sign ESC = "0x1b". However, the ESC value or "0x1b" may also appear in the data. To prevent false detection of ESC it uses another modified value "0x1b45". Before transmitting from host to swarm device, each message must be checked for SYN or ESC signs. Every discovered sign must be escaped beside the initial SYN. After receiving a message from a swarm device, it must be de-escaped.

To escape a SYN:	ESC + 'S'
	0x1b53
To escape an ESC:	ESC + 'E'
	0x1b45

Example1: Sending SMBW 2 command in binary mode:

1. 0x7f03555402ba9f => nothing to escape, no SYN (0x7f) or ESC (0x1b) spotted after initial SYN.
2. The answer to this request from swarm to host is: 0x7f035754021b455f. This is already escaped and must be de-escaped after reception: 0x7f0355540**1b**5f.

Example2: Sending GRWL command in binary mode:

1. 0x7f025415471b => last byte is 0x1b and must be escaped to: 0x7f0254154**1b45**
2. The answer to this request, if the list is empty, is: 0x7f03561500fb0e => nothing to de-escape.

The cyclic redundancy check (CRC) detects accidental changes to raw data. The swarm implementation uses CRC-16-ansi reversed, which is x^16 + x^14 + x^1 + 1 (0xA001, Koopman notation) and is initialized

© 2019 All Rights Reserved

with 0. The CRC check will be applied to SYN, LEN and DATA field. It only considers the original sign, not the escape sequences.

Example: ESC + 'S' reflects a 0x7f and an ESC + 'E' a 0x1b, therefore the 0x7f or 0x1b is used for calculating the CRC. Not the ESC + 'S' or ESC + 'E' itself.

> **Note**: The CRC is transmitted LSB first. Ensure that order for CRC before transmitting.
>
> CRC_LOW = second byte of CRC result
>
> CRC_HIGH = first byte of CRC result

4.2.2.2. *DATA Structure*

The DATA consists of a TYPE, CMD opcode and a CMD_DATA field. The TYPE describes the kind of interaction with the swarm module and the CMD, the chosen command with the corresponding command data (CMD_DATA) e.g. parameters.

Table 4-2: Data Structure

TYPE	CMD	CMD_DATA
1 byte	1 byte	0 .. 254 byte

4.2.2.3. *TYPE*

The command types are divided into two groups. Commands which are sent to the swarm devices (GET, SET) and command types which are included into a response or message of a swarm device (RESP, ERR, NOTI).

Table 4-3: Type field definitions

TYPE	Value	description
GET	0x54	Get command
SET	0x55	Set command
G_RESP	0x56	Get response
S_RESP	0x57	Set response
ERR	0x60	Error
NOTI	0x61	Asynchronous notification

TYPE GET must be used to read from swarm device. If something goes wrong, ERR is returned.
Example: GNID (get node id) request from host to swarm:

Table 4-4: GET example request

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x54	0x00	0x86	0xd4

Note: CMD_DATA doesn't exist for GNID, because this command doesn't have parameters, other commands may have this field.

GNID response from swarm to host:

Table 4-5: Get example response TYPE **G_RESP**

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x08	0x56	0x00	0x0000b6f31103	0x4a	0xe7

TYPE SET must be used to write to swarm device. If something goes wrong, ERR is returned.

Example: SNID (set node id) request from host to swarm:

Table 4-6: SET example request

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x08	0x55	0x00	0x0000b6f31103	0x0a	0xf2

SNID response from swarm to host with OK:

Table 4-7: SET example response TYPE S_RESP

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x08	0x57	0x00	0x0000b6f31103	0x8b	0x2b

© 2019 All Rights Reserved

TYPE ERR indicates that something went wrong. Following errors may occur:

ERR_CRC = 0x01 CRC is wrong
ERR_CMD_UNKNOWN = 0x02 unknown command
ERR_PARAMETER = 0x03 wrong parameter for command (too many / too few parameters, or range of values violated)
ERR_BUFFER_OVERFLOW = 0x04 data did not fit into reception buffer
ERR_GARBAGE = 0x06 unexpected sign during frame reception
ERR_TIMEOUT = 0x07 incomplete packet (timeout 5ms)
ERR_LOCKED = 0x08 parameter is locked and must unlocked first.
ERR_BLOCKED = 0x09 parameter is blocked and cannot be changed
API_NOT_SUPPORTED = 0x10 not supported by API

In any response with the TYPE ERR the CMD field does not contain the opcode instead it contains the corresponding error code, which is one of the values listed above.

Example: ERR because CMD unknown:

Table 4-8: ERR example request

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x60	0x02	0x95	0xad

TYPE NOTI occurs within asynchronous communication messages. Several notification messages exist. To separate the several notifications following types are defined:

Data Notification Message DNO (0x60)

Node ID Notification Message NIN (0x61)

Ranging Result Notification Message RRN (0x62)

SDAT Notification Messages SDAT (0x63)

AIR Notification Message AIR (0x64)

The CMD field holds the information which type of notifications was transmitted. Depending on the type of the notification the CMD_DATA field is set up.

Example: NOTI because of Node ID Notification Message (Parameter is a node ID)

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x08	0x61	0x61	0x0000b6f31103	0x78	0xd3

4.2.3. AIR protocol

This protocol describes the bidirectional AIR interface. This allows reconfiguration of remote swarm devices through another swarm device using the SDAT command, where the parameter (or message sent) is the AIR packet. The information is provided the same way as it is done by the binary interface. Therefore an AIR protocol packet can be easily created using the binary description. SDAT always returns whether the packet reached the destination or not. In addition to that, a response packet from remote swarm device is generated. The response is generated right before the change takes effect. If the command is not known to the device, an error packet is generated.

Note: In autonomous mode, changed settings must be saved immediately. Otherwise they are lost, with the next blink, because all settings will be restored from EEPROM after each wake-up. Therefore send a streaming start (SSTART) to enable the receiver for a certain time on the remote device. Now change settings with the corresponding commands. Finish it with save settings (SSET) and stop streaming (SSTOP).

Note: When sending random data it may happen to switch the remote swarm device unintentionally to the AIR protocol. This will happen if the payload of the SDAT command starts with 0x08125554 which corresponds to the P_TYPE of the AIR protocol. See below. Either the sending device takes care not to send this sequence at the beginning of frame or we recommend to prepend 0x08125566 before any random payload.
Alternatively it is also possible to use the EAIR command to disable over the air configuration completely. Once disabled the four first bytes of the payload can be random. This command can only be used via the UART interface. Refer to section 5.1.13.

© 2019 All Rights Reserved

4.2.3.1. *General Packet Structure*

The general packet structure is given by the following table.

Table 4-9: General packet structure

P_TYPE	P_VERSION	C_TYPE	RESERVED	C_LEN	C_OPCODE	DATA
4 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 … 119 bytes

P_TYPE = Protocol identifier = **0x08125554**

P_VERSION = Protocol version = **0x02**

C_TYPE = Command type =

Table 4-10: Type field definitions

C_TYPE	Value	description
GET	0x54	Get command
SET	0x55	Set command
G_RESP	0x56	Get response
S_RESP	0x57	Set response
ERR	0x60	Error

C_LEN = Payload length (all bytes after C_LEN until end of packet without CRC or binary LEN minus one)

C_OPCODE = Command opcode (same as binary command opcode)

DATA = Contains OPCODE specific data and is described in the command description. (Same as binary parameter description)

4.2.3.2. *ERR*

Table 4-11: Error packet structure

P_TYPE	P_VERSION	C_TYPE	RESERVED	C_LEN	C_OPCODE	DATA
4 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte

TYPE ERR (0x60 in C_TYPE) indicates that something went wrong. Following errors may occur:

ERR_CRC = 0x01 CRC is wrong
ERR_CMD_UNKNOWN = 0x02 unknown command
ERR_PARAMETER = 0x03 wrong parameter for command (too many / too few parameters, or range of values violated)
ERR_BUFFER_OVERFLOW = 0x04 data did not fit into reception buffer
ERR_GARBAGE = 0x06 unexpected sign during frame reception
ERR_TIMEOUT = 0x07 incomplete packet (timeout 5ms)
ERR_LOCKED = 0x08 parameter is locked and must unlocked first.
ERR_BLOCKED = 0x09 parameter is blocked and cannot be changed
API_NOT_SUPPORTED = 0x10 not supported by API

How to assemble an air packet for reconfiguration of swarm nodes

To reconfigure a certain setting an over the air packet must be assembled. The protocol type P_TYPE (0x08125554) and version P_VERSION (0x02) are always the same.

=> **0x0812555402**

Next part of protocol is the command type C_TYPE, which is either GET (0x54) or SET (0x55). - Table 4-10: Type field definitions.

=> 0x08125554025**5**

Followed by a reserved byte.

=> 0x0812555402550**0**

Followed by the length C_LEN, which depends on the command itself. For changing the node ID blink interval the command SBIV is used. This command has one parameter with length of 2 bytes. The one byte command opcode counts also into the length. This information can be found in the binary description of each command.

=> 0x0812555402550003

© 2019 All Rights Reserved

Next to C_LEN field is the opcode field (C_OPCODE). SBIV opcode is 0x31. Opcode information can be found at sect. 4.3.7 Command Opcode Overview.

=> 0x081255540255000331

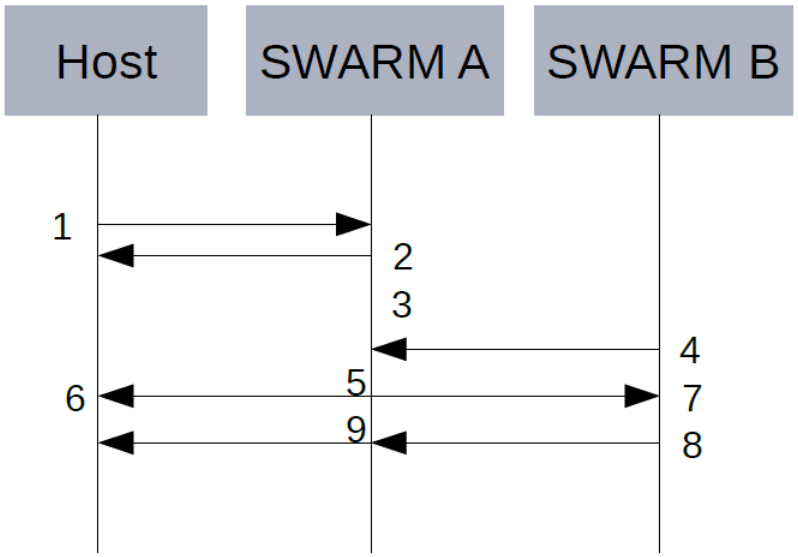
After C_OPCODE the command specific parameter must be placed. In this case it has only 2 bytes, which configures the new blink interval. It is transmitted MSB first. See command parameter description for detailed information about range and order of each parameter. One second blink interval is 0x03e8.

=> 0x0812555402550003310 3e8

This can now be transmitted to any *swarm* device to reconfigure its blink interval. This can be done with SDAT command, or any other device which is able to send nanoLOC packets.

© 2019 All Rights Reserved

Example GTXP (get tx power) on ASCII interface using SDAT



1. Host transmits a SDAT command with GTXP (get tx power) as content:
   SDAT <option> <ID> <len> **<data>** <timeout>
   SDAT 1 000000000011 9 **081255540254000105** 60000
   Data part which is transmitted over air:

P_TYPE	P_VERSION	C_TYPE	RESERVED	C_LEN	C_OPCODE	DATA
0x08125554	0x02	0x54	0x00	0x01	0x05	-

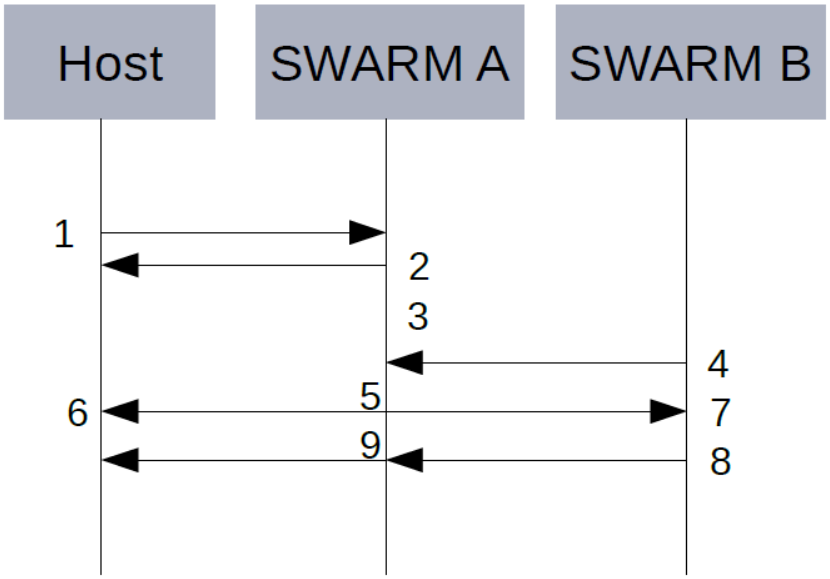
2. swarm A sends back payload ID to host.
   =575090200
3. Wait for next node ID blink with RX slot open of swarm B device with ID (000000000011)
4. Node ID blink broadcast from swarm B.
5. swarm A transmits the prepared payload to swarm B (081255540254000105) and sends error code after transmission to Host.
6. Host receives asynchronous *SDAT response. No error occurred.
   *SDAT:000000000011,0,575090200
7. swarm B receives the packet, and identifies the AIR protocol through P_TYPE and P_VERSION. Reads the C_TYPE and C_OPCODE which is set to GET and GTXP.
8. swarm B sends a GTXP response packet back to swarm A over air.

P_TYPE	P_VERSION	C_TYPE	RESERVED	C_LEN	C_OPCODE	DATA
0x08125554	0x02	0x56	0x00	0x02	0x05	0x3f

9. swarm A receives the response packet, and generates an *AIR notification at Host.
   *AIR:000000000011,05,56,3f

© 2019 All Rights Reserved

Example GTXP (get tx power) on BINARY interface using SDAT



1. Host transmits a SDAT command with GTXP (get tx power) as content:

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x15	0x55	0x21	0x01000000000011090812555 40254000105ea60	0x75	0xe1

SDAT command contains <Option>, <ID>, <Len>, <Data>, <TimeOut>.
In this case, <Option> is set to 1, which means it must wait for next rx slot of *swarm* B. Destination <ID> is 0x000000000011. The <Timeout> is 60000ms (0xea60).
<Data> is the part, which is transmitted over air:

P_TYPE	P_VERSION	C_TYPE	RESERVED	C_LEN	C_OPCODE
0x08125554	0x02	0x54	0x00	0x01	0x05

2. *swarm* A sends back the payload ID (0x0e4a0a96)

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x06	0x57	0x21	0x22472e18	0x37	0x43

3. Wait for next node ID blink with RX slot open of *swarm* B device with ID (000000000011)
4. Node ID blink broadcast from *swarm* B.
5. SWARM A transmit the prepared payload to *swarm* B (081255540254000105) and sends error code after transmission to Host.
6. Host receives asynchronous *SDAT response. No error occurred

.

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x0d	0x61	0x63	0x0000000000110022472e18	0x90	0xce

7. *swarm* B receives the packet, and identifies the AIR protocol through P_TYPE and P_VERSION. Reads the C_TYPE and C_OPCODE which is set to GET and GTXP.
8. *swarm* B sends a GTXP response packet back to *swarm* A over air.

P_TYPE	P_VERSION	C_TYPE	RESERVED	C_LEN	C_OPCODE	DATA
0x08125554	0x02	0x56	0x00	0x02	0x05	0x3f

9. *swarm* A receives the response packet, and generates an *AIR notification at Host.

