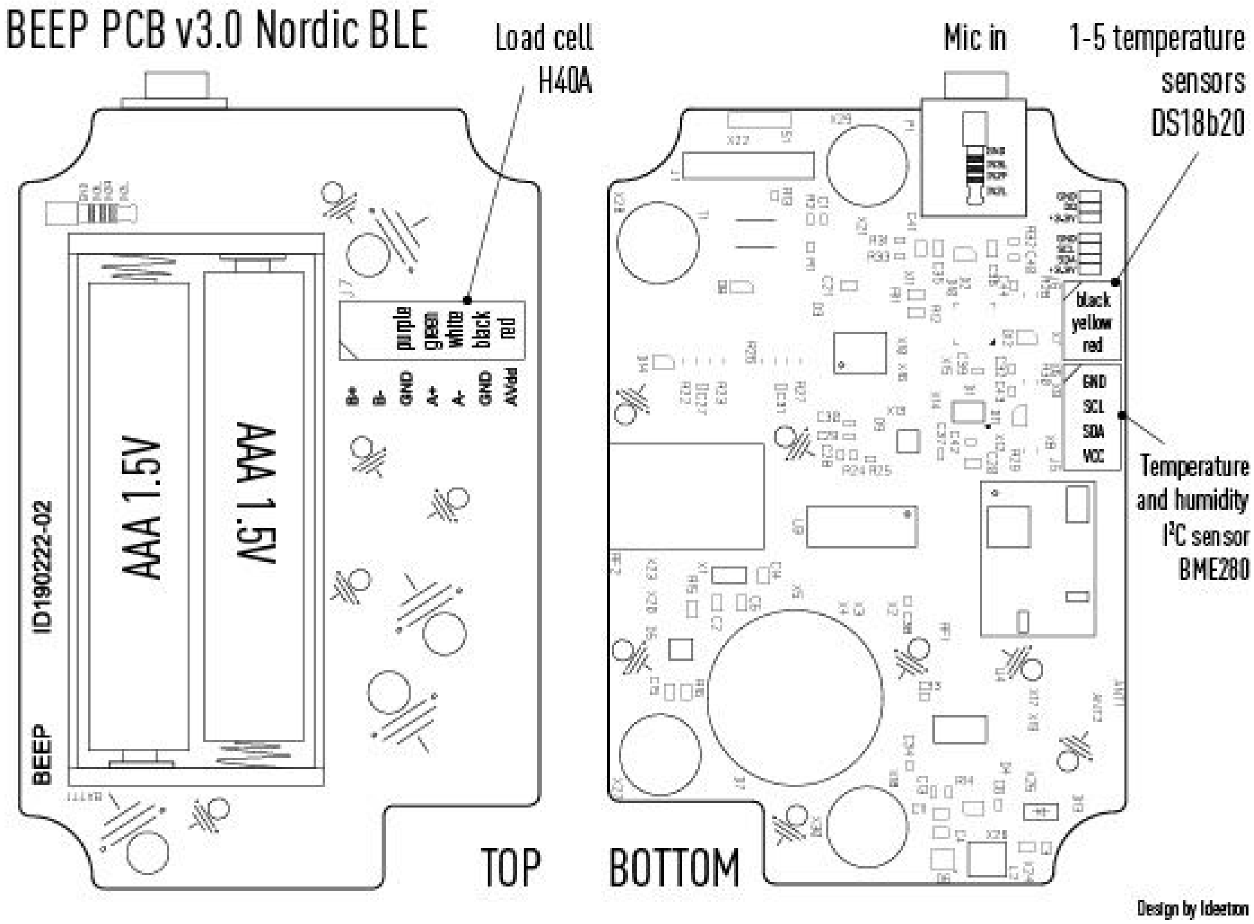
BEEP base - PCB ID190222-02
Firmware manual



Date:	08-01-2020
Version:	1.3.3
Title:	BEEP base - ID190222-02 - Firmware - Manual

Document revision and distribution

Document revision:

Version	Date	By	Changes
1.0	31-10-2019	Adri	1st edition: Introduction, Hardware, Software added.
1.1	4-11-10	Adrienne	HX711 command modified, added HX711 measurement configuration, buzzer commands added, Programming chapter is added.
1.2	29-11-2019	Adri	Pincode read / write assignments changed, Flash assignments added; read, erase, size, TX log characteristic added, flash log content. Added pin code reset.
1.3	12-12-2019	Pim	Purely cosmetic and functional: Google Drive doc made to facilitate collaboration. Image on front page, title no longer has a version number.
1.3	18-12-2019	Adri	Version number not increased, so this is the same as the firmware version. Added: - BME280 BEEP protocol messages - Alarm settings protocol messages - FFT description added
1.3	20-12-2019	Adri	Description flash erase adjusted with extra erase option
1.3.2	06-01-2020	Adri	Description BME280 alarm limits adjusted: difference alarm is now also switched off if the limit value is set to 0.
1.3.3	08-01-2020	Adri	Debug bootloader compilation script, release compilation and outputs and Segger Studio License description extended. SES code editor tab and indent settings changes added.

Table of contents

Introduction

The BEEP measurement system (BEEP base) is a system for monitoring a hive by means of weight, temperature and sound. All this measured data is logged by the BEEP base.

LoRaWAN is used to regularly send that to the BEEP back-end, alarm and to change settings remotely.

With bluetooth low energy, the logged data can be read out by the BEEP app. The BEEP app is also used for the initial configuration of the BEEP base. With the App the settings of the sensors and measurement intervals can be adjusted, but also the encryption keys for LoRaWAN can be adjusted.

Hardware

The hardware for this project has already been designed for BEEP: ID190222. The print has the following components:

- nRF52840 BLE low power microcontroller (BMD-340 module).
- 2x AA battery.
- DS18B20 temperature probe sensor with one-wire interface.
- HX711 double weighbridge sensor for measuring the weight of the hive.
- SQ-SEN-645 Tilt switch: horizontal and vertical detection.
- Reed switch for user activation
- TPS61292 boost converter.
- TPS22917 supply switch.
- Buzzer for audio feedback to the user.
- Flash for logging measurement data.
- RFM95 for LoRaWAN communication.
- ATECC608A encryption and unique key.
- BME280 measure temperature, humidity and air pressure.
- TLV320ADC3100 Electret signal conditioner and recorder for Fourier analysis.

nRF52840

The NordicnRF52840 microcontroller used to implement the functionality of the BEEP base. The nRF52840 has a radio module that supports low energy through the SDK of Nordic bluetooth.

AA batteries

To energize the electronics, two lithium AAA batteries from Energizer are used in series.

DS18B20 temperature sensor

To measure the temperature in different places in the hive, several DS18B20 temperature probes are used. These sensors use a one-wire protocol to set up the sensor, start a temperature conversion and read the result.

HX711 strain gauge sensor

The HX711 strain gauge sensor is used to measure the strain gauge on which the weight of the hive rests. With the measurement result and the sensitivity of the strain gauge, the weight of the hive can be calculated.

Reed switch

The user can activate the BLE communication by means of the reed switch. Optionally, the reed switch is also used to reset the pin code.

TPS boost converter and supply switch

The TPS61291 boost converter is used to increase the battery voltage to 3V if the battery voltage is lower. The boost converter can be switched off, after which the battery voltage is passed directly to the output.

Because not all components work or are specified for under 3V, a power switch has been used to disconnect those components from the power supply.

Buzzer

If BLE parameters are written or the sensor is placed in a new orientation, the buzzer is used for feedback to the user. The buzzer will only support a number of tones / melodies.

Flash storage

The MX25R6435 flash storage IC is used to store the measured data. The flash IC has a storage size of 64Mb. With BLE this can then be read out with the BEEP app.

RFM95

The RFM95 for the 868MHz EU band is used for LoRaWAN communication. An antenna can be connected through a micro UFL connector.

ATECC608A

The ATEC608A is a crypto authentication IC that supports various forms of encryption, decryption, hash calculations, a 72-bit unique serial number and storage of keys or certificates. For the BEEP base, however, only the unique serial number is used to hardware-identify the BEEP base in the back-end.

BME280

Temperature, humidity and air pressure sensor from Bosch that is mounted on a 1 meter long cable, so that it can be placed in the hive. Is controlled via I2C by means of the nRF52840.

TLV320ADC3100

The TLV320ADC3100 is an audio ADC that can feed and read out two electret microphones. The Audio measurement data is transported to the nRF52840 via an I2S interface. With an I2C interface, the audio ADC is set to the correct input and filter responses.

In the nRF52840 the measured audio data is converted with an FFT to amplitudes in a number of frequency bands, which is then logged.

Logging

The logging protocol has not yet been specified. In all likelihood this will be an ASCII protocol. Optionally a binary log in the Flash memory and translating to an ASCII format when reading.

Software

BEEP Protocol

The BEEP protocol is made up of complementary read and write commands that are identified by a single byte of which the seventh bit is always 1 for write commands. For example, the READ_DS18B20_CONVERSION command with the value 4d / 0x04h has a complementary write command 132d / 0x84h.

Below is a brief overview of the defined commands:

Dec / hex		Name	Description
0	0x00	RESPONSE	Response to a write
1	command0x01	READ_FIRMWARE_VERSION	Read the firmware version
2	0x02	READ_HARDWARE_VERSION	Read the hardware version
3	0x03	READ_DS18B20_STATE	Read the temperature resolution and status.
131	0x83	WRITE_DS18B20_STATE	Describe the temperature resolution and status.
4	0x04	READ_DS18B20_CONVERSION	Read the latest temperature conversion values
132	0x84	WRITE_DS18B20_CONVERSION	Start a temperature conversion
5	0x05	READ_DS18B20_CONFIG	AFTER
6	0x06	BME280_CONFIG_READ	AFTER
7	0x07	BME280_CONVersion	air pressure, air pressure, air pressure, air temperature, air pressure.
135	0x87	BME280_CONVERSION_START	Start a temperature, humidity and barometric pressure conversion.
8	0x08	READ_BME280_I2C	NA
9	0x09	READ_HX711_STATE	Read the HX711 conversion of channels and number of measurements per channel of which an average is calculated
137	0x89	WRITE_HX711_STATE	to change the conversion of channels and number of measurements per channel of which an average is calculated.
10	0x0A	READ_HX711_CONVERSION	Read the latest measurement result
138	0x8A	WRITE_HX711_CONVERSION	Start a new measurement on a channel
11	0x0B	READ_AUDIO_ADC_CONFIG	Read the audio channel, gain, volume and FFT settings.

139	0x8B	WRITE_AUDIO_ADC_CONFIG	Set the audio channel, gain, volume and FFT settings.
12	0x0C	READ_AUDIO_ADC_CONVERSION	Read the latest Audio ADC conversion.
13	0x0D	START_AUDIO_ADC_CONVERSION	Start an Audio ADC conversion with FFT calculation.
14	0x0E	READ_ATECC_READ_ID	Reads the ATECC ID from the flash.
15	0x0F	READ_ATECC_I2C	AFTER
16	0x10	READ_BUZZER_STATE	AFTER
17	0x91	WRITE_BUZZER_DEFAULT_TUNE	Plays a standard buzzer tune based on the specified index valueof the specified
18	0x92	WRITE_BUZZER_CUSTOM_TUNE	Plays a buzzer with the specified number of buzzer, and the timebuzzer tune.
19	0x13	READ_SQ_MIN_STATE	NA
20	0x14	READ_LORAWAN_STATE	Read the LoRaWAN status: on / off, joined, duty-cycle, Adaptive Data Rate, correct keys
148	0x94	WRITE_LORAWAN_STATE	Write the LoRaWAN status: on / off, duty-cycle, Adaptive Data Rate
21	0xANREAD_DEAU_AD15_AANREAD_DE_AU	READEADUDEAUAAD_UANREAD_AW READ_DE_AWREAD_DE_AU_READ_DE_AU_READ_DE_AU_EAD_:	Read DEVEUI, 8 bytes
149	0x95	WRITE_LORAWAN_DEVEUI	Be the DEVEUI, 8 bytes
22	0x16	READ_LORAWAN_APPEUI	read APPEUI, 8 bytes
150	0x96	WRITE_LORAWAN_APPEUI	Be the APPEUI, 8 bytes
23	0x17	READ_LORAWAN_APPKEY	read APPKEY, 16 byte
151	0x97	WRITE_LORAWAN_APPKEY	Be the APPKEY 16 bytes
136	0x88	WRITE_LORAWAN_TRANSMIT	Send a LoRaWAN message
25	0x19	READ_CID_nRF_FLASH	AFTER
26	0x1A	READ_nRF_ADC_CONFIG	NA
27	0x1B	READ_nRF_ADC_CONVERSION	Read the latest conversion values of the battery, nRF supply voltage.
155	0x9B	WRITE_nRF_ADC_CONVERSION	Start an ADC conversion
28	0x1C	READ_APPLICATION_STATE	AFTER

29	0x1D	READ_APPLICATION_CONFIG	Read the measurement interval and the ratio between measuring and sending.
15 7	0x9D	WRITE_APPLICATION_CONFI G	set the sampling interval and the ratio between measurement and send it.
30	0x1E	READ_PINCODE	Read the BLE pin code, 6 numbers: '0' - '9'
15 8	0x9E	WRITE_PINCODE	Write the BLE pin code, 6 numbers: '0' - '9'
31	0x1F	READ_BOOT_COUNT	Read the boot account of the BEEP base
32	0x20	READ_MX_FLASH	Command to read the log from the flash memory. An offset can be given so that you do not have to read the entire flash memory.
33	0x21	ERASE_MX_FLASH	Command to clear the log
34	0x22	SIZE_MX_FLASH	Command to read the size of the log
35	0x23	ALARM_CONFIG_READ Read	alarm configuration for a specific sensor.
16 3	0xA3 ALA RM_ CON FIG_ WRI TE	Set	alarm configuration for a specific sensor.
36	0x24 ALA RM_ STA TUS _RE AD	Read out	the current active Alarms.

Table: 1

All commands and values are in big endian.

For LoRaWAN it is possible to put multiple commands in a single message, for example:

Hex	Commands
0102	READ_FIRMWARE_VERSION, READ_HARDWARE_VERSION

With BLE, only 1 command per message will be executed and a message will have to be sent per command and the possible answer will have to be collected.
LoRaWAN maximum buffer size is 52 bytes and the BLE control point has a size of 30 bytes.

0d / 0x00 - RESPONSE

Response to a command with a status indication of the error or success. Only sent by the BEEP base

Field	Major	Value	Description
RESPON SE	Uint8_t	0x00	RESPONSE command ID
command	Uint8_t	-	The command ID to which a response is sent
Error code	Uint32_t	-	See the table below for the error code and the description

nRF SDK Error codes:

Error code	#	Description
NRF_SUCCESS	0	Successful command
NRF_ERROR_SVC_HANDLER_MISSI NG	1	SVC handler is missing
NRF_ERROR_SOFTDEVICE_NOT_E NABLED	2	Soft Device has not bone enabled
NRF_ERROR_INTERNAL	3	Internal Error
NRF_ERROR_NO_MEM	4	No Memory for operation
NRF_ERROR_NOT_FOUND	5	not found
NRF_ERROR_NOT_SUPPORTED	6	not supported
NRF_ERROR_INVALID_PARAM	7	Invalid Parameter
NRF_ERROR_INVALID_STATE	8	Invalid state, operation disallowed in this state
NRF_ERROR_INVALID_LENGTH	9	Invalid Length
NRF_ERROR_INVALID_FLAGS	10	InvalidFlags
NRF_ERROR_INVALID_DATA	11	Invalid Data
NRF_ERROR_DATA_SIZE	12	Invalid Data size
NRF_ERROR_TIMEOUT	13	Operation timed out
NRF_ERROR_NULL	14	Null Pointer
NRF_ERROR_FORBIDDEN	15	forb idden Operation
NRF_ERROR_INVALID_ADDR	16	Bad Memory Address
NRF_ERROR_BUSY	17	Busy
NRF_ERROR_CONN_COUNT	18	Maximum connection count exceeded.
NRF_ERROR_RESOURCES	19	Not enough resources for operation

Table: 2

If a command is sent that expects a few extra bytes, but too few bytes are sent with the command. Then the error code "Invalid Length" is sent back.

Example:

	Hex message	Contents
Command	0x96	WRITE_LORAWAN_APPEUI
Answer	0x009600000009	RESPONSE for WRITE_LORAWAN_APPEUI, error code: Invalid Length

If an unknown command is sent,

	Hex message	Content
Command	0xFE	No specified command
Answer	0x00FE00000005	RESPONSE for 0xFE, error

code_IR READ_FIRVED_IR_REF_IR_REF_IR_REF_IR_REF_IR_REF_IR_REF_IR_RED_F_ for: 1

The firmware version is read with this command.

Field	Great	Value	Description
READ_FIRMWARE_VERSION	uint8_t	0x01	READ_FIRMWARE_VERSIONcommand ID
Major	Uint16_t	-	Major Firmware number
Minor	Uint16_t	-	Minor Firmware number
Sub	Uint16_t	-	Firmware sub number

Example:

	Hex Message	Contents
Command	0x01	READ_FIRMWARE_VERSION command
Answer	0x01000000000001	READ_FIRMWARE_VERSION answer: Firmware Version 0.0.1

2D / 0x02 - READ_HARDWARE_VERSION

With this command the Hardware version and ID number are read.

Field	Great	Value	Description
READ_HARDWARE_VERSION	uint8_t	0x02	READ_HARDWARE_VERSION command ID
Major	Uint16_t	-	Hardware major number
Minor	Uint16_t	-	Hardware minor number
ID	uint32_t	-	Hardware ID number

Example:

	Hex Message	Contents
Command	0x02	READ_HARDWARE_VERSION command
Answer	0x02000100000002E70E	READ_HARDWARE_VERSION answer: Hardware version 1.0, ID = 190222

Firmware and read out hardware

	Hex message	Contents
Command	0x0102	READ_FIRMWARE_VERSION, READ_HARDWARE_VERSION command
Answer	010000000000102000100000002 E70E	READ_FIRMWARE_VERSION = 0.0.01 READ_HARDWARE_VERSION = 1.0, ID = 190222

3d / 0x03 -READ_ATE_REX_DATE_

Read out temperature.

Assignment:

Field	Major	Value	Description
READ_DS18B20_STATE	Uint8_t	0x03	READ_DS18B20_STATE assignment ID Answer2

:

Field	Major	Value	Description
READ_DS18B20_STATE	Uint8_t	0x03	READ_DS18B20_STATE assignment ID
status	Uint8_t		Bit [0] = On / Off: 0 = Off, 1 = On Temperature 1 Resolution 1 = 9 Bit resolution 2 = 10 Bit resolution 3 = 11 Bit resolution 4 = 12 Bit resolution Bit [4: 7] = unused

Example:

	Hex message	Contents
Command	0x03	READ_DS18B20_STATE
Answer	0x0309	0b0000 1001 = On / off = 1 and resolution = 4:12 bit resolution

131d / 0x83 - WRITE_DS18B20_STATE

Sets the temperature resolution and status.

Command:

Field	Great	Value	Description
WRITE_DS18B20_STATE	Uint8_t	0x83	WRITE_DS18B20_STATE command ID

status	Uint8_t		Bit [0] = On / Off: 0 = Off, 1 = On Bit [1: 3] = Temperature Resolution 1 = 9 Bit resolution 2 = 10 Bit resolution 3 = 11 Bit resolution 4 = 12 Bit resolution Bit [4: 7] = unused

Example:

	Hex message	Contents
Command	0x8309	WRITE_DS18B20_STATE, on / off = 1, 12-bit resolution
Answer	0x008300000000	

	Hex message	Contents
Command	0x8305	WRITE_DS18B20_STATE, on / off = 1 , 10-bit resolution
Answer	0x008300000000	

4d / 0x04 - READ_DS18B20_CONVERSION

Read the latest temperature conversion values of the connected DS18B20s.

Assignment:

Field	Great	Value	Description
READ_DS18B20_CONVERSION	uint8_t	0x04	READ_DS18B20_CONVERSION command ID

Answer:

Field	Great	Value	Description
READ_DS18B20_CONVERSION	uint8_t	0x04	READ_DS18B20_CONVERSION
N	uint8_t	<10	Number DS18B20 sensors
Temperature sensor	N * int16_t		MSB int16_t temperature hundredth degree accuracy

Example single temperature sensor:

	Hex Message	Contents
Assignment	0x04	READ_DS18B20_CONVERSION

Answer	0x04010898	READ_DS18B20_CONVERSION, 1 DS18B20, temperature, [0] = 0x0898 / 2200d = 22:00 ° C

Example plurality of temperature sensors:

	Hex message	Contents
command	0x04	READ_DS18B20_CONVERSION
Answer	0x0403089809C4FF9C	READ_DS18B20_CONVERSION, 3 DS18B20 sensors, temperature is [0] = 0x0898 / 2200d = 22:00 ° C temperature [1] = 0x09C4 / 2500d = 25.00 ° C temperature [2] = 0xFF9C / -10000d = -100.00 ° C

If a measured temperature is -100.0C, this means that the sensor had a communication error while starting the conversion or during reading. This can happen until now if the soft device is busy with a high priority action, such as writing or reading the flash.

132d / 0x84 - WRITE_DS18B20_CONVERSION

Start a temperature conversion for a single DS18B20 with a specified index or for all temperature sensors.

Assignment:

Field	Great	Value	Description
WRITE_DS18B20_CONVERSION	Uint8_t	0x84	WRITE_DS18B20_CONVERSION assignment ID
Index	Uint8_t	<10	DS18B20 index. For values below 10, only a specific sensor is measured. For values above 10, all sensors are measured and read out.

Answer:

Field	Great	Value	Description
READ_DS18B20_CONVERSION / WRITE_DS18B20_CONVERSION	uint8_t	0x04	READ_DS18B20_CONVERSION for all temperature sensors or WRITE_DS18B20_CONVERSION for a single specific sensor
index	uint8_t	<10 or 0xFF	for WRITE_DS18B20_CONVERSION gives this specific sensor value. For READ_DS18B20_CONVERSION this is the number of temperature sensors
Temperature sensor	N * int16_t		MSB of int16_t temperature in hundredths degrees accuracy

EXAMPLE single temperature sensor:

	HexMessage	Content
Assignment	0x8400	WRITE_DS18B20_CONVERSION, start a temperature conversion by the temperature sensor at index 0.
Answer1	0x008400000000	NRF_SUCCESS, conversion will be started . This message is not returned on the LoRaWAN interface if the error code is NRF_SUCCES.
Answer2	0x04000898	WRITE_DS18B20_CONVERSION, DS18B20 temperature sensor at index 0, temperature [0] = 0x0898 / 2200d = 22:00 ° C

EXAMPLE temperature index = 8, with only 2 sensors connected

	Hex Message	Contents
Assignment	0x8408	WRITE_DS18B20_CONVERSION, start a temperature conversion by the temperature sensor at index 8 with only 2 sensors connected.
Answer	0x008400000007	NRF_ERROR_INVALID_PARAM for command WRITE_DS18B20_CONVERSION

Example of all temperature sensors:

	Hex message	Contents
Command	0x84FF	WRITE_DS18B20_CONVERSION, start a temperature conversion with all connected temperature sensors.
Answer1	0x008400000000	NRF_SUCCESS, conversion starts. This message is not returned on the LoRaWAN interface if the error code is NRF_SUCCES.
Answer2	0x0403089809C4FF9C	READ_DS18B20_CONVERSION, 3 DS18B20sensors temperature[0] = 0x0898 / 2200d = 22:00 ° C temperature [1] = 0x09C4 / 2500d = 25.00 ° C temperature [2] = 0xFF9C / -10000d = -100.00 ° C.

If a measured temperature at -100.0C means that the sensor had a communication error while starting the conversion or while reading. This can happen until now if the soft device is busy with a high priority action, such as writing or reading the flash.

7d / 0x07 - BME280_CONVERSION_READ

Read the temperature, humidity and barometric pressure of the last conversion.

Mission:

Field	Great	Value	Description
BME280_CONVERSION_READ	uint8_t	0x07	BME280_CONVERSION_READ command ID

Answer:

Field	Great	Value	Description
BME280_CONVERSION_READ	uint8_t	0x07	BME280_CONVERSION_READ
Temperature	int16_t	-	Temperature in two decimal signed integer
relative humidity	uint16_t	-	Relative humidity in two decimal unsigned integer
Barometric pressure	uint16_t	-	Barometric pressure in hPa

Example:

	Hex message	Contents
Assignment	0x07	0x07 BME280_CONVERSION_READ
Answer	0x07086C11D602A8	BME280_CONVERSION_READ: Temp = 21.56 C, RH = 45.66%, Pressure = 680 hPa

7d / 0x87 -

BME280_Crep. When the conversion is finished, a BME280_CONVERSION_READ message is returned.