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x0c	0x61	0x64	0x00000000001105563f0132	0x65	0x9b

© 2019 All Rights Reserved

4.3. UART API Command Set Overview

This chapter summarizes and categorizes the API Command Set:

4.3.1. *swarm* radio Setup Commands

SNID (0x00) **S**ets the **N**ode **ID** of *swarm* node
GNID (0x00) **G**et **N**ode **ID** of the node connected to host
SSET (0x01) **S**ave **SET**tings
RSET (0x02) **R**estore **SET**tings from EEPROM (node configuration)
GSET (n.a.) **G**et current **SET**tings (node configuration)
SFAC (0x03) **S**et node configuration to **FAC**tory default settings
SPSA (0x04) **S**et **P**ower **S**aving **A**ctive
BLDR (0x07) **B**oot**L**oa**D**e**R**
SBIN (n.a.) **S**et/**S**elect **BIN**ARY mode/interface
GFWV (0x08) **G**et the **F**irm**W**are **V**ersion of the swarm Node
GUID (0x09) **G**et **U**nique **ID** of swarm Node
SUAS (0x0a) **S**et **UA**RT **S**peed
EAIR (0x0c) **E**nable **AIR** interface

4.3.2. *Ranging Commands*

EPRI (0x10) **E**nable **PRI**vacy mode
SPBL (0x11) **S**ets **P**rivacy **B**lack**L**ist
GPBL (0x11) **G**et **P**rivacy **B**lack**L**ist
RATO (0x12) **RA**nge **TO**
BRAR (0x13) **B**roadcast **RA**nging **R**esults
SROB (0x14) **S**elects **R**anging **O**peration **B**links
SRWL (0x15) **S**ets **R**ange **W**hite **L**ist
GRWL (0x15) **G**et **R**anging **W**hite **L**ist
ERRN (0x16) **E**nables (or disables) **R**anging **R**esult **N**otification
SROF (0x17) **S**ets **R**ange **OF**fset (in ms)

4.3.3. *Data Communication Commands*

EDAN (0x20) **E**nables and disables **DA**ta **N**otification
EDNI (0x2B) **E**nables and disables user **D**ata **N**otification of Node **I**D broadcast
SDAT (0x21) **S**ends **DAT**a to node ID
GDAT (0x27) **G**ets received **DAT**a
BDAT (0x22) **B**roadcasts **DAT**a
FNIN (0x28) **F**ill data into **N**ode **I**D **N**otification packets
FRAD (0x2A) **F**ills the **RA**nging data buffer
EIDN (0x26) **E**nables and disables Node **ID** Broadcast **N**otification

4.3.4. *swarm radio Node Identification*

EBID (0x30) **E**nable **B**roadcast **ID**
SBIV (0x31) **S**ets the **B**roadcast **I**nterval **V**alue (or blinking rate)

© 2019 All Rights Reserved

NCFG (0x32) **N**otification **C**on**F**i**G**uration

4.3.5. *Medium Access Commands*

4.3.5.1. *Common Commands*

SRXW (0x40) **S**ets reception, **RX W**indow

SRXO (0x41) **S**ets **RX** Window **O**ccurrence

SDCL (0x42) **S**ets the **D**evice **CL**ass of the node

4.3.5.2. *swarm bee LE specific Commands*

STXP (0x05) **S**ets transmission (**TX**) **P**ower of the node

SSYC (0x06) **S**et the PHY **SY**n**C**word of *swarm* node

SFEC (0x43) **S**witches **F**orward **E**rror **C**orrection (FEC) on / off

SDAM (0x44) **S**ets **DA**ta **M**ode

CSMA (0x45) Switches **CSMA** mode on and off and determines back-off factor for CSMA

4.3.5.3. *swarm bee ER specific Commands*

STPD (0x70) **S**ets the node's **T**ransmission **P**ower and **D**isable/enable the smart power feature

SDMD (0x71) **S**ets the node's data mode

SDMC (0x72) **S**ets the custom data mode.

SOFF (0x75) **S**ets the node's ranging distance **OFF**set

GOFF (0x75) **G**et the node's ranging distance **OFF**set values

SPAN (0x0E) **S**ets the **PAN** ID of swarm bee ER

4.3.6. *Sensors Commands*

EMSS (0x50) **E**nables the **M**EMS **S**ensor

EBMS (0x51) **E**nable **B**roadcast **M**EM**S** within Node ID Blink packets

SMRA (0x52) **S**ets **M**EMS' **RA**nge, i.e. +/-2,4,8 or 16g

SMTH (0x53) **S**ets the **M**EMS' **TH**reshold for the slope interrupt.

SMBW (0x54) **S**ets **M**EMS' filter **B**and**W**idth of the MEMS

SMSL (0x55) **S**ets **M**EMS' **SL**eep time of the sensor

SMDT (0x56) **S**ets **M**EMS' **D**ead**T**ime

GMYA (0x57) **G**ets **MY A**cceleration

GMYT (0x58) **G**ets **MY T**emperature

GBAT (0x59) **G**ets **BAT**tery Voltage

GPIO (0x5A) Configure **GPIO** pins.

SPIN (0x5B) **S**ets GPIO **PIN**s.

GPIN (0x5B) **G**ets GPIO **PIN** status

ICFG (0x5C) **I**nterrupt **C**on**F**i**G**uration

SMAI (0x5D) **S**ets **M**EMS' **A**lternative blink **I**nterval.

SADC (0x5E) **S**ets **ADC** voltage divider parameters

GADC (0x5E) **G**ets **ADC** computed voltage

4.3.7. *AIR Interface Commands*

SSTART (0x23) **S**treaming **START**

© 2019 All Rights Reserved

SEXTEND (0x24) **S**treaming **EXTEND**

SSTOP (0x25) **S**treaming **STOP**

© 2019 All Rights Reserved

4.4. Command OP_CODE Overview

CMD			ASCII		BINARY		AIR	
NAME	OP_CODE	Relevance	GET	SET	GET	SET	GET	SET
snid	0x00	Common		x	x	x	x	locked
gnid	0x00	Common	x		x	x	x	locked
sset	0x01	Common		x		x		x
rset	0x02	Common		x		x		locked
gset	NA	Common	x					
sfac	0x03	Common		x		x		locked
spsa	0x04	Common		x	x	x	x	x
stxp	0x05	SB LE		x	x	x	x	locked
ssyc	0x06	SB LE		x	x	x	x	locked
bldr	0x07	Common		x		x		
sbin	NA	Common		x				
gfwv	0x08	Common	x		x		x	
guid	0x09	Common	x		x		x	
suas	0x0a	Common		x	x	x	x	
eair	0x0c	Common		x	x	x		
span	0x0e	SB ER		x	x	x	x	locked
epri	0x10	Common		x	x	x	x	x
spbl	0x11	Common		x	x	x	x	x
gpbl	0x11	Common	x		x	x	x	x
rato	0x12	Common		x		x		
brar	0x13	Common		x	x	x	x	x
srob	0x14	Common		x	x	x	x	x
srwl	0x15	Common		x	x	x	x	x
grwl	0x15	Common	x		x	x	x	x
errn	0x16	Common		x	x	x	x	x
srof	0x17	Common		x	x	x	x	x
edan	0x20	Common		x	x	x	x	x
edni	0x2b	Common		x	x	x	x	x
sdat	0x21	Common		x		x		
bdat	0x22	Common		x		x		
sstart	0x23	Common						x
sextend	0x24	Common						x
sstop	0x25	Common						x
gdat	0x27	Common	x		x			
fnin	0x28	Common		x	x	x	x	x
frad	0x2a	Common		x	x	x	x	x
eidn	0x26	Common		x	x	x	x	x
ebid	0x30	Common		x	x	x	x	locked
sbiv	0x31	Common		x	x	x	x	x*
ncfg	0x32	Common		x	x	x	x	x
srxw	0x40	Common		x	x	x	x	locked

© 2019 All Rights Reserved

CMD			ASCII		BINARY		AIR	
srxo	0x41	Common		x	x	x	x	locked
NAME	**OP_CODE**	**Relevance**	**GET**	**SET**	**GET**	**SET**	**GET**	**SET**
sdcl	0x42	Common		x	x	x	x	x
sfec	0x43	SB LE		x	x	x	x	locked
sdam	0x44	SB LE		x	x	x	x	locked
csma	0x45	SB LE		x	x	x	x	locked
emss	0x50	Common		x	x	x	x	x
ebms	0x51	Common		x	x	x	x	x
smra	0x52	Common		x	x	x	x	x
smth	0x53	Common		x	x	x	x	x
smbw	0x54	Common		x	x	x	x	x
smsl	0x55	Common		x	x	x	x	x
smdt	0x56	Common		x	x	x	x	x
gmya	0x57	Common		x	x	x	x	
gmyt	0x58	Common		x	x	x	x	
gbat	0x59	Common		x	x	x	x	
gpio	0x5a	Common		x	x	x	x	x
spin	0x5b	Common		x	x	x	x	x
gpin	0x5b	Common	x		x	x	x	x
icfg	0x5c	Common		x	x	x	x	x
smai	0x5d	Common		x	x	x	x	x
sadc	0x5e	Common		x	x	x	x	x
gadc	0x5e	Common	x		x	x	x	x
stpd	0x70	SB ER		x	x	x	x	locked
sdmd	0x71	SB ER		x	x	x	x	locked
sdmc	0x72	SB ER		x	x	x	x	locked
soff	0x75	SB ER		x	x	x	x	x
goff	0x75	SB ER	x		x	x	x	x

* SBIV command needs to be unlocked first, if new interval is less than 10ms.

“Common” stands for commands to both swarm bee LE and swarm bee ER
“SB LE” stands for swarm bee LE specific commands
“SB ER” stands for swarm bee ER specific commands

In the table above an overview about the get or set capability of each command is given. To avoid changing some command settings by mistake and losing access to the remote node, a locking mechanism is introduced. “locked” stands for not changeable without unlocking first. Locked commands may not be changed because it would disrupt air communication. If you really need to change locked settings via the AIR interface, please contact nanotron support and they will provide you with the necessary information.

© 2019 All Rights Reserved

5. UART API Command Set

In order to interact with the embedded ranging hardware platform the following API command set is implemented:

5.1. *swarm* radio Setup Commands

5.1.1. SNID

SNID <ID>:

Sets the Node ID of *swarm* node to **<ID>**. Besides the default Node ID which is set during production a customized ID can be set to adapt it to the needs of the target system.

Parameter Description:

<ID> 000000000000 is not a valid address but resets the original unique Node ID which was set during device production if supported by µC otherwise: 000000000001

Range: 000000000000 … FFFFFFFFFFFE

Return Value Description: **<ID>**

<ID> configured 6 byte Node ID of swarm node if set ID = 000000000000 then default ID is returned

Range: 000000000000 … FFFFFFFFFFFE

Important Note: For ranging it is possible to use the swarm bees with the above mentioned range. However, to guarantee a correct functioning with nanoLES 3 RTLS the address range should be within:

Range: 000000000000 ... 0000EFFFFFFE

ASCII

Parameter (Format):

<ID> 12 bytes (hex)

```
Example: SNID 0000BF260468
```

Return value (Format):
<ID> 12 bytes (hex)

```
Example: =0000BF260468
```

BINARY

Type: SET

Parameter (Format):

<ID> 6 bytes (uint8_t [6])

```
Example: SET SNID 0000b6f31103
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x08	0x55	0x00	0x0000b6f31103	0x0a	0xf2

Return value (Format):

<ID> 6 bytes (uint8_t [6])

© 2019 All Rights Reserved

```
Example: RESP SNID 0000b6f31103
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x08	0x57	0x00	0x0000b6f31103	0x8b	0x2b

5.1.2. GNID

GNID <void>:

Readout the configured Node ID.

Parameter description: <void>

Return value description: **=<ID>**

<ID> configured 6 byte Node ID of *swarm* node
Range: 000000000001 … FFFFFFFFFFFE

ASCII

Parameter (Format):
<ID> 12 bytes (hex)

```
Example: GNID
```

Return value (Format):
<ID> 12 bytes (hex)

```
Example: =0000BF260468
```

BINARY

Type: GET
Parameter (Format):
none

```
Example: GET GNID
```

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x54	0x00	0x86	0xd4

Return value (Format):
<ID> 6 bytes (uint8_t [6])

```
Example: RESP GNID 0000b6f31103
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x08	0x56	0x00	0x0000b6f31103	0x4a	0xe7

© 2019 All Rights Reserved

5.1.3. SSET

SSET <void>:

Saves all settings including Node ID permanently to EEPROM. This must be done after changing settings if they should persist through a power cycle or switch off state.

Parameter description: <void>

Return value description: **=<ErrorCode>**

<ErrorCode> Result of saving operation

Range: 0 … 1

0 = Saving of all parameters successfully verified

1 = Saving of parameters not successful; verification failed

ASCII

Example: SSET

Return value (Format):

<ErrorCode> 1 byte (dec)

```
Example: =0
```

BINARY

Type: SET

```
Example: SET SSET
```

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x55	0x01	0x46	0x84

Return value (Format):

<ErrorCode> 1 byte (uint8_t)

```
Example: RESP SSET 0
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x01	0x00	0xa5	0xce

5.1.4. RSET

RSET <void>:

Restores all parameter settings from EEPROM. In contrast to the factory setting this command restores all the parameters which have been saved since the last SSET. Ref. to 5.1.3.

Parameter description:
<void>

Return value description: **=<ErrorCode>**
<ErrorCode> Result of restoring operation

Range: 0 … 1

© 2019 All Rights Reserved

0 = Restoring of all parameters successful

1 = Restoring parameters from EEPROM failed

ASCII

```
Example: RSET
```

Return value (Format):

<ErrorCode> 1 byte (dec)

Example: =0

BINARY

Type: SET

```
Example: SET RSET
```

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x55	0x02	0x06	0x85

Return value:

<ErrorCode> 1 byte (uint8_t)

```
Example: RESP RSET
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x02	0x00	0xa5	0x3e

5.1.5. GSET

GSET <void>:

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

Note: This command generates a response which has a variable number of lines. It is recommended to use a parser as in future the number of lines and the parameters may change. Refer also to sect. 10, point 6. See also differences between APIs sect. 10, point 6.

Parameter description:

<void>

Return value description: #**<NumLines>**\r\n**<ParameterName>**:**<Value>**\r\n ..

<NumLines> Number of lines after this line

Range: 000 … 255

<ParameterName> Name of following parameter value.

<Value> Depends on parameter.

ASCII

```
Example: GSET
```


© 2019 All Rights Reserved

Return value (Format):

<NumLines> 4 bytes (dec), first byte fixed „**#**“

<ParameterName> ASCII

<Value> depends on parameter

```
Example for swarm bee LE:
          #036
          BRAR:1
          CSMA:1,0,0
          EAIR:1
          EBID:1
          EBMS:1
          EDAN:1
          EDNI:0
          EIDN:0
          EMSS:1
          EPRI:0
          ERRN:1
          GIO0:0,3,0,1
          GIO1:1,3,0,1
          GIO2:2
          GIO3:0,3,0,1
          ICFG:0
          NCFG:0004
          SBIV:30000
          SDAM:1
          SDCL:1
          SFEC:0
          SMAI:100,1,30000
          SMBW:6
          SMDT:01000
          SMRA:2
          SMSL:01
          SMTH:30
          SNID:XXXXXXXXXXXX
          SPSA:0
          SROB:01
          SROF:000
          SRXO:001
          SRXW:00010
          SSYC:1
          STXP:63
          SADC:2700,2200
```

BINARY

Type: Not supported, use the get command for each setting.

5.1.6. SFAC

SFAC <void>:

Resets device configuration to factory settings. Default configuration is described in chapter 8 Default Settings.

Note: Settings not saved to EEPROM until SSET command is issued.

© 2019 All Rights Reserved

Parameter description:

<void>

Return value description: **=<ErrorCode>**

<ErrorCode> Result of configuration**.**

Range: 0…1

0 = Successful.

1 = Not successful.

ASCII

```
Example: SFAC
```

Return value (Format):

<ErrorCode> 1 byte (dec)

```
Example: =0
```

BINARY

Type: SET

```
Example: SET SFAC
```

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x55	0x03	0xc7	0x45

Return value (Format):

<ErrorCode> 1 byte (uint8_t)

```
Example: RESP SFAC 0
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x03	0x00	0xa4	0xae

5.1.7. SPSA

SPSA <Type>:

Sets power management mode for *swarm* radio. This command allows to set the power consumption to the needs of the application. The effect is immediate, but depends on the current state and settings. It is recommended to read the application note AN513 [6] to get an understanding of the different modes and their corresponding settings.

Note: When setting the device to power mode 3, it is important that previously at least 1 of the GPIO pins have been configured as wake-up pin. Otherwise the communication through the UART can be permanently blocked. For example, if all four GPIOs are used as output, only over the AIR configuration is possible.

© 2019 All Rights Reserved

Parameter description:

<Type>

Range: 0, 1 and 3

0 = Power saving off, device is continuously receiving

1 = Power saving on, wake up on any interrupt

3 = Dynamic power saving on, wake up depending on the settings of ICFG.

Note: Power mode 2 can only be set by hardware. See sect. 2.4.

Return value description: **=<Type>**

ASCII

Parameter (Format):

<Type> 1 byte (dec)

```
Example: SPSA 1
```

Return value (Format):

<Type> 1 byte (dec)

```
Example: =1
```

BINARY

Type: GET/SET

Parameter (Format):

<Type> 1 byte (uint8_t)

```
Example: SET SPSA 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x04	0x01	0xc6	0x9e

Return value (Format):

<Type> 1 byte (uint8_t)

```
Example: RESP SPSA 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x04	0x01	0x67	0x5e

5.1.8. BLDR

BLDR <void>:

Restarts the device and start bootloader.

Parameter description:

<void>

Return value description: **=0**

© 2019 All Rights Reserved

ASCII

```
Example: BLDR
```

Return value (Format):

<> 1 byte (dec)

```
Example: =0
```

BINARY

Type: SET

```
Example: SET BLDR
```

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x55	0x07	0xc6	0x86

Return value (Format):

NO PARAMETER

```
Example: RESP BLDR
```

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x57	0x07	0xc7	0xe6

5.1.9. SBIN

SBIN <void>:

This command switches the interface from ASCII to BINARY.

When the mode is switched to binary the swarm device acknowledges it in ASCII format before changing to binary mode. The selected interface isn't stored; thus when the module resets, it starts again in ASCII mode. Consequently, to return back to ASCII mode the RESET should be triggered.

Note: The SBIN command is not supported in the BINARY Interface

ASCII

```
Example: SBIN
```

Return value (Format):

<> 1 byte (dec)

```
Example: =0
```

BINARY

Type: NOT SUPPORTED

5.1.10. GFWV

GFWV <void>:

Returns the firmware version of the swarm Node.

© 2019 All Rights Reserved

Parameter description:

<void>

Return values: **=<FirmwareVersion>**

<FirmwareVersion> Firmware version of the swarm node. The version consists of 5 parts. Major number, minor number, sub-minor, release candidate number, number of changes after last tag. Example: ver2.0.3-rc0-98 is translated to 0x02003062.

Range: major[0 … 0xFF]
minor[0 ... 0xFF]
sub-minor[0 … 0xF]
release candidate[0 … 0xF] (F stands for final release version)
changes after last tag [0 .. 0xFF]

ASCII

```
Example: GFWV
```

Return value (Format):

<FirmwareVersion> 8 byte (hex)

```
Example: =02003062
```

BINARY

Type: GET

```
Example: GET GFWV
```

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x54	0x08	0x87	0x12

Return value (Format):

<FirmwareVersion> 4 byte (uint8_t)

```
Example: RESP GFWV
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x06	0x56	0x08	0x02003500	0xda	0x7b

5.1.11. GUID

GUID <void>:

Description: Read the unique ID of µC.

© 2019 All Rights Reserved

Return value description: **=<UID>**

<UID>

Range: 0 … FFFFFFFFFFFFFFFFFFFFFFFF

ASCII

```
Example: GUID
```

Return value (Format):

<UID> 24 byte (hex)

```
Example: =353347083138373128003200
```

BINARY

Type: GET

Parameter (Format):

<void>

```
Example: GET UID
```

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x54	0x09	0x46	0xd2

Return value (Format):

<UID> 12 byte (uint8_t)

```
Example: RESP GUID 353347083138373128003200
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x0e	0x56	0x09	0x353347083138373128003200	0x5a	0xb8

5.1.12. SUAS

SUAS <Speed>:

Sets the UART bit rate.

Note 1: It may happen that the desired <Speed> is not reachable. Therefore the closest possible value will be configured and returned. The error will be <1% of the desired value. For example 115200 bps translates to 115108 bps. This is 0.08% Error.

The UART speed can only be set when the communication is over the UART itself, not over the air.

Note 2: The modified speed cannot be saved via SSET, thus after a restart of the module the default speed of 115 200 is set again.

Parameter description:

<Speed>

Range: 115200 … 2000000 bps

Return value description: **=<Speed>**

© 2019 All Rights Reserved

ASCII

Parameter (Format):

<Speed> 7 byte (dec)

```
Example: SUAS 115200
```

Return value (Format):

<Speed> 7 byte (dec)

```
Example: =115108
```

BINARY

Type: GET/SET

Parameter (Format):

<Speed> 4 byte (uint32_t)

```
Example: SET SUAS 115200
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x06	0x55	0x0a	0x0001c200	0xb5	0xc0

Return value (Format):

<Speed> 4 byte (uint32_t)

```
Example: RESP SUAS 115108
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x06	0x57	0x0a	0x0001c1a4	0xb5	0x69

5.1.13. EAIR

EAIR <Enable>:

Enable/Disable the backchannel air interface. This means the (re)configuration of the swarm bee device over the air interface can be switched-on or off.

Parameter description:

<Enable>

Range: 0 … 1

0 = Node will not respond to AIR interface read/write commands.

1 = Node will respond to AIR interface read/write commands.

Return value description **=<Enable>**

Note: For security, this command can only be used over the UART, not over the air.

ASCII

Parameter (Format):

<Enable> byte (dec)

```
Example: EAIR 1
```

© 2019 All Rights Reserved

Return value (Format):

<Enable> 1 byte (dec)

```
Example: =1
```

BINARY

Type: GET/SET

Parameter (Format):

<Enable> 1 byte (uint8_t)

```
Example: SET EAIR 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x0c	0x01	0x61	0x54

Return value (Format):

<Enable> 1 byte (uint8_t)

```
Example: RESP EAIR 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x0c	0x01	0x60	0x9e

5.2. Ranging Commands

5.2.1. EPRI

EPRI <Enable>:

Enable Privacy mode: Enables and disables response to a received ranging request.

Parameter description:

<Enable>

Range: 0 … 1

0 = Node will respond to ranging requests

1 = Node will not respond to ranging requests

Return value description **=<Enable>**

ASCII

Parameter (Format):

<Enable> 1 byte (dec)

```
Example: EPRI 1
```

Return value (Format):

<Enable> 1 byte (dec)

```
Example: =1
```


© 2019 All Rights Reserved

BINARY

Type: GET/SET

Parameter (Format):

<Enable> 1 byte (uint8_t)

```
Example: SET EPRI 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x10	0x01	0xc9	0x9e

Return value (Format):

<Enable> 1 byte (uint8_t)

```
Example: RESP EPRI 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x10	0x01	0x68	0x5e

5.2.2. SPBL

SPBL <Option=0> :
SPBL <Option=1/2> <ID>:

Sets privacy blacklist: Handles the privacy black list used to block ranging operation with specified node IDs; maximum number of entries is 19.
The blacklist is a sort of stealth mode. That means when the device gets a ranging request from a swarm bee which node ID is contained in the blacklist, it will ignore it and thus not answer to the request.

In contrast to the white list the black list doesn't respond to a ranging request. The white list react on a Node ID blink and will only emit a ranging request if this Node ID is contained in this (white) list. See also section 5.2.7

Parameter description:

<Option>

Range: 0 … 2

0 = Clear black list

1 = Add ID to the black list

2 = Remove ID from the black list

<ID> 6 byte Node ID of ranging partner node

Range: 000000000001 … FFFFFFFFFFFE

Return value description **=<ErrorCode>**

<ErrorCode> indicating status of operation

Format: 1 byte (dec)

Range: 0…2

0 = success ◊ value stored/removed or list cleared

1 = list full, not possible to add more ID

2 = ID to be removed is not in the list

ASCII

© 2019 All Rights Reserved

Parameter (Format):

<Option> 1 byte (dec)

<ID> 12 bytes (hex)

```
Example: SPBL 1 0000BF260468
```

Return value (Format):

<ErrorCode> 1 byte (dec)

```
Example: =1
```

BINARY

Type: SET

Parameter (Format):

<Option> 1 byte (uint8_t)

<ID> 6 bytes (uint8_t[6])

```
Example: SET SPBL 1 0000bf260468
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x09	0x55	0x11	0x010000BF260468	0x9e	0x98

Return value (Format):

<ErrorCode> 1 byte (uint8_t)

```
Example: RESP SPBL 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x11	0x00	0xa8	0x0e

5.2.3. GPBL

GPBL <void>:

Gets (displays) privacy blacklist.

Parameter description:

<void>

Return value description: **<NumEntries>**\r\n[**<NodeID>**]*

<NumEntries> Number of lines after this line

Range: 0 … 19

<NodeID>

Range: 000000000001 … FFFFFFFFFFFE

ASCII

```
Example: GPBL
```

Return value (Format):

<NumEntries> 4 bytes (dec), first byte fixed „#“

© 2019 All Rights Reserved

<NodeID> 12 bytes (hex)

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

BINARY

Type: GET

```
Example: GET GPBL
```

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x54	0x11	0x46	0xd8

Return value (Format):

<NumEntries> 1 byte (uint8_t)

<NodeID> 6 bytes (uint8_t[6])

```
Example: RESP GPBL 3 DDF451534C23 134683567ABC 33A441FFB311
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x15	0x56	0x11	0x03DDF451534C23134683567ABC33A441FFB311	0x3a	0xc7

5.2.4. RATO

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

Range to: Initiates an elementary ranging cycle to a node with node ID **<ID>**

Parameter description:

<Option> defines if ranging starts synchronous or asynchronous

Range: 0 … 1

0 = Immediate start ranging operation

1 = Wait for node <ID> blink before starting ranging operation

<ID> 6 byte Node ID of ranging partner node

Range: 000000000001 … FFFFFFFFFFFE

<TimeOut> maximum time in ms to wait for the node <ID> blink.

Range: 0 … 65000

Return Value description:

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

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

Option = 0

<ErrorCode> indicating status of ranging operation

Range: 0 … 6

0 = success ranging result valid

© 2019 All Rights Reserved

1 = ranging to own ID
2 = no hardware ack received
3 = ranging unsuccessful, ranging timer expired
5 = overload, try again
6 = medium blocked CSMA timer expired

<RangingResult> returning the measured ranging distance in centimeter. Range depends on mode.

Range(ASCII): 000000 … 999999 ranging distance in centimeter [cm]
Range(BINARY): 0...999999 ranging distance in centimeter [cm]

<RSSI> RSSI value of remote node

Range: -128 .. -35

Option = 1
<Error> Indicate status of operation

Range: 0 … 1
0 = success, request accepted
1 = error, ranging to own ID is not allowed

ASCII

Parameter (Format):

<Option> 1 bytes (dec)
<ID> 12 bytes (hex)
<TimeOut> 5 bytes (dec)

```
Example: RATO 1 0000BF260468 1000
```

Return Values (Format):

<ErrorCode> 1 byte (dec)
<RangingResult> 6 bytes (dec) in [cm]
<RSSI> 4 byte (dec)
<Error>: 1 byte (dec)

```
Example: 0, 001843,-56
```

BINARY

Types: SET

Parameter (Format):

<Option> 1 byte (uint8_t)
<ID> 6 bytes (uint8_t[6])
<TimeOut> 2 bytes (uint16_t)

Return Values (Format with Option=0):

© 2019 All Rights Reserved

<ErrorCode> 1 byte (uint8_t)

<RangingResult> 4 bytes (uint32_t) in [cm]

<RSSI> 1 byte (int8_t)

Return Values (Format with Option=1):

<Error> 1 byte (uint8_t)

```
Example: SET RATO 1 0000BF260468 1000 (asynchronous)
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x0B	0x55	0x12	0x010000BF26046803E8	0x60	0xbc

1. Response: Return Value <Error=0> (accept request)

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x12	0x00	0xa8	0xfe

2. Response: *RRN (ranging result notification)

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x16	0x61	0x62	0x0000000000020000bf26046800000000940004cd	0x8d	0xd8

```
Example: SET RATO 0 0000BF260468
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x09	0x55	0x12	0x000000bf260468	0xce	0x4d

```
Response: RESP RATO 0
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x08	0x57	0x12	0x0000000045cb	0x50	0x07

5.2.5. BRAR

BRAR <Enable>:

Broadcast ranging results: Enables/Disables the broadcast transmission of ranging results after performing a ranging operation.

Parameter description:

<Enable>

Range: 0 … 1

0 = broadcast disabled

1 = broadcast enabled

Return value description: **=<Enable>**

ASCII

Parameter (Format):

<Enable> 1 byte (dec)

```
Example: BRAR 0
```

Return value (Format):

<Enable> 1 byte (dec)

© 2019 All Rights Reserved

```
Example: =0
```

BINARY:

Type: GET/SET

Parameter (Format):

<Enable> 1 byte (uint8_t)

```
Example: SET BRAR 0
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x13	0x00	0x08	0xae

Return value (Format):

<Enable> 1 byte (uint8_t)

```
Example: RESP BRAR 0
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x13	0x00	0xa9	0x6e

5.2.6. SROB

SROB <ClassMask>:

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 ID blink.

Parameter description:

<ClassMask> Device classes can be combined by performing a logical OR of ClassMask, e.g. ClassMask=0xA2 (=bin 1010 0010). The Node will initiate a ranging operation on broadcast to classes 2, 6, 8.

Range: 00 … FF

00 = (=bin 0000 0000) Range on Broadcast disabled

01 = (=bin 0000 0001) Range on Broadcast to device class 1

02 = (=bin 0000 0010) Range on Broadcast to device class 2

04 = (=bin 0000 0100) Range on Broadcast to device class 3

08 = (=bin 0000 1000) Range on Broadcast to device class 4

10 = (=bin 0001 0000) Range on Broadcast to device class 5

20 = (=bin 0010 0000) Range on Broadcast to device class 6

40 = (=bin 0100 0000) Range on Broadcast to device class 7

80 = (=bin 1000 0000) Range on Broadcast to device class 8

Return value: **=<ClassMask>**

ASCII

Parameter (Format):

<ClassMask> 2 bytes (hex)

```
Example: SROB A2
```


© 2019 All Rights Reserved

Return value (Format):

<ClassMask> 2 bytes (hex)

```
Example: =A2
```

BINARY:

Type: GET/SET

Parameter (Format):

<ClassMask> **1 byte (uint8_t)**

```
Example: SET SROB A2
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x14	0xa2	0x8b	0x27

Return value (Format):

<ClassMask> 1 byte (uint8_t)

```
Example: RESP SROB A2
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x14	0xa2	0x2a	0xe7

5.2.7. SRWL

SRWL <Option> <ID>:

Sets range white list: Handles the list of node IDs the node should range to. Maximum number of entries is 19.

The white list allows to range only to those swarm bee who's Node ID is contained in it. When a swarm bee receives a blink it looks if the ID is in the list and if it is the case it will start ranging.

In contrast to the white list the black list doesn't respond to a ranging request. See section 5.2.2

Parameter description:

<Option>

Range: 0 … 2

0 = Clear list

1 = Add ID to the list

2 = Remove ID from the list

<ID> 6 byte Node ID of ranging partner node

Range: 000000000001 … FFFFFFFFFFFE

Return value description: **=<Error>**

<Error> Indicating status of operation

Range: 0…2

0 = success, value stored/removed or list cleared

1 = list full, not possible to add more ID

2 = ID to be removed is not in the list

© 2019 All Rights Reserved

ASCII

Parameter (Format):

<Option> 1 byte (dec)

<ID> 12 bytes (hex)

```
Example: SRWL 1 0000BF260468
```

Return value (Format):

<Error> 1 byte (dec)

```
Example: =0
```

BINARY

Type: SET

Parameter (Format):

<Option> 1 byte (uint8_t)

<ID> 6 bytes (uint8_t[6])

```
Example: SET SRWL 1 0000BF260468
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x09	0x55	0x15	0x010000BF260468	0x9f	0x6b

Return value (Format):

<Error> 1 byte (uint8_t)

```
Example: RESP SRWL 0
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x15	0x00	0xaa	0xce

5.2.8. GRWL

GRWL <void>:

Gets (displays) ranging whitelist.

Parameter description:

<void>

Return value description: **<NumEntries>[<NodeID>]***

<NumEntries> Number of lines after this line

Range: 0 … 19

<NodeID>

Range: 000000000001 … FFFFFFFFFFFE

ASCII

© 2019 All Rights Reserved

```
Example: GRWL
```

Return value (Format):

<NumEntries> 4 bytes (dec), first byte fixed „#“

<NodeID> 12 bytes (hex)

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

BINARY

Type: GET

```
Example: GRWL
```

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x54	0x15	0x47	0x1b

Note: CRC_HIGH is 0x1b and must be escaped before transmitting (0x1b43).

Return value (Format):

<NumEntries> 1 byte (uint8_t)

<NodeID> 6 bytes (uint8_t[6])

```
Example: RESP GRWL 3 DDF451534C23 134683567ABC 33A441FFB311
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x15	0x56	0x15	0x03DDF451534C23134683567ABC33A441FFB311	0x43	0x52

5.2.9. ERRN

ERRN <Notify>:

Enables and disables ranging result notification

Parameter description:

<Notify>

Range: 0 … 1

0 = Node will not trigger host when data packet has been received

1 = Node will trigger host when data packet has been received

Return value description: **=<Notify>**

ASCII

Parameter (Format):

<Notify> 1 byte (dec)

```
Example: ERRN 1
```

© 2019 All Rights Reserved

Return value (Format):

<Notify> 1 byte (dec)

```
Example: =1
```

BINARY

Type: GET/SET

Parameter (Format):

<Notify> 1 byte (uint8_t)

```
Example: SET ERRN 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x16	0x01	0xca	0x3e

Return value (Format):

<Notify> 1 byte (uint8_t)

```
Example: RESP ERRN 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x16	0x01	0x6b	0xfe

5.2.10. SROF

SROF <Duration>:

Sets ranging delay offset: This sets the delay between the reception of a Node ID blink and the ranging request to this node. This setting can be useful if the blink is received by several nodes. With different offsets it can be avoided that all nodes are performing a ranging request at the same time to the same node.

Parameter description:

<Duration>

Range: 0 … 65000

Return value description **=<Duration>**

ASCII

Parameter (Format):

<Duration> 5 bytes (dec)

```
Example: SROF 00100
```

Return value (Format):

<Duration> 5 bytes (dec)

```
Example: =00100
```

BINARY

Type: GET/SET

© 2019 All Rights Reserved

Parameter (Format):

<Duration> 2 bytes (uint16_t)

```
Example: SET SROF 100
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x04	0x55	0x17	0x0064	0x5a	0x2c

Return value (Format):

<Duration> 2 byte (uint16_t)

```
Example: RESP SROF 100
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x04	0x57	0x17	0x0064	0x5b	0x94

5.3. Data Communication Commands

5.3.1. EDAN

EDAN <Notify>:

Enables and disables data notification towards the host. This allows to pass user data through a SDAT or BDAT and notify the host (*DNO) that data has been received. See also sect. 7.1.

Parameter description:

<Notify>

Range: 0 … 1

0 = Node will not trigger host when data packet has been received

1 = Node will trigger host when data packet has been received

Return value description **=<Notify>**

ASCII

Parameter (Format):

<Notify> 1 byte (dec)

```
Example: EDAN 1
```

Return value (Format):

<Notify> 1 byte (dec)

```
Example: =1
```

BINARY:

Type: GET/SET

Parameter (Format):

<Notify> 1 byte (uint8_t)