Mission:

Field	Great	Value	Description
BME280_CONVERSION_START	uint8_t	0x87	BME280_CONVERSION_START command ID

Answer:

Field	Great	Value	Description
BME280_CONVERSION_READ	uint8_t	0x07	BME280_CONVERSION_READ
Temperature	int16_t	-	Temperature in two decimal signed integer
relative humidity	uint16_t	-	Relative humidity in two decimal unsigned integer
Barometric pressure	uint16_t	-	Barometric pressure in hPa

Example:

	Hex message	Contents
Assignment	0x87	0x87 BME280_CONVERSION_START

Answer	0x07086C11D602A8	BME280_CONVERSION_READ: Temp = 21.56 C, RH = 45.66%, Pressure = 680 hPa the

9d and 07REHATEREADATE

The READATE Metx7X_STHATE the11the H117ST_STHATE the117HSTwith the H117STHATE the READATE Metx average is calculated read from the flash memory of the nRF52840. At every meeting at the measurement interval, these settings are used for the HX711.
Multiple channels can be set for which an average is calculated. For each channel, the set number of samples is measured and the average is calculated.

Measuring channel	Value
CH_A_GAIN128	0x01
CH_B_GAIN32	0x02
CH_A_GAIN64	0x04

Message structure:

Field	Great	Value	Description
READ_HX711_STATE	uint8_t	0x09	READ_HX711_STATE command ID

Answer:

Field	Great	Value	Description
READ_HX711_STATE	uint8_t	0x89	WRITE_HX711_CONVERSION command ID
Measurement channels	uint8_t	1-7	HX711 measurement channels, see table above
Number of samples	uint8_t>	0	Number of samples over which the average is calculated.

Example 1:

	Hex Message	Contents
command	0x09	READ_HX711_STATE
Answer	0x090102	0x09 = READ_HX711_STATE 0x01 = CH_A_GAIN128 0x02 = 2 samples per channel

Example 2:

	Hex Message	Contents
command	0x09	READ_HX711_STATE
Answer	0x09070A	0x09 = READ_HX711_STATE

		0x07 = CH_A_GAIN128, CH_B_GAIN32, CH_A_GAIN64 0x0A = 10 measurements per channel,

137d / 0x89 - WRITE_HX711_STATE

The WRITE_HX711_STATE command sets the HX711 channels and the number of samples over which an average is calculated. At every meeting at the measurement interval, these settings are used for the HX711. This data is stored in the flash of the nRF52840.

Message structure:

Field	Major	Value	Description
WRITE _HX711_STATE	Uint8_t	0x89	WRITE_HX711_CONVERSION command ID
Measuring channels	Uint8_t	1 - 7	HX711 measuring channels, see table above
Number of samples	Uint8_t	> 0	Number of samples over which the average is calculated.

Example:

	Hex Message	Contents
Command	0x89070A	WRITE_HX711_STATE 0x89 = 0x07 = CH_A_GAIN128, CH_B_GAIN32, CH_A_GAIN64 0x0A = 10 measurements per channel
Response	0x0089000000 00	RESPONSE_COMMAND 0x00 = 0x89 =WRITE_HX711_STATE 0x00000000 =NRF_SUCCESS

10d / 0x0A - READ_HX711_CONVERSION

Read the latest result with the HX711. From 1.1, the HX711 statemachine supports the measurement of the different channels behind one. Instead of the number of clock pulses, the channels are now passed on why the measurement was or must be done, and several measurement results follow in a single message. If a channel is not measured, no measurement result or 0 value is sent in the result message. In the table below the measuring channels can be found with the bit value for each channel.

Measurement channel	Value
CH_A_GAIN128	0x01
CH_B_GAIN32	0x02

CH_A_GAIN64	0x04

Message structure:

Field	Large	Value	Description
READ_HX711_CONVERSION	uint8_t	0x0A	READ_HX711_CONVERSION command ID

Answer:

Field	Large	Value	Description
READ_HX711_CONVERSION	uint8_t	0x0A	READ_HX711_CONVERSION command ID
Measurement channels	uint8_t	1-7	HX711 measuring channels, see the table above
measurement result	Int24_t	-	Signed measurement result CH_A_GAIN128, CH_B_GAIN32 or CH_A_GAIN64
measurement result (optional)	Int24_t	-	Signed measurement result CH_B_GAIN32 or CH_A_GAIN64
measurement result (optional)	Int24_t	-	Signed measurement result CH_A_GAIN64

Example:

	Hex message	Contents
Assignment	0x0A	READ_HX711_CONVERSION
Answer	0x0A010182E6	READ_HX711_CONVERSION, channel 0x01: CH_A_GAIN128: , result: 99046 / 0x0182e6

138d / 0x8A - WRITE_HX711_CONVERSION

Start a new average measurement on the specified channels. When the conversion is started, a confirmation of the command or an error code first follows if a parameter is wrong. When the measurement is complete, the results are sent back.

Measuring channel	Value
CH_A_GAIN128	0x01

CH_B_GAIN32	0x02
CH_A_GAIN64	0x04

Mission:

Field	Great	Value	Description
READ_HX711_CONVERSION	uint8_t	0x8A	WRITE_HX711_CONVERSION command ID
measurement channels	uint8_t	1-7	HX711 measurement channels, see table above
Number of samples	uint8_t>	0	Number of samples over which the average is calculated.

Answer:

Field	Great	Value	Description
READ_HX711_CONVERSION	uint8_t	0x8A	WRITE_HX711_CONVERSION command ID
Measurement channels	uint8_t	1-7	HX711 measurement channels, see table above
measurement result	Int24_t	-	Signed measurement result CH_A_GAIN128, CH_B_GAIN32 or CH_A_GAIN64
measurement result (optional)	Int24_t	-	Signed measurement result CH_B_GAIN32 or CH_A_GAIN64
measuring result (optional)	Int24_t	-	Signed measurement result CH_A_GAIN64

Example 1:

	Hex message	Contents
Assignment	0x8A010A	WRITE_HX711_CONVERSION, CH_A_GAIN128, 10 samples
Answer1	0x008A00000000	WRITE_HX711_CONVERSION, NRF_SUCCESS. Not returned with a command from the LoRaWAN interface if the error code is NRF_SUCCESS.
Answer2	0x8A010183A4	WRITE_HX711_CONVERSION = 0x8A 0x01 = CH_A_GAIN128 0x0183A4 = 99.236decimaal

Example 2:

	HexMessage	Contents

Assignment	0x8A070A	WRITE_HX711_CONVERSION, CH_A_GAIN128, CH_B_GAIN32, CH_A_GAIN64,10 samples
Answer1	0x008A00000000	WRITE_HX711_CONVERSION, NRF_SUCCESS. Not returned with a command from the LoRaWAN interface if the error code is NRF_SUCCESS.
Answer2	8A030183B90058D1	WRITE_HX711_CONVERSION = 0x8A 0x03 = CH_A_GAIN128, CH_B_GAIN32 CH_A_GAIN128: 99257 / 0x0183b9 CH_B_GAIN32: 22737 / 0x0058d1

Example 3:

	Hex Message	Contents
Assignment	0x8A010A	WRITE_HX711_CONVERSION, CH_A_GAIN128, CH_B_GAIN32, CH_A_GAIN64,10 samples
Answer1	0x008A00000000	WRITE_HX711_CONVERSION, NRF_SUCCESS. Not returned with a command from the LoRaWAN interface if the error code is NRF_SUCCESS.
Answer2	0x8A070183A700564100C1FE	WRITE_HX711_CONVERSION = 0x8A 0x03 = CH_A_GAIN128, CH_B_GAIN32 CH_A_GAIN128: 99257 / 0x0183b9 CH_B_GAIN32: 22737 / 0x0058d1 CH_A_GAIN64: 49662 / 0x00c1fe

11d / 0x0B - READ_AUDIO_ADC_CONFIG

Read the Audio ADC and FFT configuration.

Mission:

Field	Great	Value	Description
READ_AUDIO_ADC_CONFIG	uint8_t	0x0B	READ_AUDIO_ADC_CONFIG command ID

Answer:

Field	Great	Value	Description
READ_AUDIO_ADC_CONFIG	uint8_t	0x0B	READ_AUDIO_ADC_CONFIG
Audio channel	uint8_t		AIN_IN3LM = 0, AIN_IN2LP = 1 AIN_IN2RP = 2
+ Gain attenuator	uint8_t		bit [7]: 1 = 6dB , 0 = 0 dB bits [6: 0] = gain in 0.5 dB per bit: 80d / 0x50 = +40.0 dB 40d / 0x28 = +20.0 dB

			1d / 0x01 = 0.5 dB 0d / 0x00 = 0 dB
Volume	int8_t	-24 - 40	Volume in 0.5 dB per bit 40d / 0x28 = +20.0 dB -24d / 0xE8 = -12.0 dB
FFT bins	uint8_t	0 - 12	Number of bins that the FFT result is reduced to.
FFT start	uint8_t	0-255	bin Start time 2: 255 = 510
FFT stop	uint8_t	0-255	Stop bin times 2255= 510

Example:

	Hex Message	Contents
Command	0x0B	0x07 READ_AUDIO_ADC_CONFIG
Answer	0x0B0228000A00FF	See Table

Field	Great	Value	Description
READ_AUDIO_ADC_CONFIG	uint8_t	0x0B	READ_AUDIO_ADC_CONFIG
Audiochannel	uint8_t	02	AIN_IN2RP
attenuator Reinforcement	uint8_t	0x28	bit [7]: = 0dB 0 bits [6: 0] = 40d / 0x28 = +20.0 dB
Volume	int8_t	00	Volume 0 dB
FFT bins	uint8_t	0x0A	10 bins result
FFT start	uint8_t	0x00	Start bin 0
FFT stop	uint8_t	0xFF	Stop bin 510

139d / 0x8B - WRITE_AUDIO_ADC_CONFIG

Read the Audio ADC and FFT configuration.

Mission:

Field	Great	Value	Description
WRITE_AUDIO_ADC_CONFIG	uint8_t	0x8B	WRITE_AUDIO_ADC_CONFIGopdracht ID

Answer:

Field	Great	Value	Description
WRITE_AUDIO_ADC_CONFIG	uint8_t	0x8B	WRITE_AUDIO_ADC_CONFIG
Audio channel	uint8_t		AIN_IN3LM = 0,

			AIN_IN2LP = 1 AIN_IN2RP = 2
+ Gain attenuator	uint8_t		bit [7]: 1 = 6dB, 0 = 0dB bits [6: 0] = gain in 0.5 dB per bit: 80d / 0x50 = +40.0 dB 40d / 0x28 = +20.0 dB 1d / 0x01 = 0.5 dB 0d / 0x00 = 0 dB
Volume	int8_t	-24 - 40	Volume in 0.5 dB per bit 40d / 0x28 = +20.0 dB -24d / 0xE8 = -12.0 dB
FFT bins	uint8_t	0 - 12	Number of bins that the FFT result is reduced to.
FFT start	uint8_t	0-255	bin Start time 2: 255 = 510
FFT stop	uint8_t	0-255	Stop bin times 2255= 510

Example 1:

	Hex Message	Contents
Command	0x8B0228000A00FF	See table below
Answer	0x008B00000000	NRF_SUCCESS

Field	Great	Value	Description
WRITE_AUDIO_ADC_CONFI G	uint8_t	0x8B	WRITE_AUDIO_ADC_CONFIG
Audiochannel	uint8_t	02	AIN_IN2RP
attenuator Reinforcement	uint8_t	0x28	bit [7]: = 0dB 0 bits [6: 0] = 40d / 0x28 = +20.0 dB
Volume	int8_t	00	Volume 0 dB
FFT bins	uint8_t	0x0A	10 bins result
FFT start	uint8_t	0x00	Start bin 0
FFT stop	uint8_t	0xFF	stop bin 510

Example 1:

	Hex message	Contents
Command	0x8B0228000D00FF	See table below
Answer	0x008B00000007	NRF_SUCCESS

Field	Great	Value	Description
WRITE_AUDIO_ADC_CONFI G	uint8_t	0x8B	WRITE_AUDIO_ADC_CONFIG

Audio channel	uint8_t	02	AIN_IN2RP
attenuator Strengthening	uint8_t	0x28	bit [7]: 0 = 0dB bits [6: 0] = 40d / 0x28 = +20.0 dB
Volume	int8_t	00	Volume 0 dB
FFT	bin uint8_t	0x0D	13result
FFT start	uint8_t	0x00	BinStart bin 0
FFT st op	uint8_t	0xFF	Stop bin 510

12d / 0x0C - READ_AUDIO_ADC_CONVERSION

Read the latest Audio ADC FFT result. The Audio ADC takes a number of samples that are then converted with an FFT to a frequency spectrum of 512 bins. Each bin has a resolution of 3.937752016 Hz per FFT bin.