```
Example: SET EDAN 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH

© 2019 All Rights Reserved

0x7f	0x03	0x55	0x20	0x01	0xdd	0x9e

Return value (Format):

<Notify> 1 byte (uint8_t)

```
Example: RESP EDAN 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x20	0x01	0x7c	0x5e

5.3.2. EDNI

EDNI <Notify>

Enables and disables notification of Node ID broadcast with user data. This allows to pass user data through a blink and notify the host (*DNI) that data has been received. See also sect. 7.2.

Parameter description:

<Notify>

Range: 0 … 1

0 = Node will not trigger host when Node ID Broadcast with user data has been received

1 = Node will trigger host when Node ID Broadcast with user data has been received

Return value description: **=<Notify>**

ASCII

Parameter (Format):

<Notify> 1 byte (dec)

```
Example: EDNI 1
```

Return value (Format):

<Notify> 1 byte (dec)

```
Example: =1
```

BINARY

Type: GET/SET

Parameter (Format):

<Notify> 1 byte (uint8_t)

```
Example: SET EDNI 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x2b	0x01	0xda	0xae

Return value (Format):

<Notify> 1 byte (uint8_t)

```
Example: RESP EDNI 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x2b	0x01	0x7b	0x6e

© 2019 All Rights Reserved

5.3.3. SDAT

This commands sends user payload as described below ro it can be used to configure another node via the AIR interface. See section 4.2.3.

Important note: Please read carefully the note in section 4.2.3 to avoid unintentional (re)configuration of a remote swarm bee device.

SDAT <Option=0> <ID> <Len> <Data>

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

Sends <Data> of length <Len> to node <ID>. Depending on <OPTION> it transmits immediately or after the node <ID> blink.

Parameter description:

<Option>

Range: 0 … 1

0 = Immediate transmission

1 = Wait for node <ID> blink before transmitting

Note: With option 0 the receiving device must be listening at exactly at that time, otherwise the data is lost. This condition can be reached if the receiving node is in mode SPSA 0.
With option 1 the receiver must be enabled to receive node ID blinks. Thus disable power saving (SPSA 0) to receive asynchronous node ID blinks.
The data is stored temporarily until it has been transmitted or the given time out has occurred. The buffer can contain up to 10 different ID - Data pairs. If the data of a particular node <ID> hasn't been transmitted and a new SDAT with the same <ID> is issued it will overwrite the old data. To avoid that, wait until the asynchronous *SDAT with the associated <PayloadID> acknowledges its transmission.

<ID> 6 byte Node ID of ranging partner node.

Range: 000000000001 … FFFFFFFFFFFE

<Len> length of payload in bytes (hex)

Range: 01 … 80 for swarm bee LE

Range: 01 … 67 for swarm bee ER

<Data> payload to be transmitted

Range: 00 … FF

<TimeOut> maximum time in ms to wait for the node <ID> blink.

Range: 0 … 65000

Return value description:

<Option=0>: =<ErrorCode>

<ErrorCode> Status of transmission.

Range: 0 ... 2

0 = successfully transmitted

1 = no hardware ack received

2 = overload, try again later

3 = medium blocked, CSMA timer expired

<Option=1>: =<PayloadID>

© 2019 All Rights Reserved

<PayloadID> ID of <Data>, used to identify the asynchronous return value (*SDAT)

Range: 00000000 … FFFFFFFF

Note: After issuing a SDAT command an asynchronous response will be generated when the transmission is processed. The notification is described in chapter 7.5.

ASCII

Parameter (Format):

<Option> 1 byte (dec)

<ID> 12 bytes (hex)

<Len> 2 bytes (hex) = times 2 bytes of payload

<Data> 2 bytes (hex)

<TimeOut> 5 bytes (dec)

```
Example: SDAT 1 1F318052001A 02 FA13 1000
```

Return value (Format):

<ErrorCode> 1 byte (hex)

<PayloadID> 8 bytes (hex)

```
Example: =A4A865F8
```

BINARY

Type: GET/SET

Parameter (Format):

<Option> 1 byte (uint8_t)

<ID> 6 bytes (uint8_t[6])

<Len> 1 byte (uint8_t)

<Data> <Len> bytes (uint8_t[Len])

<TimeOut> 2 bytes (uint16_t)

```
Example: SET SDAT 0 1F318052001A 02 FA13
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x0C	0x55	0x21	0x001F318052001A02FA13	0x6b	0xa4

Return value (Format):

<ErrorCode> 1 byte (uint8_t)

```
Example: RESP SDAT <ErrorCode=1>
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x21	0x01	0x7d	0xce

```
Example: SET SDAT 1 1F318052001A 02 FA13 1000
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x0c	0x55	0x21	0x011F318052001A02FA1303e8	0x51	0xaf3

Return value (Format):

© 2019 All Rights Reserved

<PayloadID> 4 bytes (uint8_t[4])

```
Example: RESP SDAT <PayloadID=0xa4a865f8>
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x06	0x57	0x21	0xa4a865f8	0x18	0x86

5.3.4. GDAT

GDAT <void>:

Gets data: Reads out received data from the buffer. See also section 7.1 and 7.2.

Parameter description:

<void>

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

<Number of bytes> 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

Range: 00…80

<ID> returns ID of node which sent message

Range: 000000000001 … FFFFFFFFFFFF

<Payload> payload received

Range: 00 … FF

ASCII

```
Example: GDAT
```

Return value (Format):

<Number of bytes> 2 bytes (hex)

<ID> 12 Bytes (hex)

<Payload> 2 bytes (hex)

```
Example: =02,1F318052001A,FA13
```

BINARY:

Type: GET

```
Example: GET GDAT
```

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x54	0x27	0xc6	0xce

Return value (Format):

<Number of bytes> 1 byte (uint8_t)

<ID> 6 bytes (uint8_t[6])

<Payload> <Number of bytes> bytes (uint8_t[<Number of bytes>])

© 2019 All Rights Reserved

Note: If **<Number of bytes>** is 0, because no data is available, than <ID> and <Payload> do not exist in return message.

Example: RESP GDAT 02 1F318052001A FA13

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x0C	0x56	0x27	0x021F318052001AFA13	0xe8	0x5f

Example: RESP GDAT no data present

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x56	0x27	0x00	0xee	0xe6

5.3.5. BDAT

The BDAT serves to broadcast a message to any swarm bee. If it is immediate it sends the BDAT once. If option 1 is used it will send a BDAT sequentially to any received Node ID blink until timeout has been reached or if it has been canceled by BDAT option 3

BDAT <Option=0> <Len> <Data>

BDAT <Option=1> <Len> <Data> <TimeOut>

BDAT <Option=2>

Broadcasts <Data> of length <Len> to all nodes

Parameter description:

<Option>

Range: 0 … 2

0 = Immediate broadcast transmission

1 = Send the packet after any node ID blink.

2 = Cancel broadcasting

Note: See also the note in sect. 5.3.3

<Len> length of payload in bytes (hex)

Range: 01 … 70 for swarm bee LE

Range: 01 … 5D for swarm bee ER

<Data> payload to be transmitted

Range: 00 … FF

<TimeOut>

Range: 0 … 65000

0 = timeout disabled, broadcast will be perpetual.

> 0 = time in ms during which the node transmits after a node ID blink.

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

© 2019 All Rights Reserved

<Option=1>: =<PayloadID>

<Option=2>: **=0**

<ErrorCode> Status of transmission.

Format: 1 byte (dec)

Range: 0 ... 1

0 = Successfully transmitted

1 = Transmission failed

2 = Overload, try again later

<PayloadID> ID of <Data>, used to identify the asynchronous return value (*BDAT)

Format: 8 byte (hex)

Range: 00000000 … FFFFFFFF

Note: After issuing a BDAT command an asynchronous response will be generated when the transmission is processed. The response is described in chapter 7.

ASCII

Parameter (Format):

<Option> 1 bytes (dec)

<Len> 2 bytes (hex)

<Data> 2 bytes (hex)

<TimeOut> 5 bytes (dec)

```
Example: BDAT 0 2 FA13
```

Return value (Format):

<ErrorCode> 1 byte (dec)

<PayloadID> 8 bytes (hex)

```
Example: =0
```

BINARY:

Type: SET

Parameter (Format):

<Option> 1 byte (uint8_t)

<Len> 1 byte (uint8_t)

<Data> <Len> bytes (uint8_t[Len])

<TimeOut> 2 bytes (uint16_t)

```
Example: SET BDAT 0 2 FA13
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x06	0x55	0x22	0x0002FA13	0x77	0xcb

© 2019 All Rights Reserved

Return value (Format):

<ErrorCode> 1 byte (uint8_t)

<PayloadID> 4 bytes (uint8_t[4])

```
Example: RESP BDAT 0
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x22	0x00	0xbc	0xfe

5.3.6. FRAD

FRAD <Len> <Data>:

Fills the ranging data buffer with <Data> of length <Len>. The data <Data> is contained in each ranging request message until the buffer is cleared with <Len>=0. Each receiving swarm bee will generate a *DNO. This may cause a lot of traffic for each swarm device.

Parameter description:

<Len> length of ranging data payload in bytes (hex)

Range: 00 … 74 for swarm bee LE

Range: 00 … 5A for swarm bee ER

0 = delete data

<Data> payload to be transmitted

Range: 00 … FF

Return value description **=<ErrorCode>**

<ErrorCode> Status on ranging data buffer fill operation

Range: 0 … 1

0 = successful

1 = not successful

ASCII

Parameter (Format):

<Len> 2 bytes (hex)

<Data> <Len> * 2 bytes (hex)

```
Example: FRAD 0A FA13192F680426AE2345
```

Return value (Format):

<ErrorCode> 1 byte (dec)

```
Example: =1
```

BINARY

Type: GET/SET

Parameter (Format):

<Len> 1 byte (uint8_t)

<Data> <Len> byte (uint8_t[Len])

© 2019 All Rights Reserved

```
Example: SET FRAD 0A FA13192F680426AE2345
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x0d	0x55	0x2a	0x0AFA13192F680426AE2345	0xef	0xf2

Return value (Format for SET response):

<ErrorCode> 1 byte (uint8_t)

Return value (Format for GET response):

<Len> 1 byte (uint8_t)

<Data> <Len> byte (uint8_t[Len])

```
Example: RESP FRAD <ErrorCode=0>
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x2a	0x00	0xbb	0x3e

5.3.7. EIDN

EIDN <Notify>:

Enables and disables Node ID broadcast notification

Parameter description:

<Notify>

Range: 0 … 1

0 = Node will not trigger host when Node ID Broadcast has been received

1 = Node will trigger host when Node ID Broadcast has been received

Return value description **=<Notify>**

<Notify> returning parameter which has been set

Range: 0 …1

ASCII

Parameter (Format):

<Notify> 1 byte (dec)

```
Example: EIDN 1
```

Return value (Format):

<Notify> 1 byte (dec)

```
Example: =1
```

BINARY

Type: GET/SET

Parameter (Format):

<Notify> 1 byte (uint8_t)

```
Example: SET EIDN 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH

© 2019 All Rights Reserved

0x7f	0x03	0x55	0x26	0x01	0xde	0x3e

Return value (Format):

<Notify> 1 byte (uint8_t)

```
Example: RESP EDIN 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x26	0x01	0x7f	0xfe

Note: CRC_LOW is 0x7f and must be escaped before transmitting (0x1b53).

5.3.8. FNIN

FNIN <Len=0>:

FNIN <Len>0> <Data>:

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. If EDNI is not enabled, the data is not visible for swarm devices, because no *DNI is generated. This command is used to provide user data through a Node ID blink.

Parameter description:

<Len> length of ranging data payload in bytes (hex)

Range: 00 … 5B for swarm bee LE

Range: 00 … 48 for swarm bee ER

0 = delete data

<Data> payload to be transmitted

Range: 00 … FF

Return value description **=<ErrorCode>**

<ErrorCode> Status on ranging data buffer fill operation

Range: 0 … 1

0 = successful

1 = not successful

ASCII

Parameter (Format):

<Len> 2 bytes (hex)

<Data> <Len> * 2 bytes (hex)

```
Example: FNIN 0A FA13192F680426AE2345
```

Return value (Format):

<ErrorCode> 1 byte (dec)

```
Example: =0
```

BINARY

Type: GET/SET

© 2019 All Rights Reserved

Parameter (Format):

<Len> 1 byte (uint8_t)

<Data> <Len> bytes (uint8_t[Len])

```
Example: SET FNIN 0A FA13192F680426AE2345
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x0C	0x55	0x28	0x0AFA13192F680426AE2345	0xba	0x5e

Return value (Format for SET response):

<ErrorCode> 1 byte (uint8_t)

Return value (Format for GET response):

<Len> 1 byte (uint8_t)

<Data> <Len> byte (uint8_t[Len])

```
Example: RESP FNIN 0
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x28	0x00	0xba	0x5e

5.4. swarm radio node Identification

5.4.1. EBID

EBID <Enable>:

Enable/Disable broadcast node ID: Enables/disables the broadcast of the periodic Node ID blink packets. It doesn't affect the Node ID blinks generated by an interrupt coming from a GPIO or the MEMS if set so. The effect is immediate and thus no further Node ID blink is issued.

Important note: Never set EBID 0, ICFG 0 at the same time if the swarm bee module is intended to be operated by the AIR interface only, because it will never be accessible again and thus virtually dead. It can only be recovered by a reset or power off/on and if an access to the UART is possible.

Parameter description:

<Enable>

Range: 0 … 1

0 = Broadcast of Node ID blink packets disabled

1 = Broadcast of Node ID blink packets enabled

Return value description **=<Enable>**

ASCII

Parameter (Format):

<Enable> 1 byte (dec)

```
Example: EBID 1
```

Return value (Format):

© 2019 All Rights Reserved

<Enable> 1 byte (dec)

```
Example: =1
```

BINARY

Type: GET/SET

Parameter (Format):

<Enable> 1 byte (uint8_t)

```
Example: SET EBID 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x30	0x01	0xd0	0x5e

Return value (Format):

<Enable> 1 byte (uint8_t)

```
Example: RESP EBID 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x30	0x01	0x71	0x9e

5.4.2. SBIV

SBIV <Time>:

Sets the broadcast interval in which the Node ID will be sent. This command takes effect immediately.

Parameter description:

<Time> Blink interval in milliseconds [ms]

Range: 00050 … 65000

Note: AIR interface SBIV needs to be unlocked first, if new interval is less than 50ms.

Return value description **=<Time>**

ASCII

Parameter (Format):

<Time> 5 bytes (dec)

```
Example: SBIV 10000
```

Return value (Format):

<Time> 5 bytes (dec)

```
Example: =10000
```

BINARY

Type: GET/SET

Parameter (Format):

<Time> 2 bytes (uint16_t)

© 2019 All Rights Reserved

```
Example: SET SBIV 10000
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x04	0x55	0x31	0x2710	0xa0	0x30

Return value (Format):

<Time> 2 bytes (uint16_t)

```
Example: RESP SBIV 10000
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x04	0x57	0x31	0x2710	0xa1	0x88

5.4.3. NCFG

NCFG <Mask>:

<Mask> sets visible information in ranging result notifications and node ID notifications. This allows control over the information to be displayed each time a "*RRN:" or "*NIN:" is generated. If the information for a desired field is missing, it is filled up with "?".

Parameter description:

<Mask>

Range: 0...FFFF

Bit 0 = Device Class

Format ASCII: 1 byte (dec)

Format BINARY: 1 byte (uint8_t)

Range: 1... 8

Description: Device class of remote node

Bit 1 = MEMS values consists of x,y,z acceleration.

<XACC>

Format ASCII: 6 bytes (dec), signed INT16, first byte "+" or "-"

Format BINARY: 2 bytes (int16_t)

Range: -32768 … +32767

Description: Acceleration value.

<YACC> See <XACC>

<ZACC> See <XACC>

Bit 2 = RSSI values

Format ASCII: 4 bytes (dec)

Format BINARY: 1 byte (int8_t)

Range: -128 .. -35

Description: RSSI value of remote node.

Bit 3 = TEMP values

Format ASCII: 3 byte (dec), first byte "+" or "-"

Format BINARY: 1 byte (int8_t)

Range: -99 .. +99

Description: Temperature value of remote node.

Bit 4 = Power saving mode

Format ASCII: 1 byte (dec)

© 2019 All Rights Reserved

Format BINARY: 1 byte (uint8_t)

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 ASCII: 3 bytes (dec)

Format BINARY: 1 byte (uint8_t)

Range: 0 … 255

Description: Node's own battery level is in dV.

(Example: 32 = 3.2V)

Bit 6 = GPIO Status

Format ASCII: 2 bytes (hex)

Format BINARY: 1 byte (uint8_t)

Range: 00 … FF

Bit 0 = DIO_0

Bit 1 = DIO_1

Bit 2 = DIO_2

Bit 3 = DIO_3

Description: GPIO Pin status.

Bit 7 = Wakeup reason

Format ASCII 2 bytes (hex)

Format BINARY: 1 byte (uint8_t)

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 ASCII: 3 bytes (dec)

Format BINARY: 1 byte (uint8_t)

Range: 0 … 255

Description: Blink ID.

Bit 9 = RX slot counter.

Format ASCII: 3 bytes (dec)

Format BINARY: 1 byte (uint8_t)

Range: 0 … 255

Description: RX slot counter.

Bit 10 = Timestamp of broadcast message.

© 2019 All Rights Reserved

Format ASCII: 8 bytes (dec)

Format BINARY: 4 bytes (uint32_t)

Range: 0 … 4294967295

Description: Timestamp in milliseconds of broadcast message.

Return value description: **=<Mask>**

ASCII

Parameter (Format):

<Mask> 4 bytes (hex)