The start and stop indexes together with the number of bins to which the FTT is reduced, determine how many FFT bins are added. The calculation for this is as follows:

$$number\ of\ added\ bins\ =\ roundUp\,((((stop\ -\ start)\ \times 2) \div\ N)$$
$$number\ of\ added\ bins\ =\ roundUp\,(((255\ -\ 0)\ \times 2) \div\ 10)$$
$$number\ of\ added\ bins\ =\ roundUp\,(510\ \div\ 10)$$
$$number\ of\ added\ bins\ =\ 51$$

In this example, the number of summed bins is an integer. If this is not the case, in the last result bin the remaining number of bins are added up to the stop index.

The frequency for a result bin can be calculated as follows:

$$F bin\,[i\ =\ 10;\ 0:\ 9]\ =\ ((start\ *\ 2)\ +\ i\ *\ number\ of\ bins\ added)\ \times\ \ 3.937752016\ Hz\ /\ FFT\ bin$$

Example:

start = 0, stop = 255, $number\ of\ bins\ added\ =\ 51$

Result bin	Frequency
0	0.0 Hz
1	200.8 Hz
2	401.6 Hz
3	602.5 Hz
4	803.3 Hz
5	1004.1 Hz
6	1204.9 Hz
7	1405.8 Hz
8	1606.6 Hz

9	1807.4 Hz

mission:

Field	Great	Value	Description
READ_AUDIO_ADC_CONVERSI ON	uint8_t	0x0C	READ_AUDIO_ADC_CONVERSION command ID

Answer:

Field	Great	Value	Description
READ_AUDIO_ADC_CONVERS ION	uint8_t	0x0C	READ_AUDIO_ADC_CONVERSION
number of bins N	uint8_t	0 -12	Number of measurement results
Start bin	uint8_t	0-255	Start bin times 2
Stop bin	int8_t	0-255	Stop Bin times 2
Measurement results [N]	uint16_t [N]	0 - 65535	Array with N measurement results

Example 1:

	Hex message	Contents
Command	0x0C	
Response	0x0C0A00FF014E00560039001B0019001700100013001400 0F	TLV FFT [10: 0: 255] 0.0 Hz = 334, 203.2 Hz = 86, 406.4 Hz = 57, 609.6 Hz = 27, 812.8 Hz = 25, 1016.0 Hz = 23, 1219.2 Hz = 16, 1422.4 Hz = 19, 1625.7 Hz = 20, 1828.9 Hz = 15

Field	Great	Value	Description
R EAD_AUDIO_ADC_CONVERSI ON	uint8_t	0x0C	READ_AUDIO_ADC_CONVERSION
number of bins N	uint8_t	0x0A	10 measurement results
Start bin	uint8_t	0x00	Start bin 0
Stop bin	int8_t	0xFF	Stop Bin 510
Measurement results [N]	uint16_t [N]	0 - 65535	Array with N measurements

13d / 0x0D - START_AUDIO_ADC_CONVERSION

Start an Audio ADC conversion and an FFT calculation

task:

Field	Great	Value	Description
START_AUDIO_ADC_CONVERSION	uint8_t	0x0D	START_AUDIO_ADC_CONVERSION command ID

Answer:

Field	Great	Value	Description
READ_AUDIO_ADC_CONVERSION	uint8_t	0x0C	READ_AUDIO_ADC_CONVERSION
number of bins N	uint8_t	0-12	Number of measurement results
Start bin	uint8_t	0-255	bin Start time 2
Stop bin	int8_t	0- 255	Stop Bin times 2
Test results [N]	uint16_t [N]	0 - 65535	Array with N measurement results

Example 1:

	Hex message	Contents
Command	0x0C	
Answer	0x0C0A00FF012B008A004B0020001C001300110013001300 0D	TLV FFT [10: 0: 255] 0.0 Hz = 299, 203.2 Hz = 138, 406.4 Hz = 75, 609.6 Hz = 32, 812.8 Hz = 28, 1016.0 Hz = 19, 1219.2 Hz = 17, 1422.4 Hz = 1 9, 1625.7 Hz = 19, 1828.9 Hz = 13

Field	Great	Value	Description
READ_AUDIO_ADC_CONVERSION	uint8_t	0x0C	READ_AUDIO_ADC_CONVERSION
number of bins N	uint8_t	0x0A	10 measurement results
Start bin	uint8_t	0x00	Start bin 0

Stop bin	int8_t	0xFF	Stop Bin 510
Measurement results [N]	uint16_t [N]	0 - 65535	Array with N measurement results

145d / 0x91 - WRITE_BUZZER_DEFAULT_TUNE

The WRITE_BUZZER_DEFAULT_TUNE command plays a standard pwm pattern on the BEEP base. To date, there are only 3 patterns, but that can be expanded by the customer.

Message structure:

Field	Large	Value	Description
WRITE_BUZZER_DEFAULT_TUNE	uint8_t	0x91	WRITE_BUZZER_DEFAULT_TUNE ID assignment
pattern indexSound	uint8_t	0-2	

Sound Patterns

Pattern	Duty Cycle	Frequency	On-time	Off-time	repetition
0	50%	2.8 kHz	100 ms	1000ms	4
1	50%	2.8kHz	1000ms	1ms	1
2	50%	2.8 kHz	50ms	100ms	2

Example:

	Hex message	Content
Command	0x9101	0x91 = WRITE_BUZZER_DEFAULT_TUNE 0x01 = PWM pattern 1
Answer	0x009100000000	0x00 = RESPONSE_COMMAND 0x91 = WRITE_BUZZER_DEFAULT_TUNE 0x00000000 NRC0000000

146d / 0x92 - WRITE_BUZZER_CUSTOM_TUNE

With the WRITE_BUZZER_CUSTOM_TUNE command, a pwm pattern is played according to the given parameters.

Message structure:

Field	Great	Value	Description
WRITE_BUZZER_CUSTOM_TUNE	Uint8_t	0x92	WRITE_BUZZER_CUSTOM_TUNE command ID
Duty Cycle	Uint8_t	0-100	Duty Cycle in percentages

			1d = 1% 100d = 100%
Frequency in / 100 Hz	Uint8_t	1-255	1kHz =kHz = 10k 2dHz =d 2.8
Off time	Uint16_t	> 0	Time the PWM is off in milliseconds
On time	Uint16_t	> 0	Time the PWM is on in milliseconds
Repeats	iint16_t	> 0	Number of times the on-off cycle is repeated.

Example:

	Hex Message	Contents
Command	0x92321C03E801F403	WRITE_BUZZER_CUSTOM_TUNE 0x92 = 0x32 = Duty Cycle: 50% 0x1C = Frequency: 2.8kHz 0x03E8 = Off time: 1000ms 0x01F4 = On-time: 500ms = 0x03 Repetitions: 3
Reply	0x009200000000	RESPONSE_COMMAND 0x00 = 0x92 = WRITE_BUZZER_CUSTOM_TUNE 0x00000000 = NRF_SUCCESS

20d / 0x14 - READ_LORAWAN_STATE

Read the LoRaWAN status: on / off, joined, duty cycle, Adaptive Data Rate, correct keys.

Field	Great	Value	Description
READ_LORAWAN_STATE	Uint8_t	0x14	READ_LORAWAN_STATE command ID
Status	Uint8_t	-	See status table below for bit values.

Bit	Function	Bit value
0	On / off	0 = LoRaWAN off, 1 = LoRaWAN on
1	Joined	0 = Not yet joined, 1 = Network joined
2	Duty cycle restriction	0 = Duty cycle limit off, 1 = DutyCycle limit on
3	Adaptive Datarate	0 = ADR off, 1 = ADR on.
4	Keys correct	0 = Incorrect keys, 1 = Correct keys,
5: 7	Unused	Always 0

Example:

	Hex message	Contents
Command	0x14	READ_LORAWAN_STATE
Answer	0x141F	0x1F = 0001 1111b: LoRaWAN on, network joined, DutyCycle limitation on, ADR on, Correct keys

	Hex message	Contents
Assignment	0x14	READ_LORAWAN_STATE
Answer	0x141B	0x1F = 0001 1011b: LoRaWAN on, network joined, DutyCycle limit on, ADR off, Correct keys

148d / 0x94 - WRITE_LORAWAN_STATE

Write the LoRaWAN status: on / off, data rate, Adaptive. The LoRaWAN stack is reset after taking over the new parameters.

Field	Great	Value	Description
WRITE_LORAWAN_STATE	Uint8_t	0x94	WRITE_LORAWAN_STATE command ID
Status	Uint8_t	-	See status table below for bit values.

Bit	Function	Bit value
0	On / off	0 = LoRaWAN off, 1 = LoRaWAN on / off
1	Unused	Ignored
2	Duty cycle restriction	0 = Duty cycle limit off, 1 = Duty Cycle limit on
3	Adaptive Datarate	0 = ADR off, 1 = ADR on.
4: 7	Unused	Always 0. Ignored

Example to reset the LoRaWAN stack:

	Hex message	Contents
Command	940D	0x0F = 0000 1101b: LoRaWAN on, DutyCycle limit on, ADR on,
Answer	009400000000	NRF_SUCCESS for WRITE_LORAWAN_STATE

Example to test the LoRaWAN without Duty cycle limitation:

	Hex message	Contents
mission	9409	0x0F = 0000 1001b: LoRaWAN to, Duty Cycle limitation of ADR to,
Reply	009 400 000 000	NRF_SUCCESS for WRITE_LORAWAN_STATE

21d / 0x15 - READ_LORAWAN_DEVEUI

Read DEVEUI, 8 bytes

Field	Great	Value	Description
READ_LORAWAN_DEVEUI	uint8_t	0x15	READ_LORAWAN_DEVEUI command ID

Example

	Hexmessage	Contents
command	0x15	READ_LORAWAN_DEVEUI command
answer	0x150001020304050607	READ_LORAWAN_DEVEUI answer DEVEUI: 01020304050607

149d / 0x95 - WRITE_LORAWAN_DEVEUI

Be the DEVEUI, 8 bytes

Field	Great	Value	Description
WRITE_LORAWAN_DEVEUI	uint8_t	0x95	WRITE_LORAWAN_DEVEUIcommand ID
DEVEUI	8 x uint8_t	-	

Example:

	Hex message	Contents
command	0x9500010203040 50607	WRITE_LORAWAN_DEVEUI command
Answer	0x009500000000	WRITE_LORAWAN_DEVEUI successful.

22d / 0x16 - READ_LORAWAN_APPEUI

Read the APPEUI, 8 bytes

Field	Large	Value	Description
READ_LORAWAN_APPEUI	Uint8_t	0x16	READ_LORAWAN_APPKEY command ID

Example:

	Hex message	Content
Command	0x16	READ_LORAWAN_APPEUI command
Response	REPAPA-	0x60000720AP40U successful 50A-ERU_ADPE_APPEU.

150d / 0x96 - WRITE_LORAWAN_APPEUI

This command sets the APPEUI. The APPEUI must be 8 bytes long

Field	Great	Value	Description
WRITE_LORAWAN_APPEUI	Uint8_t	0x96	WRITE_LORAWAN_APPEUI command ID
AppEUI	8 x Uint8_t		

If the writing is successful, this is displayed by means of a response with NRF_SUCCESS.

Example:

	Hex message	Contents
Command	0x960001020304050607	WRITE_LORAWAN_APPEUI command
Answer	0x009600000000	WRITE_LORAWAN_APPEUI successful.

23d / 0x17 - READ_LORAWAN_APPKEY

Read APPKEY, 16 bytes

Field	Great	Value	Description
READ_LORAWAN_APPKEY	uint8_t	0x17	READ_LORAWAN_APPKEYcommand ID

Example:

	Hex Message	Contents
Command	0x17	READ_LORAWAN_APPKEY command
Answer	0x1700010203040506070809 0A0B0C0D0E0F	READ_LORAWAN_APPKEY successful.

151d / 0x97 - WRITE_LORAWAN_APPKEY

Write the APPKEY, 16 bytes

This command sets the APPKEY. The APPKEY must be 16 bytes long

Field	Major	Value	Description
WRITE_LORAWAN_APPKEY	Uint8_t	0x97	WRITE_LORAWAN_APPKEY command ID
Appkey	16 x Uint8_t	0 - 9	

If the writing is successful, this is displayed by means of a response with NRF_SUCCESS.

Example:

	Hex message	Contents
Comman d	0x970001020304050607080 90A0B0C0D 0E0F	WRITE_LORAWAN_APPKEY command
Answer	0x009700000000	WRITE_LORAWAN_APPKEY successful.

136d / 0x98 - WRITE_LORAWAN_TRANSMIT

Send a LoRaWAN message with the given payload on fport 5.

Field	Large	Value	Description
WRITE_LORAWAN_APP KEY	Uint8_t	0x97	WRITE_LORAWAN_APPKEY command ID
Length	Uint8_t	<28	Maximum size of 28 bytes
Payload	N x uload8 pointtespayo ut pointttes		Maximum,28 has a maximum size of 30 bytes.

If the writing is successful, this is represented by a response with NRF_SUCCESS.

Example:

	Hex message	Contents
Comman d	0x98100001020304050607080 90A0B0C0D 0E0F	WRITE_LORAWAN_APPKEY command with a length of 16 bytes.
Answer	0x009800000000	WRITE_LORAWAN_APPKEY successful. Is only controlled when the communication interface is BLE.

27d / 0x1B - READ_nRF_ADC_CONVERSION

Read the latest conversion values of the battery, nRF supply voltage and battery percentage.

Mission:

Field	Great	Value	Description
READ_nRF_ADC_CONVERS ION	uint8_t	0x1B	READ_nRF_ADC_CONVERSION command ID

Answer:

Field	Great	Value	Description

READ_nRF_ADC_CONVERSION	uint8_t	0x1B	READ_nRF_ADC_CONVERSION command ID
Battery voltage	Uint16_t		Battery voltage in millivolts
Power supply	Uint16_t		nRF52840 supply voltage in millivolts
Battery percentage	uint8_t	0-100%	Battery percentage, see battery voltage chapter for the calculation.

Example:

	Hex message	Contents
Assignment	1B	
Answer	1B0B530BB85C	Battery voltage 0x0B53 = 2899mV, nRF voltage = 3000mV, Battery percentage 92%

155d / 0x9B - WRITE_nRF_ADC_CONVERSION

Start an ADC conversion.

Mission:

Field	Great	Value	Description
WRITE_nRF_ADC_CONVERSION	uint8_t	0x9B	WRITE_nRF_ADC_CONVERSION command ID

Answer:

Field	Great	Value	Description
WRITE_nRF_ADC_CONVERSION	uint8_t	0x9B	WRITE_nRF_ADC_CONVERSION command ID
Battery voltage	Uint16_t		Battery voltage in millivolts
Supply	Uint16_t		nRF52840 supply voltage in millivolts
Battery percentage	uint8_t	0-100%	Battery percentage, see battery chapter for the calculation.

Example:

	Hex message	Contents
Command	0x9B	WRITE_nRF_ADC_CONVERSION, start conversion
Answer	0x9B0B530BB85C	Battery voltage 0x0B53 = 2899 mV, nRF voltage = 3000 mV, Battery percentage 92%

29d / 0x1D - READ_APPLICATION_CONFIG

Read the measurement interval in minutes and the ratio between the WAN and the RAN ratio. messages.

Assignment:

Field	Major	Value	Description
READ_APPLICATION_CONFIG	Uint8_t	0x1D	READ_APPLICATION_CONFIG assignment ID

Answer:

Field	Major	Value	Description
READ_APPLICATION_CONFIG	Uint8_t	0x1B	READ_APPLICATION_CONFIG assignment ID
Ratio measurements sent-	Uint8_W	-	The ratio between the measurements and the UR8_W measurements. If this number is three, one in three measurements is sent with LoRaWAN, provided the duty cycle does not limit this.
Interval	Uint16_t	> 0	The measurement interval in minutes.

Example:

	Hex message	Contents
Command	0x1D	READ_APPLICATION_CONFIG
Answer	0x1D03000A	Ratio of 1: 3 for measuring and sending. 1 in 3 measurements is sent with LoRaWAN. Measuring interval is 0x000A / 10d minutes.

	Hex message	Contents
Assignment	0x1D	READ_APPLICATION_CONFIG
Answer	0x1D000001	All measurements are sent. Measurement interval is 0x0001 / 1d minute.

157d / 0x9D - WRITE_APPLICATION_CONFIG

Set the measurement interval and the ratio between measuring and sending. Responses to downlink messages and the start-up message with the firmware and hardware version ignore this configuration and are always sent directly at the next message interval.

Assignment:

Field	Great	Value	Description

WRITE_APPLICATION_CONFIG	Uint8_t	0x9D	WRITE_APPLICATION_CONFIG command ID

Answer:

Field	Great	Value	Description
WRITE_APPLICATION_CONFIG	Uint8_t	0x9D	WRITE_APPLICATION_CONFIG command ID
Ratio measurements sent-	Uint8_W	-	the ratio between the measurements and the LoR number of measurements. If this number is three, one in three measurements is sent with LoRaWAN, provided the duty cycle does not limit this.
Interval	Uint16_t	> 0 && <1440	The measurement interval in minutes. Must be a minimum of 1 and a maximum of 1440. Otherwise error code NRF_ERROR_INVALID_PARAM

Example:

	Hex message	Contents
Command	0x9D03000A	Ratio of 1: 3 for measuring and sending. 1 in 3 measurements is sent with LoRaWAN. Measuring interval is 0x000A / 10d minutes.
Answer	0x009D00000000	NRF_SUCCES for WRITE_APPLICATION_CONFIG

Example with incorrect sample interval:

	Hex message	Content
Command	0x9D03FFFF	Ratio of 1: 3 for measuring and sending. 1 in 3 measurements is sent with LoRaWAN. The measuring interval is 0xFFFF / 65535d minutes, which is higher than the maximum interval of 1440 minutes.
Answer	0x009D00000007	NRF_ERROR_INVALID_PARAM

Testing:

	Hex Message	Content
Command	0x9D000001	Ratio of 1: 0 for measuring and sending. All measurements are sent with LoRaWAN. Measuring interval is 0x0001 / 65535d minutes, which is higher than the maximum interval of 1440 minutes.
Answer	0x009D00000000	NRF_SUCCES for WRITE_APPLICATION_CONFIG

30d / 0x1E - READ_PINCODEreads

Read the BLE pin code, 6 numbers: '0' - '9'

This commandthe BLE pin code. The answer is structured according to the WRITE_PINCODE command, but then with the READ_PINCODE command. The BLE standard specifies a pin code of 6 numbers. The length must therefore always be 6.

Field	Great	Value	Description
WRITE_PINCODE	Uint8_t	0x1E	WRITE_PINCODE command ID

Example:

	Hex message	Content
Command	0x1E	READ_PINCODE command
Answer	0x1E06303132333435	READ_PINCODE response, pin code 8 bytes: “012345”

158d / 0x9E - WRITE_PINCODE

pin code is set with this code. The pin code must contain between 6 ASCII numbers between '0' (0x30h) and '9' (0x39h).

Field	Great	Value	Description
WRITE_PINCODE	Uint8_t	0x9E	WRITE_PINCODE command ID
length	Uint8_t	6	Number of bytes of the pin code must be 6, otherwise errorcode: NRF_ERROR_INVALID_LENGHT
pin code	6 x Uint8_t	'0' - '9'	Byte values must be between 0x30, and error 039: NRF_ERROR_INVALID_DATA

If the writing is successful, this is represented by a response with NRF_SUCCESS.

Example:

	Hex message	Contents
Command	0x9E06303930393035	WRITE_PINCODE command: “090905”
Answer	0x009E00000000	WRITE_PINCODE successful.

Example of error message:

	Hex message	Contents
Command	0x9E06003930393035	WRITE_PINCODE command: “/ 090905”
Answer	0x009E0000000B	WRITE_PINCODE error: NRF_ERROR_INVALID_DATA.

31d / 0x1F - READ_BOOT_COUNT

With this command the number of resets can be retrieved from the flash memory.

Mission:

Field	Great	Value	Description
READ_BOOT_COUNT	uint8_t	0x1F	READ_BOOT_COUNT command ID

Answer:

Field	Great	Value	Description
READ_BOOT_COUNT	uint8_t	0x1F	READ_BOOT_COUNT command ID
length	uint32_t	-	Boat count

Example:

	Hex Message	Contents
command	0x1F	READ_BOOT_COUNT command
Answer	0x1F00000005	Reset counter reaches 5

32d / 0x20 - READ_MX_FLASH

With this command you can read the log in the FLASH memory of the MX25R6435F. This command also includes an offset from the start of the log. This makes it possible to read out part of the flash memory.

If the offset value is greater than the size of the log, the complete log is read from offset 0.

If the command is accepted, a RESPONSE is sent in the BEEP service with NRF_SUCCESS. The flash data is sent via the TX characteristic under the BEEP service. If the TX notifications are not enabled, the readout is ignored and an error code is returned.

Mission:

Field	Great	Value	Description
READ_MX_FLASH	uint8_t	0x20	READ_MX_FLASH command ID
Offset	uint32_t	-	Offset in bytes

Answer:

Field	Great	Value	Description

RESPON SE	uint8_t	0x00	RESPONSE command ID
command	uint8_t	0x20	READ_MX_FLASH task ID
Error code	uint32_t	-	See Table 2

Example:

	Hex Message	Contents
Comman d	0x2000000000	READ_MX_FLASH command, offset 0
Response	0x0020000000	RESPONSE: READ_MX_FLASH, error code = NRF_SUCCESS

33d / 0x21 - ERASE_MX_FLASH

With this command the BEEPlog in the flash memory of the MX25R6435F is deleted. However, there are two erase options: fatfs erase and full erase. With option 0, only the log is deleted from the file system. With the erase MX option the complete flash storage is written back to 1 at chip level.

If the fatfs erase command is accepted, a RESPONSE is sent in the BEEP service with a fatfs error code.

For the MX erase, only NRF_SUCCESS is sent back once the erase is ready. Data storage or reading out during deletion is not possible. The nRF checks for a timeout of 250 seconds. According to the datasheet of the MX, an erase lasts a maximum of 240 seconds. In the case of a timeout during an MX erase, NRF_ERROR_TIMEOUT is returned.

After deleting, a new start-up message is always written in the new log.

Assignment:

Field	Great	Value	Description
READ_MX_FLASH	Uint8_t	0x21	ERASE_MX_FLASH assignment ID
Erase option	uint8_t		0 = fatfs removed BEEPlog 1-0xFF = MX flash IC does a full erase that can last a maximum of 250s.

Answer:

Field	Great	Value	Description
RESPON SE	Uint8_t	0x00	RESPONSE command ID
command	Uint8_t	0x21	ERASE_MX_FLASH command ID
fatfs error Error code	Uint32_t	-	See table below 2

Value	Description fatfs error code
0	Succeeded
1	A hard error occurred in the low level disk I / O layer
2	Assertion failed
3	The physical drive cannot work
4	Could not find the file
5	Could not find the path
6	The path name format is invalid
7	Access denied due to prohibited access or directory
8	Access denied due to prohibited access
9	The file / directory object is invalid
10	The physical drive is write protected
11	The logical drive number is invalid
12	The volume has no work area
13	There is no valid FAT volume
14	The f_mkfs () aborted due to any problem
15	Could not get a grant to access the volume within defined period
16	The operation is rejected according to the file sharing policy
17	LFN working buffer could not be allocated
18	Number of open files> _FS_LOCK
19	Given parameter is raid id

Table 2 - fatfs error codes

Example 1:

	Hex message	Content
Command	0x2100	ERASE_MX_FLASH command, erase fatfs
Answer	0x0020000000	

Example 2:

	Hex message	Content
Command	0x2101	ERASE_MX_FLASH command, erase MX
Answer	0x0020000000 Sends	after max 240 seconds.

34d / 0x22 - SIZE_MX_FLASHretrieves

This command the size of the log in the flash memory of the MX25R6435F. If the command is accepted, a response is sent containing the size of the log in bytes.

Mission:

Field	Great	Value	Description
SIZE_MX_FLASH	uint8_t	0x22	SIZE_MX_FLASH command ID

Answer:

Field	Great	Value	Description
Command	uint8_t	0x22	SIZE_MX_FLASH response
Log great	uint32_t	-	Big log in bytes

Example:

	Hex Message	Contents
Comman d	0x22	SIZE_MX_FLASH command
Answer	0x220000FB40	Big log: 0x0000FB40 / 64320d bytes

35d / 0x23 - ALARM_CONFIG_READ

Read the alarm configuration for a specific sensor.

Mission:

Field	Great	Value	Description
ALARM_CONFIG_READ	uint8_t	0x23	ALARM_CONFIG_READ command ID
Sensor type	uint8_t	Sensor	0 = DS18B20 1 = BME280 2 = HX711 4 = NRF ADC

A.