```
Example: NCFG 1ff
```

Return value (Format):

<Mask> 4 bytes (hex)

```
Example: =1
```

BINARY:

Type: GET/SET

Parameter (Format):

<Mask> 2 bytes (uint16_t)

```
Example: SET NCFG 1ff
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x04	0x55	0x32	0x01ff	0x0b	0xdc

Return value (Format):

<Mask> 2 bytes (uint16_t)

```
Example: RESP NCFG 1ff
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x04	0x57	0x32	0x01ff	0x0a	0x64

5.5. Medium Access Commands

5.5.1. Common Commands

5.5.1.1. SRXW

SRXW <Time>:

Sets the reception window. It is the time span the receiver listens after it has sent its own Node ID blink in power saving mode.

Parameter description:

<Time> Aperture time in milliseconds [ms]

Range: 0 … 65000

0 = Receiver is continuously disabled

© 2019 All Rights Reserved

1 – 65000 = Time in ms that the receiver stays on after its Node ID broadcast. If <TIME> is greater or equals the NodeID broadcast interval the receiver is always active

Return value description **=<Time>**

ASCII

Parameter (Format):

<Time> 5 bytes (dec)

```
Example: SRXW 00010
```

Return value (Format):

<Time> 5 bytes (dec)

```
Example: =00010
```

BINARY:

Type: GET/SET

Parameter (Format):

<Time> 2 bytes (uint16_t)

```
Example: SET SRXW 10
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x04	0x55	0x40	0x000A	0x6a	0x10

Return value (Format):

<Time> 2 bytes (uint16_t)

```
Example: RESP SRXW 10
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x04	0x57	0x40	0x000A	0x6b	0xa8

5.5.1.2. SRXO

SRXO <Value>:

Sets reception window occurrence.

Note: If streaming is enabled, or power saving mode is disabled, the receiver is always open, but asynchronous transmissions still use this window for sending packets. If SRXO is set to 2, even if the receiver is always open, asynchronous packets are only send every other node ID notification.

Parameter description:

<Value>

Range: 0 … 255

0 = Reception window disabled.

1-255 = Number of Node ID Blinks with an RxWindow. If 1, every Node ID Blink an RxWindow, if 2, every second Node ID Blink an RxWindow.

Return value description **=<Value>**

ASCII

Parameter (Format):

© 2019 All Rights Reserved

<Value> 3 bytes (dec)

```
Example: SRXO 1
```

Return value (Format):

<Value> 3 bytes (dec)

```
Example: =1
```

BINARY

Type: GET/SET

Parameter (Format):

<Value> 1 byte (uint8_t)

```
Example: SET SRXO 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x41	0x01	0xf4	0x0e

Return value (Format):

<Value> 1 byte (uint8_t)

```
Example: RESP SRXO 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x41	0x01	0x55	0xce

5.5.1.3. SDCL

SDCL <Class>:

Sets the node's device class which will be broadcasted in its blink ID packets.

Parameter description:

<Class>

Range: 1 … 8

Return value description **=<Class>**

ASCII

Parameter (Format):

<Class> 1 byte (dec)

```
Example: SDCL 1
```

Return value (Format):

<Class> 1 byte (dec)

```
Example: =1
```

BINARY:

Type: GET/SET

Parameter (Format):

© 2019 All Rights Reserved

<Class> 1 byte (uint8_t)

```
Example: SET SDCL 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x42	0x01	0xf4	0xfe

Return value (Format):

<Class> 1 byte (uint8_t)

```
Example: RESP SDCL 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x42	0x01	0x55	0x3e

5.5.2. swarm bee LE specific Commands

5.5.2.1. STXP

STXP <Power>:

Sets the node's transmission power (0=minimum … 63=maximum).

Parameter description:

<Power> transmission power

Range: 00 … 63

Return value description: **=<Power>**

ASCII

Parameter (Format):

<Power> 2 bytes (dec)

```
Example: STXP 63
```

Return value (Format):

<Power> 2 byte (dec)

```
Example: =63
```

BINARY

Type: GET/SET

Parameter (Format):

<Power> 1 byte (uint8_t)

```
Example: SET STXP 63
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x05	0x3F	0x46	0xde

Return value (Format):

© 2019 All Rights Reserved

<Power> 1 byte (uint8_t)

```
Example: RESP STXP 63
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x05	0x3F	0xe7	0x1e

5.5.2.2. *SSYC*

SSYC <SyncWord>:

Sets the node's PHY syncword (0 … 12). The node will only listen to air packets of its own syncword.

Parameter description:

<SyncWord>

Range: 0 … 12

Return value description: **=<SyncWord>**

ASCII

Parameter (Format):

<SyncWord> 2 byte (dec)

```
Example: SSYC 7
```

Return value (Format):

<SyncWord> 2 byte (dec)

```
Example: =7
```

BINARY

Type: GET/SET

Parameter (Format):

<SyncWord> byte (uint8_t)

```
Example: SET SSYC 7
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x06	0x07	0x47	0xfc

Return value (Format):

<SyncWord> 1 byte (uint8_t)

```
Example: RESP SSYC 7
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x06	0x07	0xe6	0x3c

5.5.2.3. SFEC

SFEC <Enable>:

© 2019 All Rights Reserved

Enables the forward error correction (FEC) mechanism. More details can be found in the nanoLOC User Guide [7]

Parameter description:

<Enable>

Range: 0 … 1

0 = FEC off

1 = FEC on

Return value description **=<Enable>**

ASCII

Parameter (Format):

<Enable> 1 byte (dec)

```
Example: SFEC 1
```

Return value (Format):

<Enable> 1 byte (dec)

```
Example: =1
```

BINARY

Type: GET/SET

Parameter (Format):

<Enable> 1 byte (uint8_t)

```
Example: SET SFEC 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x43	0x01	0xf5	0x6e

Return value (Format):

<Enable> 1 byte (uint8_t)

```
Example: RESP SFEC 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x43	0x01	0x54	0xae

5.5.2.4. SDAM

SDAM <Mode>:

Sets the node's data mode

Parameter description:

<Mode>

Range: 1 … 2

1 = 80MHz, 1µs per bit

2 = 80MHz, 4µs per bit

© 2019 All Rights Reserved

Return value description **=<Mode>**

ASCII

Parameter (Format):

<Mode> 1 byte (dec)

```
Example: SDAM 2
```

Return value (Format):

<Mode> 1 byte (dec)

```
Example: =2
```

BINARY

Type: GET/SET

Parameter (Format):

<Mode> 1 byte (uint8_t)

```
Example: SET SDAM 2
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x44	0x02	0xb7	0x5f

Return value (Format):

<Mode> 1 byte (uint8_t)

```
Example: RESP SDAM 2
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x44	0x02	0x16	0x9f

5.5.2.5. CSMA

CSMA <M> <Duration> <Threshold>:

Sets CSMA mode on or off and determines back-off factor for CSMA or uses fixed back-off factor of length <DURATION>*24µs. <Threshold> is for energy detection CSMA. When gain is below this <Threshold> the medium is interpreted as busy.

Parameter needed depending on <M>.

CSMA <M=0>

CSMA <M=1/2> <Duration>

CSMA <M=3/4> <Duration> <Threshold>

Parameter description:

<M>

Range: 0 ... 4

<M> = 0 CSMA off ◊ ALOHA

<M> = 1 CSMA symbol detection on, random seed, <Duration> parameter needed.

© 2019 All Rights Reserved

<M> = 2 CSMA symbol detection on, fixed back-off time, <Duration> parameter needed.

<M> = 3 CSMA energy detection on, random seed, <Duration> parameter and <Threshold> parameter needed.

<M> = 4 CSMA energy detection on, fixed back-off time, <Duration> parameter and <Threshold> parameter needed.

<Duration>

Range: 0 … 255

For <M>=1 or <M>=3 ◊ random seed

For <M>=2 or <M>=4 ◊ time in 24µs ticks

<Threshold>

Range: 0 … 63

Return value: =**<M=0>**

Return value: **=<M=1/2>,<Duration>**

Return value: **=<M=3/4>,<Duration>,<Threshold>**

Note: If the air interface is blocked for longer than 50 ms, the transmission is abandoned and an error code is produced. See section 5.2.4 and 5.3.3.

ASCII

Parameter (Format):

<M> 1 byte (dec)

<Duration> 3 bytes (dec)

<Threshold> 2 bytes (dec)

```
Example: CSMA 2 10
```

Return value (Format depends on <M>):

=<M>,<Duration>,<Threshold>

```
Example: = 2,10
```

BINARY

TYPE: GET/SET

Parameter (Format):

<M> 1 byte (uint8_t)

<Duration> 1 byte (uint8_t)

<Threshold> 1 byte (uint8_t)

```
Example: SET CSMA 4 255 18
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x05	0x55	0x45	0x04ff12	0x17	0xae

Return value (Format depends on <M>):

<M> 1 byte (uint8_t)

<Duration> 1 byte (uint8_t)

<Threshold> 1 byte (uint8_t)

```
Example: RESP CSMA 4 255 18
```


© 2019 All Rights Reserved

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x05	0x57	0x45	0x04ff12	0x6e	0x6e

5.5.3. swarm bee ER specific Commands

5.5.3.1. *STPD*

STPD <sm> <gain>

Sets the node's transmission power and disable/enable the smart power feature.
The power output regulations typically specify a transmit power limit of -41.3 dBm in each 1 MHz bandwidth, and generally measure this using a 1 ns dwell time in each 1 MHz segment. When sending short frames at 6.8 Mbps it is possible for a single frame to be sent in a fraction of a millisecond. Provided the transmitter does not transmit again within that same millisecond, the power of transmission can be increased above the -41.3 dBm limit while remaining in compliance with the regulations. If smart power is enabled, the output power is boosted by up to 9 dBm depending on the actual packet length.

Parameter description:

<sm> This enables/disables the smart power feature.

Range: 0 ... 1

<gain> This is the output power gain control in steps of 0.1 dB measured at the RF port. Depending on the selected antenna, connectors and wires, the gain must be adjusted. This value is added to the current output power. If the antenna has 3 dB gain, the output power must be reduced with this command "STPD 0, -30".

Range: -146 ... 59 (-14.6 … +5.9 dB)

Return value description: =<sm>,<gain>

ASCII:

Parameter(Format):
<sm> 1 byte (dec) <gain> 3 byte (dec)

```
Example: STPD 1 0
```

Return value(Format):
<sm> 1 byte (dec) <gain> 3 byte (dec)

```
Example: =1,0
```

BINARY:

TYPE: GET/SET

Parameter(Format):
<sm> 1 byte (uint8_t) <gain> 2 byte (int16_t)

```
Example: SET STPD 1 0
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x05	0x55	0x70	0x010000	0xC9	0xC9

Return value(Format):
<sm> 1 byte (uint8_t) <gain> 2 byte (int16_t)

```
Example: RESP STPD 1 0
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x05	0x57	0x70	0x010000	0xB0	0x5E

© 2019 All Rights Reserved

5.5.3.2. SDMD

SDMD <mode>

Sets the node's data mode by selecting the channel and the symbol duration.

Parameter description:

<mode> Selects the desired data transmission mode.
Range: 0 ... 18

Mode	Bit Rate (Kbps)	Channel	Centre freq. (MHz)	Bandwidth (MHz)
0**	--	--	--	--
1	110	1	3494.4	499.2
2	850			
3	6800			
4	110	2	3993.6	499.2
5	850			
6	6800			
7	110	3	4492.8	499.2
8	850			
9	6800			
10	110	4	3993.6	1331.2*
11	850			
12	6800			
13	110	5	6489.6	499.2
14	850			
15	6800			
16	110	7	6489.6	1081.6*
17	850			
18	6800			

* The swarm bee ER has a maximum receive bandwidth of 900 MHz
** Mode 0 is a custom mode. The settings of mode 0 are defined by the SDMC command.

Return value description: =<mode>

ASCII:

Parameter(Format):
<mode> 1 byte (dec)

```
Example: SDMD 14
```

Return value(Format):
<mode> 1 byte (dec)

```
Example: =14
```

BINARY:

TYPE: GET/SET

Parameter(Format):
<mode> 1 byte (uint8_t)

```
Example: SET SDMD 14
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x71	0x0e	0xa0	0x0a

Return value(Format):
<mode> 1 byte (uint8_t)

© 2019 All Rights Reserved

```
Example: RESP SDMD 14
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x71	0x0e	0x01	0xca

5.5.3.3. SDMC

SDMC <ch> <prf> <txPreambLen> <rxPAC> <txCode> <rxCode> <nsSFD> <dataRate> <phrMode> <sfdTo> <gain>

Sets the node's custom data mode. This mode is selected by using the command SDMD 0.

Note: When using this mode it is very important to have a deep knowledge on the parameters and their setting as well as their implications. Otherwise, it makes the system and communication unusable. More details can be found in chapter 3 and 4 of the DW 1000 User Manual [3].

Parameter description:

<ch> Channel number.

Range: 1 ... 7 (except 6)

<prf> Pulse repetition frequency.

prf	Freq. MHz
1	16
2	64

<txPreambLen> Preamble length.

txPreambLen	n bytes
1	64
2	128
3	256
4	512
5	1024
6	1536
7	2048
8	4096

Recommended preamble length depending on selected data rate:

Data Rate	Recommended
6.8Mbps	64, 128, 256
850kbps	256, 512, 1024
110kbps	1024, 1536, 2048

<rxPAC> Acquisition chunk size. A larger PAC size offers a better performance when the preamble is long enough to allow it.

rxPAC	n bytes
0	8
1	16
2	32
3	64

Recommended PAC size in relation to preamble length:

Preamble length in bytes	Recommended PAC size
64	8
128	8
256	16
512	16

© 2019 All Rights Reserved

1024	32
1536	64
4096	64

<txCode> TX preamble code:
The preamble code can be used to split networks, but there is still cross correlation between these channels. Recommended preamble codes:

Channel	Preamble Codes	
	16 MHz PRF	64 MHz PRF
1	1,2	9,10,11,12
2	3,4	9,10,11,12
3	5,6	9,10,11,12
4	7,8	17,18,19,20
5	3,4	9,10,11,12
7	7,8	17,18,19,20

<rxCode> RX preamble code. See <txCode> for more information.

<nsSFD> Enable/disables the non-standard DecaWave proprietary SFD sequence.

nsSFD	Description
0	IEEE 802.15.4-2011 standard
1	non-standard sequence

<dataRate> Selects the data rate.

nsSFD	Description
0	110 kbps
1	850 kbps
2	6800 kbps

<phrMode> Selects between either the standard or the extended PHR mode.

phrMode	Frame length
0	5 - 127 Bytes
1	5 - 1023 Bytes

<sfdTo> SFD timeout value in symbols. The purpose of detection timeout is to recover from occasional false preamble detection events that may occur. This value should cover the complete preamble. If shorter preambles are used, this value can be reduced appropriately.

Range: 0 ... 4161

<sfdTo> = 0 will set 4161 symbols, which covers the largest possible preamble.
To calculate the appropriate sfdTo use following formula:
sfdTo = preambleLen + 64 + 1
Example: 4096 + 64 + 1 = 4161 symbols

<gain> This is the RF output power in steps of 0.1 dB
Range: 0 … 335

ASCII:

Parameter(Format):
<ch> 1 byte (dec) **<prf>** 1 byte (dec) **<txPreambLen>** 1 byte (dec) **<rxPAC>** 1 byte (dec) **<txCode>** 2 byte (dec) **<rxCode>** 2 byte (dec) **<nsSFD>** 1 byte (dec) **<dataRate>** 1 byte (dec) **<phrMode>** 1 byte (dec) **<sfdTo>** 4 byte (dec) **<gain>** 3 byte (dec)

```
SDMC 1 1 2 1 1 1 1 2 0 193 227
```

Return value(Format):
=<ch>,<prf>,<txPreambLen>,<rxPAC>,<txCode>,<rxCode>,<nsSFD>,<dataRate>,<phrMode>,<sfdTo>

 © 2019 All Rights Reserved

```
SDMC 1 1 2 1 1 1 1 2 0 193 227
```

BINARY:

TYPE: GET/SET

Parameter(Format):

<ch> 1 byte (uint8_t) **<prf>** 1 byte (uint8_t) **<txPreambLen>** 1 byte (uint8_t) **<rxPAC>** 1 byte (uint8_t) **<txCode>** 1 byte (uint8_t) **<rxCode>** 1 byte (uint8_t) **<nsSFD>** 1 byte (uint8_t) **<dataRate>** 1 byte (uint8_t) **<phrMode>** 1 byte (uint8_t) **<sfdTo>** 2 byte (uint16_t) **<gain>** 2 byte (uint16_t)

```
Example: SDMC 1 1 2 1 1 1 1 2 0 193 227
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x0f	0x55	0x72	0x01010201010101020000c100e3	0xda	0xa5

Return value(Format):

<ch> 1 byte (uint8_t) **<prf>** 1 byte (uint8_t) **<txPreambLen>** 1 byte (uint8_t) **<rxPAC>** 1 byte (uint8_t) **<txCode>** 1 byte (uint8_t) **<rxCode>** 1 byte (uint8_t) **<nsSFD>** 1 byte (uint8_t) **<dataRate>** 1 byte (uint8_t) **<phrMode>** 1 byte (uint8_t) **<sfdTo>** 2 byte (uint16_t)

```
Example: RESP SDMC 1 1 2 1 1 1 1 2 0 193 227
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x0f	0x57	0x72	0x01010201010101020000c100e3	0x58	0xa4

5.5.3.4. SOFF/GOFF

Set ranging distance offset, and get ranging offset are used to reduce the ranging error to a minimum. Each device have depending on the channel, the data rate as well as cable length a certain distance measurement offset. The SOFF command allows to set a constants for each individual data mode (SDMD).