Field	Great	Value	Description
ALARM_CONFIG_READ	uint8_t	0x23	ALARM_CONFIG_READ
sensor type	uint8_t	Sensor	0 = DS18B20 1 = BME280 2 = HX711 4 = nRF ADC
Sensor specific data			

DS18B20

If the measured temperature exceeds the maximum temperature, below the minimum temperature or the absolute difference with respect to the previous measured value exceeds the set limit values, an alarm is indicated. In the alarm status the DS18B20 bit is then set to 1.

Field	Large	Description

ALARM_CONFIG_READ	Uint8_t	0x23 = ALARM_CONFIG_READ
Sensor type	Uint8_t	0 = DS18B20
Max	Int16_t	INT16_MAX (32767) turns off this check, temperature is accurate to 2 decimal places int16_t
Min	Int16_t	INT16_MIN (-32768) sets this check to 2 decimals temperature is set to 2 decimals int16_t
Diff	uint16_t	UINT16_MAX (65535) puts this check, temperature in 2 decimal int16_t

Example:

	Hex message
Command	0x2300
Answer	0x23001F40F63C03E8

Field	Great	Value	Description
ALARM_CONFIG_READ	uint8_t	0x23	ALARM_CONFIG_READ
sensor type	uint8_t	0x00	DS18B20
Maximum	Int16_t	0x1F40	8000d = 80.00 C °
Minimum	Int16_t	0xF63C	-2500d = -25.00 C °
Difference	uint16_t	0x03E8	1000d = 10.00 C °

BME280

The BME280 measures temperature, humidity and barometric pressure, for each measurement result a maximum, minimum and difference value can be set.

Field	Large	Description
ALARM_CONFIG_READ	uint8_t	0x23 = ALARM_CONFIG_READ
sensor type	uint8_t	0x01 = BME280
maximum temperature	Int16_t	INT16_MAX (32767) converts the control of
temperature minimum	Int16_t	INT16_MIN (-32768) puts this check
temperature difference	Uint16_t	UINT16_MAX (65535) puts this check
Humidity maximum	Uint16_t	UINT16_MAX (65535) puts this check
Humidity minimum	Uint16_t	UINT16_MAX (65535) puts this check
Humidity difference	Uint16_t	UINT16_MAX (65535) puts this check
Barometric pressure maximum	Uint16_t	UINT16_MAX (65535) puts this check
Barometric pressure minimum	Uint16_t	UINT16_MAX (65535) puts this check
Barometric pressure difference	Uint16_t	UINT16_MAX (65535) puts this check

Example:

	Hexpost
Comman d	0x2301
Reply	0x23011F40F63C03E8232803E801F4271000C80 0C8

Field	Great	Value	Description
ALARM_CONFIG_READ	uint8_t	0x23	0x23 = ALARM_CONFIG_READ
sensor type	uint8_t	0x01	0x01 = BME280
Temperature maxi mum	Int16_t	0x1F40	8000 = 80.00
Temperature minimum	Int16_t	0xF63C	-2500 = 25.00
Temperature difference	Uint16_ t	0x03E8	1000 = 10.00
Humidity maximum	Uint16_ t	0x2328	9000 = 90.00% RH
Humidity minimum	Uint16_ t	0x03E8	1000 = 10.00% RH
Humidity difference	Uint16_ t	0x01F4	500 5.00% RH =
barometric pressure maximum	Uint16_ t	0x2710	10000 hPa
Barometric pressure minimum	Uint16_ t	0x00C8	200 hPa
Barometric pressure difference	Uint16_ t	0x00C8	200 hPa

HX711

The same limits are used for each channel. It is therefore not the intention to generate an alarm at a specific value. I can best solve that in the back-end. But more to get an indication if, for example, the load cell is no longer connected.

Field	Great	Value	Description
ALARM_CONFIG_RE AD	uint8_t	0x23	ALARM_CONFIG_READ
sensor type	uint8_t	2	HX711
Maximum	int32_t	-	INT32_MAX (0xFFFFFFFF) puts this check
Minium	int32_t	-	INT32_MIN (0x00000000) puts this check
Difference	uint32_t	-	A value above 0xFFFFFFFF put this check

Example:

	Hex message
Comman d	0x2302
Answer	0x230200003E80FFFFC180000003E8

Field	Great	Value	Description
ALARM_CONFIG_RE AD	uint8_t	0x23	ALARM_CONFIG_READ
sensor type	uint8_t	0x02	HX711
Maximum	int32_t	0x00003E80	16000
Minium	int32_t	0xFFFFC18 0	-16 000
Difference	uint32_t	0x000003e8	1000

NRF ADC

the limit values for the ADC in the nRF52840 be the battery voltage and the nRF52 supply checked.

Field	Great	Value	Description
ALARM_CONFIG_READ	uint8_t	0x23	ALARM_CONFIG_READ
sensor type	uint8_t	0x04	NRF ADC
Maximum	Uint16_t		UINT16_MAX (65535) puts this check
Minimum	Uint16_t		UINT16_MAX (65535) puts this check
Difference	Uint16_t		UINT16_MAX (65535) puts this check

Example:

	Hex message
Comman d	0x2304
answer	0x23040CE4070801F4

Field	Great	Value	Description
ALARM_CONFIG_READ	uint8_t	0x23	ALARM_CONFIG_READ
Sensor type	uint8_t	0x04	NRF ADC
Maximum	Uint16_t	0x0CE4	3300 mV
Minimum	Uint16_t	0x0708	1800 mV
differential	Uint16_t	0x01F4	500 mV

163d / 0xA3 - ALARM_CONFIG_WRITE

Set the alarm configuration for a specific sensor. The DS18B20, BME280, HX711 and nRF ADC can be set with the following message:

Command:

Field	Large	Value	Description
ALARM_CONFIG_WRITE	Uint8_t	0xA3	ALARM_CONFIG_WRITE command ID
Sensor type	uint8_t	Sensor	0 = DS18B20 1 = BME280 2 = HX711 X =RX= nRF
dependent datapayloadpayload	X	X	X

Answer:
If the message is accepted, a status message will be sent back with NRF_SUCCESS. If the message structure or a parameter has an invalid value, an appropriate status error code is sent back.

DS18B20

If the measured temperature exceeds the maximum temperature, below the minimum temperature or the absolute difference with respect to the previous measured value exceeds the set limit values, an alarm is indicated. In the alarm status the DS18B20 bit is then set to 1.

Field	Large	Description
ALARM_CONFIG_WRIT E	Uint8_t	0x23 = ALARM_CONFIG_WRITE
Sensor type	Uint8_t	0 = DS18B20
Max	Int16_t	INT16_MAX (32767) turns off this check, temperature is accurate in 2 decimals int16_t
Min	Int16_t	INT16_MIN (-3276im) sets this value for 2 int16_t
Diff	uint16_ t	UINT16_MAX (65535) puts out this check, temperature is in the second decimal place int16_t

Example:

Field	Large	Value	Description
ALARM_CONFIG_WRITE	uint8_t	0xA3	ALARM_CONFIG_WRITE

Sensor type	uint8_t	0x00	DS18B20
Maximum	int16_t	0x1F40	8000d = 80.00 ° C
Minimum	int16_t	0xF63C	-2500d = -25.00 C °
Difference	uint16_t	0x03E8	1000d = 10.00 C °

	Hex message
Comman d	0xA3001F40F63C03E8
Answer	0x00A300000000

BME280

The BME280 measures temperature, humidity and barometric pressure, for each measurement result a maximum, minimum and difference value can be set.

Field	Large	Description
ALARM_CONFIG_WRITE	Uint8_t	0xA3 = ALARM_CONFIG_WRITE
Sensor type	Uint8_t	0x01 = BME280
Temperature maximum	Int16_t	INT16_MAX (32767) turn this check off
Minimum temperature	Int16_t	INT16_MIN (-32768) turn off this check65) turn off this
Temperature difference	U5_t65 5	65 (UintMAt 65
Maximum humidity	Uint16_t	UINT16_MAX (65535) turns this check off
Humidity minimum	Uint16_t	0 turns this check off
Humidity difference	Uint16_t	UINT16_MAX (65535 or 0) turns off this check
Barometric pressure maximum	Uint16_t	UINT16_MAX (65535) turns this check off
pressure minimum	Bar16T urns161 6	0offthis check
Barometric pressure difference	Uint16_t	UINT16_MAX (65535 0) puts this check

Field	Great	Value	Description
ALARM_CONFIG_WRITE	uint8_t	0xA3	0xA3 = ALARM_CONFIG_WRITE
sensor type	uint8_t	0x01	0x01 = BME280
maximum temperature	Int16_t	0x1F40	8000 = 80.00
minimum temperature	Int16_t	0xF63C	-2500 = 25.00

Temperature difference	Uint16_t	0x03E8	1000 = 10.00
Humidity maximum	Uint16_t	0x2328	9000 = 90.00% RH
Humidity minimum	Uint16_t	0x03E8	1000 = 10.00% RH
Humidity difference	Uint16_t	0x01F4	500 = 5.00% RH
Barometric pressure maximum	Uint16_t	0x2710	10000 hPa
Barometric pressure minimum	Uint16_t	0x00C8	200 hPa
Barometric pressure difference	Uint16_t	0x00C8	200 hPa

Example:

	Hex message
Command	0xA3011F40F63C03E8232803E801F4271000C800C8
Answer	0x00A300000000

HX711

The same limits are used for each channel. It is therefore not the intention to generate an alarm at a specific value. I can best solve that in the back-end. But more to get an indication if, for example, the load cell is no longer connected.

The limits

Field	Great	Value	Description
ALARM_CONFIG_WRITE	uint8_t	0xA3	ALARM_CONFIG_WRITE
sensor type	uint8_t	0x02	HX711
Maximum	Int24_t	-	INT32_MAX (0x7FFFFFFF) puts this check
Minium	Int24_t	-	INT32_MIN (0x80000000) puts this check
Difference	Int24_t	-	A value above 0xFFFFFF put this check

example 1:

Field	Great	Value	Description
ALARM_CONFIG_WRITE	uint8_t	0xA3	ALARM_CONFIG_WRITE
sensor type	uint8_t	0x02	HX711
Maximum	int32_t	0x00003E80	16000

Minium	int32_t	0xFFFFC180	-16 000
Difference	uint32_t	0x000003e8	1000

	post Hex
Command	0xA30200003E80FFFFC180000003E8
Answer	0x00A300000000

example 2:

Field	Great	Value	Description
ALARM_CONFIG_WRITE	uint8_t	0xA3	ALARM_CONFIG_WRITE
sensor type	uint8_t	0x02	HX711
Maximum	Int32_t	0x7FFFFFFF	2147483647
Minium	Int32_t	0x80000000	-2147483648
Difference	Uint32_t	0xFFFFFFFF	1000

	Hex message
Command	0xA3027FFFFFFF80000000FFFFFFFF
Answer	0x00A300000000

nRF ADC

With the limit values for the AD5 the RF28 deRF nF28 deRF and the battery voltage and the nRF52 supply voltage checked.

Field	Great	Value	Description
ALARM_CONFIG_WRITE	uint8_t	0xA3	ALARM_CONFIG_WRITE
sensor type	uint8_t	0x04	NRF ADC
Maximum	Uint16_t		UINT16_MAX (65535) puts this check
Minimum	Uint16_t		0 disables this check
Difference	Uint16_t		UINT16_MAX (65535) puts this check

field	Large	Value	Description
ALARM_CONFIG_WRITE	uint8_t	0x23	ALARM_CONFIG_WRITE
Sensor type	Uint8_t	0x04	nRF ADC
Maximum	Uint16_t	0x0CE4	3300 mV

Minimum	Uint16_t	0x0708	1800 mV
Difference	Uint16_t	0x01F4	500 mV

Example:

	Hex message
Comman d	0xA3040CE4070801F4
Response	0x00A300000000

36d / 0x24 - ALARMADSTATUS

current alarm. An alarm is generated or turned off with a new measurement of the relevant sensor. The DS18B20 temperature, HX711, BME280 and nRF ADC have maximum, minimum and difference limit values where the measured value must fall in order not to generate an alarm. The difference limit is with respect to the previous measurement value, provided that the previous measurement value is valid and has the same number of measurement values or resolution.

Mission:

Field	Great	Value	Description
ALARM_STATUS_READ	uint8_t	0x24	ALARM_STATUS_READ command ID

Answer:

Field	Great	Value	Description
ALARM_STATUS_READ	uint8_t	0x24	ALARM_STATUS_READ
Alarms status bits	uint8_t		bit 0 = DS18B20 bit 1 = BME280 bit 2 = HX711 bit 3 = Audio ADC bit 4 = NRF ADC

Alarms are only generated on the DS18B20, BME280, HX711 and the nRF ADC. The index value is used to set a bit high or low in the alarm status byte that is returned.

Sensor	(Bit) Index
DS18B20	0
BME280	1
HX711	2

Audio ADC	3
NRF ADC	4
SQ_MIN	5
ATECC	6
BUZZER	7
LORAWAN	8
MX_FLASH	9
nRF_FLASH	10
Application	11

Example 1:

	Hex Message	Contents
Command	0x24	ALARM_STATUS_READ
Answer	0x2404	ALARM_STATUS_READ, HX711

Bluetooth Low Energy

The nRF52840 will support the following services:

1. DIS: Device information service.
2. BAS: Battery service.
3. BEEP unique service Read
4. log.
5. DFU: Nordic's firmware update service

The device will advertise itself as BEEPXXXX (Example). Where the last 4 characters are the 4 least important characters of DEVEUI in hexadecimal. For example, if DEVEUI is 0x01 23 45 67 89 AB CD EF, the BLE ad name is BEEPCDEF.

Pin code

The standard PIN code is “123456”. The pin code can be changed using the BEEP protocol. The Pin code of the BLE specification must always consist of 6 ASCII numbers ('0' - '9').

The PIN code can be manually reset by energizing the reed switch with a magnet for 30 seconds. The BEEP base will immediately emit two short tones when energizing the reed switch, indicating that the BEEP base has started BLE advertising. If the pin code is reset, the BEEP base indicates this with a long beep with the standard buzzer melody 1. Melody 1 is a long beep of 4 seconds.

Device information service

The device information service supports the following characteristics with the following values:

- Manufacturer Name String: “BEEP”
- Model Number String: “BEEP base”
- Serial number String: “TODO: ATTEC”
- Hardware Revision String: “1.0”
- Firmware Revision String : "0.0.1"

Battery service

The Battery service (BAS) indicates a rough estimated battery percentage. The percentage is only an indication, since batteries have a very strong temperature and power consumption dependence.

The battery percentage is calculated on the sum of 10 ADC measurements with which an average battery voltage in mV is calculated. Based on the nominal battery voltage and the cutt-off battery voltage, a battery percentage is linearly calculated.

$$Battery\ cutt-off\ voltage: \ mV_{cutt-off} = 1600\ mV$$
$$Battery\ nominal\ voltage: \ mV_{nom} = 3000 mV$$
$$average\ battery\ voltage: \ mV_{batt\_gem}$$
$$mV_{batt\_gem} = \frac{\sum ADC_{samples} \times 3600.0}{N_{samples} \times 4096.0}$$
$$Percentage = \frac{(mV_{battgem} - mV_{cutt-off}) * 100}{mV_{batt\_gem} - mV_{cutt-off}}$$

Example:

$$mV_{batt\_gem} = \frac{32993 \times 3600.0}{10 \times 4096.0} = 2899.77 mV$$

$$Percentage = \frac{(2899.77 mV - 1600\ mV) * 100}{3000 mV - 1600\ mV} = 92.84\%$$

The percentages are rounded to whole percentages, so the battery percentage then becomes 92.

DFU

Not yet implemented, this always requires a boot loader with SDK15.3.

BEEP service

The BEEP service has the following UUID:

UUID:	1bc3f8c5-ebc6-4050-ad4b-9f71d4a647be
Hex UUID	{0x1b, 0xc3, 0xf8, 0xc5, 0xeb, 0xc6, 0x40, 0x50, 0xad, 0xxx, 0xx4xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx 0xa6, 0x47, 0xBE}

the following short UUID are used to the characteristics of the service:

Characteristic	ShortUUID	LongUUID
BEEP service	0x68A1	BE4768A1-719F-4BAD-5040-C6EBC5F8C31B
DS18B20 measurement result	0x68A2	C6EBC5F8C31B
BE4768A2-719F-4BAD-5040-TXlog	0x68A3	BE4768A3-719F-4BAD-5040-C6EBC5F8C31B
BEEP Control Point	0x68B0	000068B0-0000-1000-8000-00805F9B34FB

Attention! NRF Connect displays the Long UUIDs in reverse order!

DS18B20 temperature result characteristic

The DS18B20 temperature result characteristic shows the last temperature measurement. Just like in the BEEP protocol, the first byte is the number of sensors, followed by an int16_t for each sensor with the temperature in one hundredth degree of accuracy. So in order to convert the temperature to degrees Celsius, the temperature number must be divided by one hundred.

Byte	Content
0	Number of DS18B20 sensors
(N * 2) + 0	MSB of int16_t temperature in hundredth degree accuracy
(N * 2) + 1	LSB of int16_t temperature in hundredth degree accuracy

For example:

01-08-98

Byte 0 is 1, so but 1 temperature sensors.
The temperature value is 0x0898 / 2200d, but must be interpreted as int16_t for signedness. By dividing the value of 2200 by one hundred, the temperature is calculated in degrees: 2200/100 = 22.00 ° C.

TX log data

The TX log characteristic is used to send the data from the flash log to the client as soon as the order is given with the BEEP Control Point.

If the READ_FLASH command is sent to the BEEP control point with a valid offset, then the data stream from the log starts. Of all messages received with the TX characteristic, the first two bytes are a frame counter in big endian, which always starts at zero. This allows the client to check whether any messages are missing. The rest of all data is log data. All log data bytes must be merged before the data can be interpreted according to the description in the "Flash log" chapter.

Example from nRF Connect:

```
I	09: 34: 06.734 Notification received from be4768a3-719f-4bad-5040-c6ebc5f8c31b, value: (0x)
00-00-02-25-30-31-30-30-30-31-30
-30-30-32-30-30-30-30-30-32-30-30-30-31-30-30-30-30-30-30-30-32-45-37-30-45
-30-45-30-31-32-33-33-44-32-33-30-38-45-43-38-45-39-31-45-45-31-46-30-30-30
-30-30-30-30-31-30-33-30-39-31-44-30-30-30-30-30-31-0A-03-11-31-42-30-41-41
-45-30-41-41-34-36-32-30-41-30-31-30-31-38-35-38-30-30-34-30-32-30-38-35-46
-30-38-35-39-0A-03-11-31-42-30-41-41-37-30-41-41-35-36-32-30-41-30-31-30-31
-38-35-43-34-30-34-30-32-30-38-35-33-30-38-35-46-0A-03-11-31-42-30-41-41-44
-30-41-41-38-36-32-30-41-30-31-30-31-38-34-45-46-30-34-30-32-30-38-35-33-30
-38-35-33-0A-03-11-31-42-30-41-41-44-30-41-41-34-36-32-30-41-30-31-30-31-38
-35-30-35-30-34-30-32-30-38-34-44-30-38-35-39-0A-03-11-31-42-30-41-41-46-30
-41-41-35-36-32-30-41-30
I	09: 34: 06.783 Notification received from be4768a3-719f-4bad-5040-c6ebc5f8c31b, value: (0x)
00-01-31-30-31- 38-34-31-42-30-34-30-32-30-38-35-33-3
0-38-35-39-0A-03-11-31-42-30-41-41-39-30-41-41-38-36-32-30-41-30-31-30-31-
38-34-35-37-30-34-30-32-30-38-35-46-30-38-35-46-0A
received
```

The orange colored bytes indicate the frame counters with the values 0 and 1.

BEEP Control point

The control point supports the BEEP protocol, but unlike LoRaWAN can only handle one command at a time. If no notifications are on, sent commands are executed, but the answer is never received by the sender.

LoRaWAN

The LoRaWAN stack will start initializing the hardware and retrieve the LoRaWAN keys as soon as the BEEP base is in a horizontal position. If LoRaWAN is disabled using the BEEP protocol or if one of the keys is disabled, LoRaWAN communication remains disabled.

The LoRaWAN keys are disabled if the entire DEVEUI, APPKEY or APPEUI is 0x00 or 0xFF. Even though LoRaWAN is switched on via the BEEP protocol, if one of the keys is incorrect, the LoRaWAN stack goes to an off position.

If new LoRaWAN keys are set via the BEEP protocol, the LoRaWAN stack must be reset so that the new keys are loaded. Until this is done, the LoRaWAN stack uses the old keys.

If LoRaWAN starts up because it has been reset via the BEEP protocol or if the BEEP base is placed in a horizontal position, it will attempt to log in to the back-end with the keys supplied. The LoRaWAN stack will then send Join Request messages. If there is a gateway within the range of the BEEP base and the mote is registered at the back-end, the BEEP base will receive a Join Accept message.

As soon as the BEEP base is registered with the back-end, it will first send a message with the firmware and hardware version and the unique ID of the ATECC. If the back-end misses this message, a downlink can always be used to find out what the firmware and hardware versions are.

If a downlink message is received, the payload is checked with the BEEP protocol. If there are valid commands, these are executed. Any answers are buffered by the LoRaWAN stack and sent with the first following uplink message.

Standard message types

The following message types are defined and are indicated by the Fport value.

Message type	Fport	Contains BEEP Protocol field
Sensor on	2	READ_FIRMWARE_VERSION, READ_FIRMWARE_VERSION
Keep alive	3	READ_nRF_ADC_CONVERSION, READ_HX711_CONVERSION, READ_DS18B20_CONVERSION
Alarm	4	The first byte in the payload displays the active alarm.
Uplink custom	5	Uplink payload is specified by the BEEP protocol command.
Downlink response	6	Contains the answer to a downlink BEEP command.

Table 3 - Message types

Alarm message

A LoRaWAN alarm message shows the alarms that have been active since the last LoRaWAN message. This means that if an alarm was generated for a certain measured value, but the next measured value has already reset the active alarm status, the alarm will still be sent.

The content of the alarm byte is as follows:

Sensor	Sensor bit	Comment
DS18B20	0	
BME280	1	
HX711	2	
Audio ADC	3	Cannot generate an alarm
nRF ADC	4	

The rest of the payload is according to the BEEP protocol as for a Keep-alive message.

Example:

LoRaWAN Message:	0x041B0B370B2D640A01018BBA040208BD08D00C0A00FF0117006D0045002A001E00190 01700150015000C07086C11D602A8
Content:	Alarm: 0x04, bits: HX711 Saadc: Vcc: 2871 mV, V bat: 2861 mV, Battery: 100% HX711: A128: 101306 / 0x018bba DS18B20 2 results: [0]: 0x08BD - 22,370 C , [1]: 0x08D0 - 22.560 C TLV FFT [10: 0: 255] 0.0 Hz = 279.401.7 Hz = 109.803.3 Hz = 69.1205.0 Hz = 42.1606.6 Hz = 30.2008.3 Hz = 25.2409.9 Hz = 23.2811.6 Hz = 21.3213.2 Hz = 21.3614.9 Hz = 12 BME read: Temp = 21.56 C, RH = 45.66%, Pressure = 680 hPa

Application

Buzzer sounds

The buzzer is used to inform the user of the state of the BEEP base. The buzzer can also be controlled via BLE or LoRaWAN messages. The BEEP base gives the following status indications:

State	Melody / indication
If the BEEP base is placed vertically and the BEEP base switches off.	A long beep
If the BEEP base is placed horizontally or starts up after the batteries have been installed.	Four beeps
If the reed switch is energized with a magnet	Two short beeps

Flash log

To store measurement data and other information, the MX25R6435 flash IC is used with the fatfs file system to save the data. At startup, a startup message is always written to the log with relevant data such as the bootcount and firmware and hardware version numbers. After each measurement based on the sampling interval, the measurement data is stored according to the BEEP protocol.

Data: ADC; Battery and supply voltage, HX711 (1 channel), DS18B20 (2 temperature sensors), FFT (10 results), BME280
Hex payload: 48 * 2 + 3 = 99 bytes at a time
Binary payload: 48 + 3 = 51 bytes at a time
Hex payload : 5 minutes interval = 0.8 Years
Hex payload 15 minutes interval = 2.4 Years
Binary payload 5 minutes = 1.56 Years
Binary payload 15 minutes = 4.69 Years

Message structure

To distinguish a start-up message and a measurement data message, each message starts with a byte specifying the message type, then a byte specifying the number of data bytes. The following is the data in binary format. Previous firmware versions <= 1.2.2 had the payload in ASCII characters, but this did not achieve the desired data storage. At the end of every message a new line feed character "\ n" follows. This makes the display of data in a text editor such as notepad ++ easier since each message is displayed on a separate line.