5.5.3.4.1 SOFF

SOFF <mode> <offset>

Sets the node's ranging distance offset depending on the selected mode (SDMD).
Parameter description:

<mode> Selects the data mode to which the ranging offset shall belong.

Range: 0 ... 18

<offset> The offset in [cm].

Range: -32768 ... 32767

ASCII:

Parameter(Format):

<mode> 2 byte (dec) <offset> 6 byte (dec)

```
Example: SOFF 1 15500
```

Return value(Format): =<mode>,<offset>

BINARY:

TYPE: SET

Parameter(Format):

<mode> 1 byte (uint8_t) <offset> 2 byte (int16_t)

```
Example: SOFF 1 15500
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x05	0x55	0x75	0x013c8c	0xd9	0xf7

© 2019 All Rights Reserved

Return value(Format): **<mode>** 1 byte (uint8_t) <offset> 2 byte (int16_t)

```
Example: RESP SOFF 1 15500
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x05	0x57	0x75	0x013c8c	0xa0	0x37

5.5.3.4.2 GOFF

GOFF <void>

Readout (display) the node's ranging offset values.

Parameter description:
<void>

Return value description:

#**<NumLines>**rn<mode>,<value>rn ..
<NumLines> Number of lines after this line.
Range: 0 ... 255
<mode> The data mode (SDMD). <value> Offset value related to data mode.

ASCII:

```
Example: GOFF
```

Return value(Format):
<NumLines> 2-4 byte (dec), first byte fixed "**#**"
<mode> 1-2 byte (dec) **<value>** 1-6 byte

```
Example: #19
         0,0
         1,15567
         2,15567
         3,15552
         4,15545
         5,15545
         6,15531
         7,15542
         8,15542
         9,15527
         10,15508
         11,15508
         12,15592
         13,15541
         14,15541
         15,15529
         16,15504
         17,15504
         18,15409
```


© 2019 All Rights Reserved

BINARY:

TYPE: GET

Parameter(Format):
None

```
Example: GOFF
```

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x54	0x75	0x47	0x33

Return value(Format):
<NumEntries> 1 byte (uint8_t)
<offset SDMD mode 0> 2 byte (int16_t) **<offset SDMD mode 1>** 2 byte (int16_t)
<offset SDMD mode 2> 2 byte (int16_t) **<offset SDMD mode 3>** 2 byte (int16_t)
<offset SDMD mode 4> 2 byte (int16_t) **<offset SDMD mode 5>** 2 byte (int16_t)
<offset SDMD mode 6> 2 byte (int16_t) **<offset SDMD mode 7>** 2 byte (int16_t)
<offset SDMD mode 8> 2 byte (int16_t) **<offset SDMD mode 9>** 2 byte (int16_t)
<offset SDMD mode 10> 2 byte (int16_t) **<offset SDMD mode 11>** 2 byte (int16_t)
<offset SDMD mode 12> 2 byte (int16_t) **<offset SDMD mode 13>** 2 byte (int16_t)
<offset SDMD mode 14> 2 byte (int16_t) **<offset SDMD mode 15>** 2 byte (int16_t)
<offset SDMD mode 16> 2 byte (int16_t) **<offset SDMD mode 17>** 2 byte (int16_t)
<offset SDMD mode 18> 2 byte (int16_t)

```
Example: RESP GOFF
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x28	0x56	0x75	0x00003ccf3ccf3cc0 3cb93cb93cab3cb6 3cb63ca73c943c94 3ce83cb53cb53ca9 3c903c903c31	0x41	0x79

5.5.3.5. SPAN

SPAN <id>

Configure the pan ID of swarm bee ER. The node will only listen to messages with its same PAN Id.

Parameter description:
<id> defines the pan id.
Range: 0 .. 0xfffe

Return values: =**<id>**

ASCII:

Parameter(Format):
<id> 1 ... 4 byte(hex)

```
Example: SPAN 17
```

Return value(Format):
<id> 1 ... 4 byte(hex)

```
Example: =0017
```

BINARY:

Parameter(Format):
<id> 2 byte(uint16_t)

```
Example: SET SPAN 17
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x04	0x55	0x0e	0x0017	0x3a	0x0e

© 2019 All Rights Reserved

Return value(Format):
<id> 2 byte(uint16_t)

```
Example: RESP SPAN 17
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x04	0x57	0x0e	0x0017	0x3b	0xb6

5.6. MEMS & Temperature Sensor Commands

5.6.1. EMSS

EMSS <Enable>:

Enables and disables the MEMS sensor on the *swarm* radio. This has an influence on the power consumption.

Parameter description:

<Enable>

Range: 0 … 1

0 = MEMS sensor will be disabled on node

1 = MEMS sensor will be enabled on Node

Return value description **=<Enable>**

ASCII

Parameter (Format):

<Enable> 1 byte (dec)

```
Example: EMSS 1
```

Return value (Format):

<Enable> 1 byte (dec)

```
Example: =1
```

BINARY

Type: GET/SET
Parameter (Format):

<Enable> 1 byte (uint8_t)

```
Example: SET EMSS 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x50	0x01	0xf8	0x5e

Return value (Format):

<Enable> 1 byte (uint8_t)

```
Example: RESP EMSS 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x50	0x01	0x59	0x9e

© 2019 All Rights Reserved

5.6.2. EBMS

EBMS <Notify>:

Enables and disables broadcasting of MEMS and TEMP values in blink packets

Parameter description:

<Notify>

Range: 0 … 1

0 =Node will not broadcast MEMS & TEMP values in blink packet

1 = Node will broadcast MEMS &TEMP values in blink packet

Return value description **=<Notify>**

<Notify> returning parameter which has been set

Range: 0 …1

ASCII

Parameter (Format):

<Notify> 1 byte (dec)

```
Example: EBMS 1
```

Return value (Format):

<Notify> 1 byte (dec)

```
Example: =1
```

BINARY

Type: GET/SET

Parameter (Format):

<Notify> 1 byte (uint8_t)

```
Example: SET EBMS 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x51	0x01	0xf9	0xce

Return value (Format):

<Notify> 1 byte (uint8_t)

```
Example: RESP EBMS 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x51	0x01	0x58	0x0e

5.6.3. SMRA

SMRA <Grange>:

Sets the MEMS range that is the sensor’s acceleration scale.

© 2019 All Rights Reserved

Parameter description:

<Grange>

Range: 1 … 4

1 = MEMS g range set to +/- 2g

2 = MEMS g range set to +/- 4g

3 = MEMS g range set to +/- 8g

4 = MEMS g range set to +/- 16g

Return value description: **=<Grange>**

Note: ERR returned if MEMS is switched off

ASCII

Parameter (Format):

<Grange> 1 byte (dec)

```
Example: SMRA 2
```

Return value (Format):

<Grange> 1 byte (dec)

```
Example: =2
```

BINARY

Type: GET/SET

Parameter (Format):

<Grange> 1 byte (uint8_t)

```
Example: SET SMRA 2
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x52	0x02	0xb9	0x3f

Return value (Format):

<Grange> 1 byte (uint8_t)

```
Example: RESP SMRA 2
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x52	0x02	0x18	0xff

5.6.4. SMTH

SMTH <Thres>:

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).

Parameter description:

<Thres>

Range: 0 … 255

© 2019 All Rights Reserved

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>

Return value description **=<Thres>**

Note: ERR returned if MEMS is switched off

ASCII

Parameter (Format):

<Thres> 3 bytes (dec)

```
Example: SMTH 100
```

Return value (Format):

<Thres> 3 bytes (dec)

```
Example: =100
```

BINARY

Type: GET/SET

Parameter (Format):

<Thres> 1 byte (uint8_t)

```
Example: SET SMTH 100
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x53	0x64	0x38	0x85

Return value (Format):

<Thres> 1 byte (uint8_t)

```
Example: RESP SMTH 100
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x53	0x64	0x99	0x45

5.6.5. SMBW

SMBW <BandWidth>:

Sets the MEMS sensor's filter bandwidth.

Parameter description:

<BandWidth>

Range: 1 … 8

1 = 7.81 Hz

2 = 15.63 Hz

3 = 31.25 Hz

4 = 62.50 Hz

© 2019 All Rights Reserved

5 = 125.00 Hz

6 = 250.00 Hz

7 = 500.00 Hz

8 = 1000.00 Hz

Return value description **=<BandWidth>**

Note: ERR returned if MEMS is switched off

ASCII

Parameter (Format):

<BandWidth> 1 byte (dec**)**

```
Example: SMBW 3
```

Return value (Format):

<BandWidth> 1 byte (dec)

```
Example: =3
```

BINARY

Type: GET/SET

Parameter (Format):

<BandWidth> 1 byte (uint8_t)

```
Example: SET SMBW 3
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x54	0x03	0x7b	0x5f

Return value (Format):

<BandWidth> 1 byte (uint8_t)

```
Example: RESP SMBW 3
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x54	0x03	0xda	0x9f

5.6.6. SMSL

SMSL <SleepTime>:

Sets the MEMS sensor's sleep time. This corresponds to the interval with which the MEMS will poll its acceleration. A longer sleep time will reduce the power consumption.

Parameter description:

<SleepTime>

Range: 1 … 11

1 = 0.5 ms

2 = 1 ms

© 2019 All Rights Reserved

3 = 2 ms

4 = 4 ms

5 = 6 ms

6 = 10 ms

7 = 25 ms

8 = 50 ms

9 = 100 ms

10 = 500 ms

11 = 1000 ms

Return value description **=<SleepTime>**

Note: ERR returned if MEMS is switched off

ASCII

Parameter (Format):

<SleepTime> 2 bytes (dec)

```
Example: SMSL 11
```

Return value (Format):

<SleepTime> 2 bytes (dec)

```
Example: =11
```

BINARY

Type: GET/SET

Parameter (Format):

<SleepTime> 1 byte (uint8_t)

```
Example: SET SMSL 11
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x55	0x55	0x0B	0x7b	0x09

Return value (Format):

<SleepTime> 1 byte (uint8_t)

```
Example: RESP SMSL 11
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x57	0x55	0x0B	0xda	0xc9

5.6.7. SMDT

SMDT <DeadTime>:

Sets the MEMS sensor's dead time. It is the guard period which is set after the µController has received the MEMS interrupt. Depending on the other settings, it prevents a cascade of interrupts which may cause unwanted air traffic and power consumption.

Parameter description:

<DeadTime> Time in milliseconds with MEMS interrupt disabled, after MEMS interrupt occurred.

© 2019 All Rights Reserved

Range: 0 … 65000

Return value: =**<DeadTime>**

ASCII

Parameter (Format):

<DeadTime> 5 bytes (dec)

```
Example: SMDT 1000
```

Return value (Format):

<DeadTime> 5 bytes (dec)

```
Example: =1000
```

BINARY

Type: GET/SET

Parameter (Format):

<DeadTime> 2 bytes (uint16_t)

```
Example: SET SMDT 1000
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x04	0x55	0x56	0x03E8	0x0b	0x6d

Return value (Format):

<DeadTime> 2 bytes (uint16_t)

```
Example: RESP SMDT 1000
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x04	0x57	0x56	0x03E8	0x0a	0xd5

5.6.8. GMYA

GMYA <void>:

Reports (displays) Node's own MEMS acceleration values

Parameter description:

<void>

Return value description:

<XACC> Acceleration value for X coordinate (positive and negative)

Range: -32768mg … +32767mg

<YACC> Acceleration value for Y coordinate (positive and negative)

Range: -32768 … +32767mg

<ZACC> Acceleration value for Z coordinate (positive and negative)

Range: -32768 … +32767mg

Note: Max. range depends on SMRA (2g,4g,8g,16g)

Note: ERR returned if MEMS is switched off

© 2019 All Rights Reserved

ASCII

```
Example: GMYA
```

Return value (Format):

<XACC> 6 bytes (dec), Signed INT16, first byte “+” or “-“

<YACC> 6 bytes (dec), Signed INT16, first byte “+” or “-“

<ZACC> 6 bytes (dec), Signed INT16, first byte “+” or “-“

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

BINARY

Type: GET

```
Example: GET GMYA
```

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x54	0x57	0xc7	0x2a

Return value (Format):

<XACC> 2 bytes (int16_t)

<YACC> 2 bytes (int16_t)

<ZACC> 2 bytes (int16_t)

```
Example: RESP GMYA +31599 +18501 -26468
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x08	0x56	0x57	0x7B6F4845989C	0x45	0x96

5.6.9. GMYT

GMYT <void>:

Reports (displays) Node’s own MEMS temperature values

Parameter description:

<void>

Return values: **=<Temperature>**

<Temperature> Node’s own current temperature (positive and negative) in °C

Range: -99 … +99

Note: ERR returned if MEMS is switched off

ASCII

```
Example: GMYT
```

Return value (Format):

<Temperature> 3 bytes (dec), first byte “+” or “-“

© 2019 All Rights Reserved

```
Example: =+26
```

BINARY:

Type: GET

```
Example: GET GMYT
```

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x54	0x58	0x87	0x2e

Return value (Format):

<Temperature> 1 byte (int8_t)

```
Example: RESP GMYT +26
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x56	0x58	0x1A	0x4f	0x95

5.6.10. GBAT

GBAT <void>:

Reports (displays) battery level of the node connected to the host or via the air interface. For V1 it measures the voltage applied at ADC_IN, for V2 it is the voltage measured at Vin. See also section 5.6.17.

Parameter description:

<void>

Return values: **=<BatteryStatus>**

<BatteryStatus> Node's own battery level in dV (example: =025 is 2.5V)

Range: 000 … 255

ASCII

```
Example: GBAT
```

Return value (Format):

<BatteryStatus> 3 bytes (dec)

```
Example: =025
```

BINARY

Type: GET

```
Example: GET GBAT
```

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x54	0x59	0x46	0xee

Return value (Format):

<BatteryStatus> 1 byte (uint8_t)

```
Example: RESP GBAT 25
```


© 2019 All Rights Reserved

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x56	0x59	0x19	0x0e	0x04

5.6.11. GPIO

This is the most complex and versatile command. It doesn't only allow to set each GPIO as input or output individually, but also to change the behavior of the swarm bee like changing the blink rate or to wake it up by an external signal. Moreover, it can be used to toggle dedicated GPIOs to let for instance blink a LED.

GPIO <Pin> <Mode=0/1> <Speed> <OType> <PUPD>

GPIO <Pin> <Mode=2>

GPIO <Pin> <Mode=3> <Interval> <ActiveState> <Priority>

GPIO <Pin> <Mode=4> <StartState> <OType> <Repetitions> <TimeHigh> <TimeLow>

Description: Configure GPIO's of µC.

Parameter description:

<Pin> defines which pin is affected by changes.

Range: 0 .. 3

0 = DIO_0

1 = DIO_1

2 = DIO_2

3 = DIO_3

<Mode>

Range: 0 ... 4

0 = GPIO Input Mode

1 = GPIO Output Mode

2 = Wake-up pin. In this mode, when the pin is set to high level the device goes to active mode and stays active as long as the pin is high. A GPIO pin in mode 2 overwrites any other power mode. Interrupts must be enabled with ICFG command separately. The pin is set as interrupt with pull-down

3 = Alternative blink interval mode. Interrupts must be enabled with ICFG command separately. The pin is set as interrupt without pull. An external pull-up or pull-down is required.

4 = GPIO twinkle mode. This mode allows to toggle the states of a dedicated GPIO pin for a given number of repetitions. This can be a LED which will blink with a certain interval for the given repetitions. This mode works only in SPSA mode 0, 1 and 3. Not in SPSA mode 2.

Note: When the inputs are used in conjunction with a switch or a push-button it is mandatory to use an appropriate debouncing circuitry.

Parameter description for <Mode=0/1>:

<Speed>

Range: 0 ... 3

0 = Very Low Speed (400 KHz)

1 = Low Speed (2MHz)

2 = Medium Speed (10MHz)

3 = High Speed (40MHz)

© 2019 All Rights Reserved

<OType>

Range: 0 … 1
0 = Push Pull
1 = Open Drain

<PUPD>

Range: 0 … 2
0 = No pull
1 = Pull up
2 = Pull down

Return values: **=<Pin>,<Mode>,<Speed>,<OType>,<PUPD>**

Parameter description for **<Mode=3>:**

<Interval> is the alternative blink interval in milliseconds.

Range: 0 ... 65000

<ActiveState> defines if the pin is low or high active.

Range: 0 … 1
0 = Low active
1 = High active

<Priority> the lowest value is the highest priority.

Range: 0 … 255

Return values: **=<Pin>,<Mode>,<Interval>,<ActiveState>,<Priority>**

Parameter description for **<Mode=4>:**

<Pin> defines which pin is affected by changes.

Range: 0 .. 3
0 = DIO_0
1 = DIO_1
2 = DIO_2
3 = DIO_3

<StartState> defines whether the pin shall start with High or Low

1: start with High
0: start with Low

<OType> defines the output type

0: push pull (no pull)
1: open drain

<Repetitions> defines the number of cycles to be done

255: repeat forever
0: stop (in any case)
Range: 1 .. 254 number of repetitions

 © 2019 All Rights Reserved

<TimeHigh> defines time in ms the output shall be High

Range: 1 .. 65535

<TimeLow> defines time in ms the output shall be Low

Range: 1 .. 65535

<Status> gives the current status

Range: 0 .. 1

0: inactive

1: active

Return values: **=<Pin>,<Mode >,<StartState>,<OType>,<Repetitions>,<TimeHigh>,<TimeLow>, <status>**

Note: To stop the endless repetitions all parameters must be provided