Field	Great	Value	Description
Message identification	Uint8_t	1 or 3	Message types are according to table 3. Only the Sensor on (1) and Keep alive message (3) types are used
Payload large	Uint8_t	> 0	
Payload	Uint8_t array [Payload large]		
Message end	Uint8_t	' \ n '= 0x0A / 10d	

Each message type has a hexadecimal data payload with the following parameters that are built according to the BEEP protocol:

Message type	Fport	Contains BEEP Protocol field
Sensor on	2	READ_FIRMWARE_VERSION = 1d, READ_FIRMWARE_VERSION = 2d, READ_ATECC_READ_ID_RAD_AD =, 14OTCOREAD_AD = 14OTCO 31, READ_DS18B20_STATE = 3, READ_APPLICATION_CONFIG = 29,
keep-alive	3	READ_nRF_ADC_CONVERSION = 27, READ_HX711_CONVERSION = 10, READ_DS18B20_CONVERSION = 4
Alarm	4	Not used
Uplink custom	5	Not used
Downlink response	6	Not used

Table 4 - message content

Start-up message

Example of a start-up post:

0x022501000100030000020001000000002e70e0e01233d2308ec8e91ee1f0000000b03091d0000010a

message structure:

Field	Content
Berichtty pe	0x02 /2d
Payload length	0x25/ 37d bytes of data,
Payload [37]	01000100030000020001000000002e70e0e01233d2308ec8e91ee1f0000000b03091d00000 1

endMessage	0x0A/ 10d '\ n'

parametersPayload

Hex	ID	Parameters
01000100030000	READ_FIRMWARE_VERSION 1d =	Firmware version: 1.3.0
02000100000002e7	READ_FIRMWARE_VERSION = 2d	Hardware version: ID 1.0 : 190222
0e0e01233d2308ec8e91ee	READ_ATECC_READ_ID = 14d	ATECC ID: 01233D2308EC8E91EE
1f0000000b	READ_BOOT_COUNT = 31	Boot count: 11
0309	READ_DS18B20_STATE = 3	DS18B20 state: 9
1D000001	READ_APPLICATION_CONFIG = 29	App Config: ratio: 0, interval 1 min

Measurement data message

Example of a measurement message:

0x03301b0a410a33590a0101893c040208e308e30c0a00ff00710014000f000d000d000a000a00090007000707086c11d602a80a

message Structure :

Field	Content
message type	0x03 /3d
Payload length	0x30/ 48ddata bytes
Payload[48]	1b0a410a33590a0101893c040208e308e30c0a00ff00710014000f000d000d000a000a00090007000707086c11d602a8
endmessage	0x0A/ 10d '\ n'

parametersPayload

Hex	ID	parameters
0x 1B0AAE0AA462	READ_nRF_ADC_CONVERSION 0x1B / 27d	Vcc: 2734 mV, V bat: 2724 mV, Battery: 98%
0x0A01018580	READ_HX711_CONVERSION 0x0A / 10d	HX711: A128: 99712 / 0x018580
0x0402085f0859	READ_DS18B20_CONVERSION 0x04 / 4d	DS18B20 2 results: [0]: 0x085F - 21430 C, [1]: 0x0859 - 21370 C
0x0c0a00ff00710014000f000d000	READ_AUDIO_ADC_CONVERSION 0x0C / 12d	TLV FFT [10: 0: 255] 0.0 Hz = 113,

d000a000a0009 00070007		203.2 Hz = 20, 406.4 Hz = 15, 609.6 Hz = 13, 812.8 Hz = 13, 1016.0 Hz = 10 , 1219.2 Hz = 10, 1422.4 Hz = 9, 1625.7 Hz = 7, 1828.9 Hz = 7,
0x07086c11d602 a8	BME280_CONVERSION_READ 0x07 / 7d	Temp = 21.56 C, RH = 45.66%, Pressure = 680 hPa

Program

BEEP base

The BEEP base must be assembled after assembly programmed. From Ideetron, zip and hex files are supplied per release. The Hex file is for programming with a programmer, for example an ARM flasher or an nRF528xx development board. The zip files are for firmware update via BLE, for example the nRF Connect desktop or telephone App.
To generate the zip and hex files, a batch file is used that compiles and merges the boot loader, application and boot loader settings under “Release” configuration.

For the zip files, encryption by means of nrfutil.exe is also used, so that only firmware that is compiled with the same keys as the boot loader is accepted by the boot loader.

Two zip files are always supplied: only the application and boot loader, soft device and application in one. The latter is very useful during development, so that all firmware is always compatible and not that an old boot loader cannot load a newer application. For this, the boot loader and application version number check is disabled, because otherwise it does not accept that boot loader or application firmware with the same version number is overwritten. If this is not disabled, the bootloader will refuse the complete firmware update.

For the final product release, it is up to the customer whether the version number control must be switched on again. It adds some extra requirements and control to firmware releases and the firmware can no longer be downloaded to a previous version.

Program script

The following batch file is used to program the BEEP base with the supplied hex file:

```
@ECHO OFF
SET hw_major = 1
SET hw_minor = 0
SET hw_ID = 190222
SET / A hw_reg_val =% hw_major% * 65536 +% hw_minor%
SET jlink_id = 682613435

ECHO Start programming HW% hw_major%.% Hw_minor%; reg:% hw_reg_val%
start / B / wait nrfjprog --snr% jlink_id% --eraseall
start / B / wait nrfjprog --snr% jlink_id% --memwr 0x10001080 --val% hw_reg_val%
start / B / wait nrfjprog - snr% jlink_id% --memwr 0x10001084 --val% hw_ID%
start / B / wait nrfjprog --snr% jlink_id% --program Release / BEEP base.hex
start / B / wait nrfjprog --snr% jlink_id% --reset
ECHO Programming Done

GOTO End
: End
pause
```

This batch script does not turn on the readback protection!

With the SET hw_major, hw_minor and hw_ID, the hardware version and ID numbers are programmed in the UUICR. These values are used in the firmware to display the hardware version in the DIS service and can be read out via the BEEP protocol.

The batch file above has a fixed Jlink set with ID 682613435. This will have to be adjusted for the programmer that is used. The ID can also be omitted, then the driver displays a pop-up with the available interface with each batch command if there are multiple programmers.

nRFutil

To create the zip files, nRFutil.exe is used, a program from Nordic that can encrypt hex files for DFU. Make sure that it is in the Util folder, since this executable is not in the repository system because of the size of the application. > 10Mb.
NRFUtil 5.2.0 was used for development. Older versions can cause problems that do not display a clear error.

Segger Embedded Studio

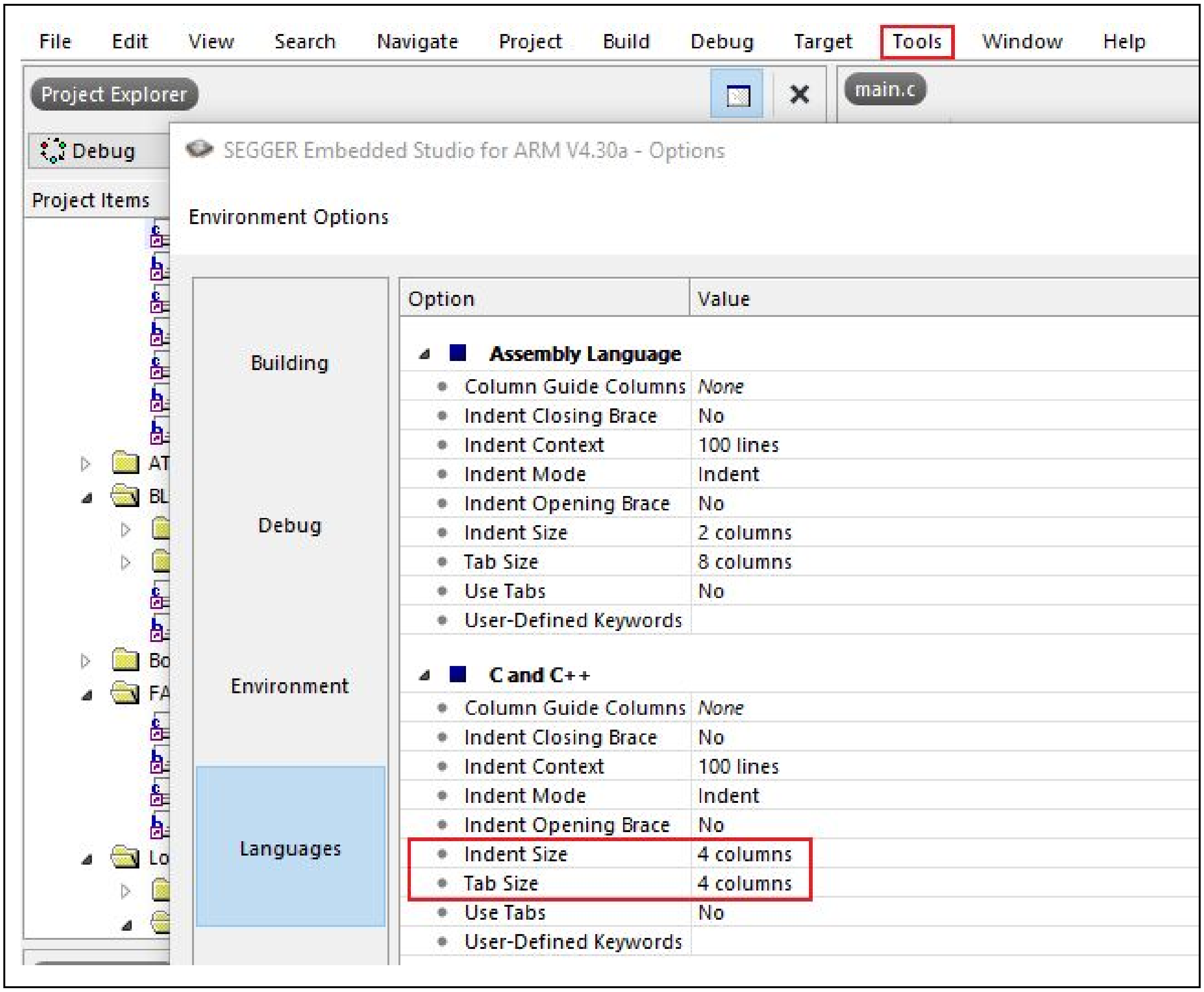
For the development of the firmware, use was made of Segger Embedded Studio, or SES abbreviated in the Nordic SDK. To use this program, a free license is required that is linked to a hardware ID of the PC.

A youtube video from Nordic Semiconductor how the license is requested through a browser can be found with the following link:
https://www.youtube.com/watch?v=fRAG6yOqt_4

Code editor adjustments

By default, the editor has an incorrect display of spaces. and tabs. As a result, code that appears to be neatly aligned in SES may have a chaotic alignment when the code is read with a different text editor.

To remedy this, the Tab size ne Indent size must be set to other values in SES. The options screen can be found under the menu "Tools"> "Options" which shows the options pop-up menu. Click on the "Languages" tab. Under C and C ++ change the Indent size to 4 columns and the tab size also to 4 columns. Click on Ok to save the settings.



Debugging

the application To debug the application with the DFU service, a boot loader is required that does not perform the CRC check and is loaded during debugging. To compile a boot loader a batch file has been created called "Compile_Bootloader_SkipCRC" that compiles the boot loader under a release that ignores the CRC check. This script can be found in the folder: \ Code \ nRF \ Util \ Program.

If the bootloader firmware has been modified or the source code has been downloaded, this script must first be executed once to create the hex file. SES will give a warning when loading all files before debugging if the hex file is missing.

If there is already a boot loader in the nRF52840, it must first be deleted to program the new boot loader without crc check.

Compilation scripts

The following batch script exists in the source code in the util / Program folder:

Batch file name	Function
Compile_BEEP_release	Compiles release versions of the application and bootloader. They are then merged with the soft device to create the hex file. From the different batch hex files created, the zip file is created for updating the firmware over BLE. Outputs: BEEP base.hex: Hex file with bootloader, application and soft-device that is programmed during production. To program these, a programmer is required such as an nRF52 development kit or a SEGGER programmer. BEEP base_app: Zip file for firmware update via Bluetooth Low Energy (DFU). Contains only the application BEEP base_sd_boot_app: Zip file for firmware update via Bluetooth Low Energy (DFU). Contains the application, soft device and the boot loader. Updating is performed in two steps. First the boot loader and soft device are updated and in the second DFU action the new boot loader is used to program the application.
Compile_Bootloader_SkipCRC	Compiles a boot loader that does not perform a CRC check. Required for debugging an application in SES.
EnableRBP	Enables the readback protection. After this, the BEEP base must be recovered, clearing the entire FLASH memory of the nRF52840.
erase	Deletes a microcontroller attached to a programmer.
erase_682613435	Deletes a microcontroller attached to a programmer with id 682613435.
FICR_read	Batch file with which some hardware parameters can be read from an nRF52840 chip, such as silicon version.
program_BEEPFirmwareprograms	Batch file that a connected BEEP base with the compiled hex file in the release folder.

Electrical diagram BEEP base PCB