```
Example: GPIO 3 4 0 0 0 0 0
```

ASCII

Parameter (Format Mode=0/1):

<Pin> 1 byte (dec)

<Mode> 1 byte (dec)

<Speed> 1 byte (dec)

<OType> 1 byte (dec)

<PUPD> 1 byte (dec)

```
Example: GPIO 3 1 3 0 1
```

Return value (Format Mode=0/1):

<Pin> 1 byte (dec)

<Mode> 1 byte (dec)

<Speed> 1 byte (dec)

<OType> 1 byte (dec)

<PUPD> 1 byte (dec)

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

Parameter (Format Mode=3):

<Pin> 1 byte (dec)

<Mode> 1 byte (dec)

<Interval> 5 byte (dec)

<ActiveState> 1 byte (dec)

<Priority> 3 byte (dec)

```
Example: GPIO 1 3 30000 1 1
```

Return value (Format Mode=3):

<Pin> 1 byte (dec)

<Mode> 1 byte (dec)

<Interval> 5 byte (dec)

<ActiveState> 1 byte (dec)

© 2019 All Rights Reserved

<Priority> 3 byte (dec)

```
Example: =1,3,30000,1,1
```

Parameter (Format Mode=4):

<Pin> 1 byte (dec)

<Mode> 1 byte (dec)

<StartState> 1 byte (dec)

<OType> 1 byte (dec)

<Repetitions> 1 byte (dec)

<TimeHigh> 2 bytes (dec)

<TimeLow> 2 bytes (dec)

```
Example: SET GPIO 1 4 1 0 5 1000 1000
```

Return value (Format Mode=0/1):

<Pin> 1 byte (dec)

<Mode> 1 byte (dec)

<StartState> 1 byte (dec)

<OType> 1 byte (dec)

<Repetitions> 1 byte (dec)

<TimeHigh> 2 bytes (dec)

<TimeLow> 2 bytes (dec)

<Status> 1 byte (dec)

```
Example: =1,4,1,0,5,1000,1000,1
```

BINARY

Type: GET/SET

Parameter (Format Mode=0/1):

<Pin> 1 byte (uint8_t)

<Mode> 1 byte (uint8_t)

<Speed> 1 byte (uint8_t)

<OType> 1 byte (uint8_t)

<PUPD> 1 byte (uint8_t)

```
Example: SET GPIO 3 1 3 0 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x07	0x55	0x5A	0x0301030001	0x18	0x17

Return value (Format Mode=0/1):

<Pin> 1 byte (uint8_t)

<Mode> 1 byte (uint8_t)

<Speed> 1 byte (uint8_t)

<OType> 1 byte (uint8_t)

© 2019 All Rights Reserved

<PUPD> 1 byte (uint8_t)

```
Example: RESP GPIO 3 1 3 0 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x07	0x57	0x5A	0x0301030001	0x3b	0xd7

Parameter (Format Mode=2):

<Pin> 1 byte (uint8_t)

<Mode> 1 byte (uint8_t)

```
Example: SET GPIO 1 2
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x04	0x55	0x5A	0x0102	0x4b	0x81

Return value (Format Mode=2):

<Pin> 1 byte (uint8_t)

<Mode> 1 byte (uint8_t)

```
Example: RESP GPIO 1 2
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x04	0x57	0x5A	0x0102	0x4a	0x39

Parameter (Format Mode=3):

<Pin> 1 byte (uint8_t)

<Mode> 1 byte (uint8_t)

<Interval> 2 byte (uint16_t)

<ActiveState> 1 byte (uint8_t)

<Priority> 1 byte (uint8_t)

```
Example: SET GPIO 1 3 30000 1 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x08	0x55	0x5A	0x010375300101	0xf5	0x5f

Return value (Format Mode=3):

<Pin> 1 byte (uint8_t)

<Mode> 1 byte (uint8_t)

<Interval> 2 byte (uint16_t)

<ActiveState> 1 byte (uint8_t)

<Priority> 1 byte (uint8_t)

```
Example: RESP GPIO 1 3 30000 1 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x08	0x57	0x5A	0x010375300101	0x74	0x86

© 2019 All Rights Reserved

Parameter (Format GET):

<Pin> 1 byte (uint8_t)

```
Example: GET GPIO 0
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x54	0x5A	0x00	0x6e	0xfe

Return value depends on configured mode for this pin. See responses for Mode=1,2,3 examples.

Parameter (Format Mode=4):

<Pin> 1 byte (uint8_t)
<Mode> 1 byte (uint8_t)
<StartState> 1 byte (uint8_t)
<OType> 1 byte (uint8_t)
<Repetitions> 1 byte (uint8_t)
<TimeHigh> 2 bytes (uint16_t)
<TimeLow> 2 bytes (uint16_t)

```
Example: SET GPIO 1 4 1 0 5 1000 1000
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x0B	0x55	0x5A	0x010401000503E803E8	0x2d	0x25

Return value (Format Mode=4):

<Pin> 1 byte (uint8_t)
<Mode> 1 byte (uint8_t)
<StartState> 1 byte (uint8_t)
<OType> 1 byte (uint8_t)
<Repetitions> 1 byte (uint8_t)
<TimeHigh> 2 bytes (uint16_t)
<TimeLow> 2 bytes (uint16_t)
<Status> 1 byte (uint8_t)

© 2019 All Rights Reserved

5.6.12. SPIN

SPIN <Mask> <Status>:

Set new pin status.

Parameter description:

<Mask> defines which pins are affected by the changes.

Range: 00.. 0F

Bit 0 = DIO_0

Bit 1 = DIO_1

Bit 2 = DIO_2

Bit 3 = DIO_3

<Status>

Range: 00 ... 0F

0 = Bit reset

1 = Bit set

Return values: **=<Mask>,<Status>**

ASCII

Parameter (Format):

<Mask> 2 bytes (hex)

<Status> 2 bytes (hex)

Return value (Format):

<Mask> 2 bytes (hex)

<Status> 2 bytes (hex)

BINARY

Type: SET

Parameter (Format):

<Mask> 1 byte (uint8_t)

<Status> 1 byte (uint8_t)

Return value (Format):

<Mask> 1 byte (uint8_t)

<Status> 1 byte (uint8_t)

```
Example: SPIN 03 01

Example: =03,01


Example: SET SPIN 03 01
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x04	0x55	0x5b	0x0301	0x5b	0x20

```
Example: RESP SPIN 03 01
```

© 2019 All Rights Reserved

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x04	0x57	0x5b	0x0301	0x5a	0x98

5.6.13. GPIN

GPIN <void>:

Read (displays) the current pin status.

Parameter description:

<void>

Return values: =<Status> as mask. See also sect. 5.6.12.

<Status> Each bit indicates a pin status 0 = reset, 1 = set.

Range: 00 … 0F

ASCII

Parameter (Format):

```
Example: GPIN
```

Return value (Format):

<Status> 2 byte (hex)

```
Example: =01
```

BINARY

Type: GET

```
Example: GET GPIN
```

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x54	0x5B	0xc7	0x2f

Return value (Format):

<Status> 1 byte (uint8_t)

```
Example: RESP GPIN 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x03	0x56	0x5B	0x00	0xce	0xae

5.6.14. ICFG

ICFG <Setting>:

Configures and enables interrupt sources.

Parameter description:

<Setting>

Range: 0 … FFFF

Bit 0 = DIO_0 = Rising edge enabled/disabled

Bit 1 = DIO_0 = falling edge enabled/disabled

© 2019 All Rights Reserved

Bit 2 = DIO_1 = Rising edge enabled/disabled

Bit 3 = DIO_1 = falling edge enabled/disabled

Bit 4 = DIO_2 = Rising edge enabled/disabled

Bit 5 = DIO_2 = falling edge enabled/disabled

Bit 6 = DIO_3 = Rising edge enabled/disabled

Bit 7 = DIO_3 = falling edge enabled/disabled

Bit 8 = MEMS interrupt enabled/disabled

Return values: **=<Setting>**

ASCII

Parameter (Format):

<Setting> 4 byte (hex)

```
Example: ICFG 1
```

Return value (Format):

<Setting> 4 byte (hex)

```
Example: =1
```

BINARY:

Type: GET/SET

Parameter (Format):

<Setting> 2 bytes (uint16_t)

```
Example: SET ICFG 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x04	0x55	0x5C	0x0001	0xea	0x11

Return value (Format):

<Setting> 2 bytes (uint16_t)

```
Example: RESP ICFG 1
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x04	0x57	0x5C	0x0001	0xeb	0xa9

5.6.15. SMAI

SMAI <Interval> <Priority> <Timeout>:

Set MEMS alternative blink interval. The alternative <Interval> is active as long movement is detected. If the device does not move for longer than <Timeout>, the blink interval will be changed to the 'default' interval which is configured with the SBIV command. <Priority> is used to ensure the most urgent event takes control over the current blink interval. The smaller the value for <Priority> is the higher the priority. The reaction time will strongly depend on the settings of SMSL and SMDT.

Parameter description:

<Interval> defines the alternative blink interval, which takes effect once motion is detected.

Range: 0 … 65000 milliseconds.

<Priority> smaller values mean higher priorities.

© 2019 All Rights Reserved

Range: 0 … 255

<Timeout> defines the time in milliseconds after which the device will fall back to 'default' blink interval if no motion is detected.

Range: 0 … 65000 milliseconds.

Return values: **=<Interval>,<Priority>,<Timeout>**

ASCII

Parameter (Format):

<Interval> 5 byte (dec)

<Priority> 3 byte (dec)

<Timeout> 5 byte (dec)

```
Example: SMAI 100 5 9000
```

Return value (Format):

<Interval> 5 byte (dec)

<Priority> 3 byte (dec)

<Timeout> 5 byte (dec)

```
Example: =100,5,9000
```

BINARY:

Type: GET/SET

Parameter (Format):

<Interval> 2 byte (uint16_t)

<Priority> 1 byte (uint8_t)

<Timeout> 2 byte (uint16_t)

```
Example: SET SMAI 100 5 9000
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x07	0x55	0x5D	0x0064052328	0x7b	0x83

Return value (Format):

<Interval> 2 byte (uint16_t)

<Priority> 1 byte (uint8_t)

<Timeout> 2 byte (uint16_t)

```
Example: RESP SMAI 100 5 9000
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x07	0x57	0x5D	0x0064052328	0x58	0x43

5.6.16. SADC

SADC <resistance1><resistance2>

Sets the value of the two resistors forming the external voltage divider in kilo-Ohm. These values are used to calculate the voltage divider ratio needed to provide the output voltage of the command GADC. See section 5.6.17

© 2019 All Rights Reserved

Parameter description

<resistance1> This is a 2-byte (or 16-bit) resistance value in kOhms.

<resistance2> This is a 2-byte (or 16-bit) resistance value in kOhms.

Range: 1 ... 65535 kOhm (dec)

Return value description: = **<resistance1>,<resistance2>**

ASCII

Parameter(Format):

<resistance1> 1 to 5 bytes (dec)

<resistance2> 1 to 5 bytes (dec)

```
Example: SADC 60000 20000
```

Return value(Format):

<resistance1>,<resistance2> 2 to 10 bytes (dec)

```
Example: =60000,20000
```

BINARY

TYPE: GET/SET

Parameter(Format):

<resistance1> 2 byte (uint16_t)

<resistance2> 2 byte (uint16_t)

```
Example: SET ADC 60000 20000
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x06	0x55	0x5E	0xEA604E20	0x84	0xD2

Return value(Format):

<resistance1> <resistance2> 4 byte (uint16_t[2])

```
Example: RESP SADC 60000,20000
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x06	0x57	0x5E	0xEA604E20	0x85	0x30

5.6.17. GADC

GADC <void>

This will return the 16 bit decimal voltage (dV) measured at ADC_IN multiplied by the voltage divider ratio given by SADC in section 5.6.16.

Parameter description: **<void>**

Return value description: =<voltage>,<resistance1>,<resistance2>

© 2019 All Rights Reserved

<voltage> 2 byte (16 bit) voltage measured at ADC_IN multiplied by the voltage divider ratio given by SADC.

Range: 0 ... 65535 deci-Volts (dec) E.g. 45 -> 4.5 Volt

<resistance1> 2 byte resistance1 value from voltage divider set by the user using SADC command

<resistance2> 2 byte resistance2 value from voltage divider set by the user using SADC command

ASCII

Parameter(Format):

<void>

```
Example: GADC
```

Return value(Format): =<voltage>,<resistance1>,<resistance2>

<voltage> 1 to 5 bytes (dec)

<resistance1> 1 to 5 bytes (dec)

<resistance2> 1 to 5 bytes (dec)

```
Example: =45,2700,2200
```

BINARY

TYPE: GET/SET

Parameter(Format):

<void>

```
Example: GET ADC
```

SYN	LEN	TYPE	CMD	CRC_LOW	CRC_HIGH
0x7f	0x02	0x54	0x5E	0x07	0x2c

Return value(Format):

<voltage> 2 byte (uint16_t)

<resistance1> 2 byte (uint16_t)

<resistance2> 2 byte (uint16_t)

```
Example: RESP GADC 45 2700 2200
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x08	0x57	0x5E	0x002D0A8C0898	0x07	0x57

© 2019 All Rights Reserved

6. AIR Interface (additional commands)

6.1. SSTART

SSTART <Time>:

OPCODE: 0x23

Force receiver active for a certain time at remote node. This is useful to change multiple commands one after another without waiting for an RxSlot.

Parameter description:

<Time> force active receiver for n milliseconds

Range: 1 … 65000

Parameter (Format):

<Time> 2 bytes (uint16_t)

Return:

<Time>

Example via ASCII SDAT: Send air command SSTART to force 1000ms receive active.

1. SDAT 1 000000000011 0B 08125554025500032303e8 60000
2. =4130055640
3. *SDAT:000000000011,0,4130055640
4. *AIR:000000000011,23,57,02,03e8

6.2. SEXTEND:

OPCODE: 0x24

Refresh timeout set by SSTART to extend receiver active window.

Parameter description:

NO PARAMETER

Return:

<ErrorCode>

0 = successful extended

1 = failed to extend

Example via ASCII SDAT: Send air command SEXTEND immediately.

1. SDAT 0 000000000011 09 081255540255000124
2. =0
3. *AIR:000000000011,24,57

© 2019 All Rights Reserved

6.3. SSTOP:

OPCODE: 0x25

Stop streaming to this device. The remote device immediately turn back to normal operation regarding the receiver active state.

Parameter description:

NO PARAMETER

Return:

NO PARAMETER

Example via ASCII SDAT: Send air command SSTOP immediately.

1. SDAT 0 000000000011 09 081255540255000125
2. =0
3. *AIR:000000000011,25,57

6.4. MRATO:

Not supported.

© 2019 All Rights Reserved

7. Notification messages

This chapter describes the format of all asynchronous messages.

7.1. 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>**

Parameter description:

<ID> returns ID of node which sent message

Range: 000000000001 … FFFFFFFFFFFE

ASCII:

Parameter(Format)

<Data Notification Flag> 4 bytes starting with "*" to signal identification flag with content *DNO

<ID> 12 bytes (hex)

```
Example: *DNO:1F3CFF322133
```

BINARY:

Type: NOTI

CMD: Message type DNO (0x60)

Parameter (Format):

<ID> 6 bytes (uint8_t[6])

```
Example: NOTI DNO 1F3CFF322133
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x08	0x61	0x60	0x1F3CFF322133	0xcd	0x69

7.2. Format for Notification Messages of Node ID Broadcast with User Data

This chapter describes the communication structure for notification of Node ID broadcast with user data.

Notification format: **<Node ID Data Notification Flag>** (fixed): **<ts>,<ID>,<len>,<data>**

Parameter description:

<ts> timestamp upon receiving notification in milliseconds (dec)

<ID> returns ID of node which sent message

Range: 000000000001 … FFFFFFFFFFFE

<len> length of **<data>**

<data> user data

© 2019 All Rights Reserved

ASCII:

Parameter (Format)

<Data Notification Flag> 4 bytes starting with "*" to signal identification flag with content *DNI

<ts> 1… 10 bytes (dec)

<ID> 12 bytes (hex)

<len> 2 bytes (hex)

<data> depends on **<len>**

```
Example: *DNI:5955512,000000000001,02,AFFE
```

BINARY:

Type: NOTI

CMD: Message type DNI (0x66)

Parameter (Format):

<ts> 4 bytes (uint32_t) in [ms]

<ID> 6 bytes (uint8_t[6])

<len> 1 byte (uint8_t)

<data> 0 ... 128 byte

```
Example: NOTI DNI 40209(dec) 000000000001 02 AFFE
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x0f	0x61	0x66	0x00009d1100000000000102AFFE	0x58	0x20

7.3. 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>**,**<NCFG>,<DataNCFG>**

Parameter description:

<ID> returns ID of node which sent message

Range: 000000000001 … FFFFFFFFFFFE

<NCFG> Defines which data follows within this value. For details see at NCFG command for more information.

<DataNCFG> Depends on command NCFG and defines the sensor data transmitted to the host. The data is ordered according to the bit numbering: Example: <NCFG> = 5 (1+4) Bit 1 and Bit 2 is transmitted from swarm node to host. After <NCFG> in the example follow MEMS- Values and then RSSI value. If a value isn't present because MEMS is disabled "?" is transmitted in ASCII mode and MAX value from corresponding data type in BINARY mode.

© 2019 All Rights Reserved

ASCII:

Parameter(Format)

<Node ID Notification Flag> 4 bytes starting with "*" to signal identification flag with content *NIN

<ID> 12 bytes (hex)

<NCFG> 4 bytes (hex)

<DataNCFG> depends on <NCFG>-Mask

```
Example: *NIN:1F3CFF322133,04,-56
```

BINARY:

Type: NOTI

CMD: Message type NIN (0x61)

Parameter (Format):

<ID> 6 bytes (uint8_t[6])

<NCFG> 2 bytes (uint16_t)

<DataNCFG> depends on **<NCFG>**-Mask

```
Example: NOTI NIN 1F3CFF322133 04 -56
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x09	0x61	0x61	0x1F3CFF32213304C8	0xe5	0xff

7.4. Format for Ranging Result Notification Messages

This chapter describes the communication structure for Ranging Results Notification when it 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:

Ranging Notification Flag**:<SrcID>, <DestID>, <ErrorCode>, <Distance>, <NCFG>, <DataNCFG>**

Parameter description:

<SrcID> Returns ID of node which requested ranging

Range: 000000000001 … FFFFFFFFFFFE

<DestID> Returns ID of node which received ranging request

Range: 000000000001 … FFFFFFFFFFFE

<ErrorCode> indicating status of ranging operation

Range: 0 … 6

0 = success, ranging result valid

1 = ranging to own ID

2 = no hardware ack received

3 = ranging unsuccessful, ranging timer expired

5 = user timer expired, message could not be delivered

6 = medium blocked CSMA timer expired

<Distance> returning the measured ranging distance in centimeter. Range depends on mode.

© 2019 All Rights Reserved

Range(ASCII): 000000 … 999909 ranging distance in centimeter [cm]

Range(BINARY): 0...99999 ranging distance in centimeter [cm]

<NCFG> Defines which data follows within this value. For details see at NCFG command for more information.

<DataNCFG> Depends on command NCFG and defines the sensor data transmitted to the host. The data is ordered according to the bit numbering: Example: <NCFG> = 5 (1+4) Bit 1 and Bit 2 is transmitted from swarm node to host. After <NCFG> in the example follow MEMS- Values and then RSSI value. If a value isn't present because MEMS is disabled "?" is transmitted in ASCII mode and MAX value from corresponding data type in BINARY mode.

ASCII:

Parameter(Format)

<Ranging Notification Flag> 4 bytes starting with "*" to signal identification flag with content *RRN

<SrcID> 12 bytes (hex)

<DestID> 12 bytes (hex)

<ErrorCode> 1 byte (dec)

<Distance> 6 bytes (dec) in [cm]

<NCFG> 4 bytes (hex)

<DataNCFG> depends on <NCFG>-Mask

```
Example: *RRN:1F3123123133,1F3CFF322133,0,001843,04,-56
```

BINARY:

Type: NOTI

CMD: Message type RRN (0x62)

Parameter (Format):

<SrcID> 6 bytes (uint8_t[6])

<DestID> 6 bytes (uint8_t[6])

<ErrorCode> 1 byte (uint8_t)

<Distance> 4 bytes (uint32_t) in [cm]

<NCFG> 2 bytes (uint16_t)

<DataNCFG> depends on <NCFG>-Mask

```
Example: NOTI RRN 1F3123123133 1F3CFF322133 0 1840 04 -56
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x13	0x61	0x62	0x1F31231231331F3CFF3221330000073004C8	0xd8	0x70

7.5. Format for SDAT Notification Messages

This chapter describes the return message format for asynchronous SDAT Notifications after issuing a SDAT command with option 1. The SDAT notification has a variable delay related to the SDAT command. It depends when the corresponding blink has been received. The notification will indicate if the transmission was successful or not. See also [4].

© 2019 All Rights Reserved

Notification format: **<SDAT Notification Flag>**:**<ID>**,**<ErrorCode>**,**<PayloadID>**

Parameter description:

<ID> returns ID of node to which the message was sent

Range: 000000000001 … FFFFFFFFFFFE

<ErrorCode> indicating status of ranging operation

Range: 0 … 4

0 = success data communication valid

1 = no hardware ack received

2 = user timer expired, message could not be delivered

3 = medium blocked, CSMA timer expired

4 = unknown.

<PayloadID> used to identify the payload

Range: 000000000001 … FFFFFFFFFFFF

ASCII:

Parameter(Format)

<SDAT> Notification Flag> 5 bytes starting with "*" to signal identification flag with content *SDAT

<ID> 12 bytes (hex)

<ErrorCode> 1 byte (dec)

<PayloadID> 8 bytes (hex)

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

BINARY:

Type: NOTI

CMD: Message type SDAT (0x63)

Parameter (Format):

<ID> 6 bytes (uint8_t[6])

<ErrorCode> 1 byte (uint8_t)

<PayloadID> 4 bytes (uint8_t[4])

```
Example: NOTI SDAT 1F3CFF322133 0 45A6213F
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x0D	0x61	0x63	0x1F3CFF3221330045A6213F	0xCE	0xCB

7.6. Format for AIR Notification Messages

This chapter describes the return message format for asynchronous responses after issuing a SDAT command with the AIR interface protocol.

Notification format: **<AIR Notification Flag>**:**<ID>**,**<C_OPCODE>**,**<C_TYPE>,<LEN>,<DATA>**

© 2019 All Rights Reserved

Parameter description:

<ID> returns ID of remote node

Range: 000000000001 … FFFFFFFFFFFE

<C_OPCODE> indicating command opcode

Range: 00 .. FF

<C_TYPE> used to identify if it's a SET, GET, or ERROR message

Range: 00 .. FF
0x54 = GET request
0x55 = SET request
0x56 = GET response
0x57 = SET response
0x60 = ERROR

<LEN> indicating length of <DATA> field.

Range: 00 .. 7F

<DATA> command data, or if it's an error message, the error code. If no data appended, this field is missing.

Range: 00 .. FF
if C_TYPE is ERROR:
0x02 = unknown command opcode
0x03 = parameter error

ASCII:

Parameter(Format)

<AIR> Notification Flag> 4 bytes starting with "*" to signal identification flag with content *AIR

<ID> 12 bytes (hex)

<C_OPCODE> 2 byte (hex)

<C_TYPE> 2 byte (hex)

<LEN> 2 byte (hex)

<DATA> 2 byte (hex)

```
Example: *AIR:000000000011,05,56,01,3f
```

BINARY:

Type: NOTI

CMD: Message type AIR (0x64)

Parameter (Format):

<ID> 6 bytes (uint8_t[6])

<C_OPCODE> 1 byte (uint8_t)

<C_TYPE> 1 bytes (uint8_t)

<LEN> 1 bytes (uint8_t)

<DATA> (n) bytes (uint8_t[n])

© 2019 All Rights Reserved

```
Example: NOTI AIR 000000000011 05 56 01 3f
```

SYN	LEN	TYPE	CMD	CMD_DATA	CRC_LOW	CRC_HIGH
0x7f	0x0b	0x61	0x64	0x000000000110556013f	0xaf	0x19

© 2019 All Rights Reserved

8. API Default Settings

When starting the *swarm* Radio the following default settings are valid:

Parameter/ Command	Relevance	Default Value	Description
BRAR	Common	1	Broadcast ranging results is enabled.
EAIR	Common	1	Enable AIR configuration interface
EBID	Common	1	ID broadcast enabled
EBMS	Common	1	Broadcast MEMS data is enabled
EDAN	Common	1	Data notification enabled
EDNI	Common	0	Notification of Node ID broadcast with user data is disabled.
EIDN	Common	0	Node ID notification is disabled
EMSS	Common	1	MEMS sensor is enabled
EPRI	Common	0	Responds to ranging request is true
ERRN	Common	1	Ranging result notification is enabled
GPIO	Common	0, 3, 0, 1	All inputs are with Pull-up
ICFG	Common	0	All interrupts are disabled
NCFG	Common	0004	Notification configuration is set to RSSI only
SBIV	Common	30000	ID broadcast interval is 30000 ms
SDCL	Common	1	Device class is 1
SFEC	Common	0	FEC is disabled
SMAI	Common	100, 1, 60000	Alternative blink interval 100 ms, fall back time 60000 ms, priority 1
SADC	Common	2700 2200	R1=2.7 MΩ; R2=2.2 MΩ
SMBW	Common	6	MEMS sensor's bandwidth is 250 Hz
SMDT	Common	01000	MEMS sensor’s dead time is 1000 ms.
SMRA	Common	2	MEMS acceleration scale (g range) is 4 g
SMSL	Common	01	MEMS sensor's sleep time is 0.5 ms
SMTH	Common	30	MEMS sensor’s threshold is set to 30
SNID	Common	0000xxxxxxxx	Factory pre-configured MAC address
SPSA	Common	0	Power saving is disabled.
SROB	Common	001	Range on broadcast to device class 1
SROF	Common	000	Range offset is 0
SRXO	Common	001	RX occurrence is 1 (RX window after every node ID blink)
SRXW	Common	00010	RX window is 10 ms
SUAS	Common	115200	UART Speed
STXP	SB LE	63	TX power is 63.

© 2019 All Rights Reserved

Parameter/ Command	Relevance	Default Value	Description
SSYC	SB LE	1	Syncword is 1
SDAM	SB LE	1	80/1 mode enabled
CSMA (Note 2)	SB LE	0, 0, 0	Disabled (mode 0, random seed 0, threshold 0)
STPD	SB ER	0, 0	Smart power disabled and 0 dBm gain
SDMD	SB ER	15	Channel 5, 6.8MHz symbol rate
SDMC	SB ER	1 1 3 1 1 1 1 2 0 321 227	Channel 1, 6.8MHz symbol rate
SOFF	SB ER	SDMD 0 is 0 SDMD >0 around 15550	Ranging distance offsets for each data mode
SPAN	SB ER	0x6e6e	PAN ID

The default values are based on the following FW Version:

- swarm bee LE V2: Version 3.0.10
- swarm bee ER: Version 3.1.2

Note 1: The default settings are subject to change and may be changed without notice. Therefore it is recommended to configure at least once the swarm module and save the settings.
Note 2: For better performance of swarm bee LE it is recommended to enable CSMA. For regional and or country settings, please refer to the following application note AN0509 [1] to comply with local regulations. swarm bee ER doesn't support CSMA.
Note 3: The default value of SOFF is approximately 15550. It depends on the used data mode and has been set for each by the mean offset which has to be subtracted from measured distance referenced at the point defined by the RF port pin 13 of the swarm bee module. With the command GOFF the current values can be read.

9. Settings for Different Node Behaviors

The *swarm* radio settings are predefined with some default values such that the user only needs to connect power to start working with it. Nevertheless, as the default settings may change without notice, it is recommended to configure at least once the swarm module and save the settings. This default behavior is as follows:

Every 30s the *swarm* device 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 if not disabled. 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 node ID blink may start a ranging operation with the swarm if not disabled. In this case, the swarm answers by sending all necessary packets.

- A range result broadcast: After a ranging operation the swarm bee modules 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 bee 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 changed using the API commands. Some of the possible changes are:

© 2019 All Rights Reserved

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

- The swarm radio can be set in low power mode: The swarm can be set in low power 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 device is in this mode it indicates this in each blink so that all the neighbors have knowledge about it. So, when they need to send something they need to wait until reception of its blink. The time during which the swarm is listening after its blink and whether it listens after each blink or after every n^{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 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). Similarly it is possible to set a device in privacy mode (API command EPRI) so that it does not accept ranging requests. 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 devices in range. For the two operations the transmitting swarm bee needs to take into account that the receiving swarms module may be in sleep mode. If they are, they should send the message in delayed mode, that is, every time one of the receivers sends a blink.

10. Differences between API V2.1 and V3.0

1. SUAS -> New command. Set UART Speed. Sect. 5.1.12
2. SMAI -> New command. Sets MEMS alternative blink interval. Sect.5.6.15
3. GPIO -> Completely refurbished. No mask. Each pin has to be programmed separately and depends on SPSA mode. Sect. 5.6.11
4. SSYC -> Range extended from 0-8 to 0-12. Sect. 5.5.2.2
5. EAIR -> New command. En-/Disables the air interface. Sect. 5.1.13
6. GSET -> This command generates a response which has more line numbers than in the API V2.1. The GPIO displays the setting for each GPIO separately and it contains new parameters. It is recommended to use a parser as in future the number of lines and the parameters may change
7. SADC and GADC -> New command. From FW version 3.0.5-3 swarm bee LE V2 and version 3.1.1 swarm bee ER. See sections 5.6.16 and 5.6.17
8. EDNI -> New command. From FW version 3.0.3-10 swarm bee LE V1, version 3.0.7 swarm bee LE V2 and version 3.1.2 swarm bee ER. Sect. 5.3.2

SPSA 3 is a new power mode which allows to reduce further the power consumption to the lowest possible value depending on how the GPIOs and MEMS are used and configured. Therefore, it is very important to be very conscious how the GPIOs, MEMS and interrupts are set in this mode. There is a lot of potential to save power consumption. More information can be found in the application notes [5] and [6].

The swarm bee ER API and swarm bee LE V2 starts from API V3.0.

© 2019 All Rights Reserved

11. References

[1] App Note – AN0509 API Country settings for swarm bee LE certification
http://nanotron.com/EN/SU_support_appnotes.php

[2] App Note – AN0508 How to Adjust and Measure the RF Output Power on swarm bee LE
http://nanotron.com/EN/SU_support_appnotes.php

[3] DW 1000 User Manual
http://decawave.com/support

[4] App Note – AN0520 How to interpret Errors on Notifications
http://nanotron.com/EN/SU_support_appnotes.php

[5] App Note – AN0511 swarm bee Alternative Blink Interval
http://nanotron.com/EN/SU_support_appnotes.php

[6] App Note – AN0513 swarm bee LE V1 Power Modes
http://nanotron.com/EN/SU_support_appnotes.php

[7] nanoLOC User Guide

© 2019 All Rights Reserved

Document History

Date	Author	Version	Description
2016-03-04	MBO	3.0	First Release of API V3.0
2016-03-09	MBO	3.0.1	Added missing commands in table of sect. 4.4, editorial
2016-04-04	MBO	3.0.2	The commands in Table in sect. 4.4 have an hyperlink and are pointing to their respective sections, editorial
2016-04-05	MBO	3.0.3	Sect. 7.2 some text under Binary was missing
2016-04-15	MBO	3.0.4	Notes added in sect. 4.2.3 and 5.3.2 to avoid unintentional switching to AIR interface. Added note for GSET to point out variable length response.
2016-05-17	MBO	3.0.5	Removed MRATO as not implemented
2016-08-16	MBO	3.0.6	• Commands and responses of swarm bee ER have been merged to this document. • STXP, SSYC, SFEC, SDAM and CSMA have been moved to the new swarm bee LE specific section 5.5.2 • swarm LE specific commands are in the new section 5.5.3 • New shape of heading to improve readability of the table of contents • The commands in sect. 4.3 have an hyperlink and are pointing to their respective sections • The command SPAN has been added in sect. 5.5.3.5 • SDMD; rxPAC range wrong • Table 2-7 updated
2017-05-30	MBO	3.0.7	• Editorial • The command SDMC got a new parameter <gain> thus the default values for this command in the table of section 8 have been changed • Note be cautious when using the SDMC command • The commands SDAT, BDAT, FRAD and FNIN shows different ranges depending on module type • SDAT got an additional note on buffering data • GADC delivers now also the two resistors values • Added Note 3 in section 8 • Added command EDNI • Added note for timeout in CSMA
2017-06-16	MBO	3.0.8	• Contains several editorial changes after review of 3.0.7
2017-07-06	MBO	3.0.9	• Editorial • Provided more explanations for certain commands • Improved description of RATO • Updated sect. 10 • Added GPIO mode 4 (Twinkle Mode)
2018-02-23	MBO	3.0.10	• Editorial • Changed Blink interval to Aperture time in sect. 5.5.1.1 • Changed in Table 2-10 Blink rate from disabled to 30 s • Aligned SB ER default values

© 2019 All Rights Reserved

			• Added details in sect. 5.6.10 • Restricted Node ID range for correct functioning with nanoLES 3 RTLS
2019-03-07	MBO	3.0.11	• Changed the minimum blink rate to 50 ms in sect. 5.4.2

© 2019 All Rights Reserved

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 interoperable location 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