PB868LRH and PB868LRI Push button sensors programming manual

1 Sommario

Document revision

Rev.	FW rev	Author	Note / remarks	Date
1.0	v0.7	R&D laboratory GM	Document created. Preliminary version (DRAFT)	Nov. 10, 2015
1.1	v0.7	R&D laboratory GM	Added new feature on Hardware re. B	March 31, 2016
1.2	v0.7	R&D laboratory GM	Added ACK/NACK messages on port 10 from sensor to server; New structure of message from server to sensor: 1 byte header and 1 byte footer introduced; TMELPS [24bit] increased to 16777215 sec; Added MODE1,2,3 feature Added CNFGRGST definition; Added TMALIVE	April 23, 2016
1.3	v0.7	R&D laboratory GM	Added FLAGS features: LED always OFF, Stop Blinking LED	May 9, 2016
1.4	v0.8	R&D laboratory GM	Added information on description chapter Text correction on port 6 and 7 on chapter 4 Added OPCNT value on ALIVE parameter definition Added CNFGRGST information on its definition on ch.3 Added more info to parameters in ch.3 Added LED blinking description message on port 40 on ch.4 Added payload information on port 12 specification – ch 5 Added graphical information on message from server to sensor on ch. 5	May 25, 2016
1.5	V0.8	R&D laboratory GM	Changed minimum battery level at 25%	June 27, 2016
1.6	V0.9	R&D laboratory GM	FW v0.9 supports OTAA Added OPCNT parameter into ALIVE Amended NACK message length definition and added description of ACK and NACK messages. Amended TMELPS definition Amended MODE 3 message description of 10 Byte	Aug 30, 2016
2.0	V1.7.0	R&D laboratory FC	FW v1.7.0 has an improved link check security algorithm configurable both from shell or server.	June 18, 2018
2.1	V1.7.0	R&D laboratory FC	Led blink behavior for PB868LRH	June 12, 2018

2 Introduction

This is the Push Button Sensor usage instructions for models PB868LRx and PBUS915LRx.
PB868LR and PBUS915LR are push buttons encapsulated on industrial casing while PB868LRH and PBUS915LR are push buttons encapsulated on casing for home and building applications.
You can install the sensor wherever you need to signal an event by pushing the button on the sensor.
The sensor transmits signals to the LoraWAN network when the button is pushed, or the number pushes programmed is reached. The sensor can also transmit after a certain programmable period is elapsed. Four diferent operational modes are available and programmable from remote.
PB868LRx (868MHz) and PBUS915LRx (915MHz) are push buttons sensors for European and US frequency bands.
PB868LRH and PBUS915LRH are equipped with a LED that blinks for 100ms every 15s if MODE 2 is set and when the number of pushed button counter equals the threshold CNTELPS. This gives a visual indication when a certain number of times of pushed button set is reached.

3 Installation

The PB868LRH can be fastened using screws or any type of adhesive that can hold the sensor’s weight. To place the screws the chassis must be pierced in the four corners where an oval shape can be seen. If the holes are placed in the right places the IP protection grade of the sensor (IP55) is not compromised.
Avoid mounting sensors in areas where they will be exposed to moisture or where the sensor operating temperature range of -4 to 120°F (-20 to 55°C) will be exceeded.

3.1 What to do and what don't to do

The PB868LRx can be used everywhere there is the need to generate an alarm.

3.2 Battery replacement

The sensor uses a 3.6V 2700mAh size AA Primary Lithium thionyl chloride battery (LiSoCl2) Spiral type as power supply.
To replace it, please pay particular attention to the type of battery.

CAUTION
RISK OF EXPLOSION IF BATTERY IS REPLACED BY AN INCORRECT TYPE.
DISPOSE OF USED BATTERIES ACCORDING TO THE INSTRUCTIONS

Only authorized and qualified personnel may do any of the assembly, disassembly, installation and commissioning work.

When the system indicates the sensor battery is low, replace it immediately.
To replace the batteries, do the following:

1. To remove the sensor cover, press a small flathead screw-driver into the slot on both sides of the sensor. This will disengage the clips holding the cover and base.
2. Disconnect the battery cable from the board. Remove the old battery and replace it with another one as per battery specification reported in chapter 2.
3. Insert the replacement battery and plug the cable in to connector (see below picture)

4 How the sensor works

The PB868LRx family is developed for indoor use only and can be operating in several modes for meet several scenario requests.
The PB868LRH version of this sensor is designed for home usage.
The PB868LRI version of this sensor is designed for industrial purposes.
At the start-up two messages are transmitted spontaneously to inform the server that the sensor has powered up:

- The first message is a string that represents the sensor model
  (see chapter 6.2.2 Message on port 5. *Presentation* pag.21)

- The second message is a string that report the version of the firmware, library and hardware
  (see chapter 6.2.4 Message on port 7. *FW release, library release, HW release* pag.21)

4.1 Confirmed and unconfirmed messages

The LoRaWAN protocol permits to send to the network server messages of confirmed or unconfirmed type.
The difference is that unconfirmed messages are transmitted one time only, and the sensor do not wait for any type of response from the server. The unconfirmed messages are preferred for information that does not have particular critical importance because is not guaranteed that messages are delivered to the destination.
Example of unconfirmed messages are ALIVE, LOW BATTERY and response from server Enquiry.

Differently, the confirmed messages are transmitted several times (from 1 to 8 programmable times) and the sensor wait to receive an acknowledge by the server.
The confirmed type messages are preferred for alarm event where the chance that message is not delivered is not permitted.

The programming of which messages are confirmed or not, is based on port.
Only by the shell in local mode is possible to set which ports are confirmed type, with the appropriate command.
Is possible to set up to 10 ports that are treated as confirmed type.

4.2 Alive

To get a periodic signal from the sensor, the ALIVE concept can be used.
The ALIVE is a periodic message transmitted to the server to inform it that the sensor is operating correctly, this packet is transmitted on port 9 (see chapter 6.2.6 Message on port 9. *Alive* pag.22)

4.3 MODE of operation

The scenario where the sensor operates is called MODE and in this firmware revision four MODE are available.

4.3.1 MODE 1

This is the basic function mode where PB868LRx sensor act as a typical push detector.
When an event is detected, the sensor sends an immediate uplink message to the server reporting the new state. For event is intended the button press or the button release.
(see the chapter 4.4.1 Event flag pag.12)

4.3.2 MODE 2

With this mode, the PB868LRx can work as a counter.
PB868LRx sensor will send messages only when a specific number of pushes is reached, this counter is called ***Detection Counter Number.***
You can set from the server or via serial shell a specific ***Detection Counter Number,*** that can be between 1 and 65535.
PB868LRx sensors will deliver an uplink message to server only when the programmed ***Detection Counter Number*** is reached.
If desired, it is possible to reset the ***Detection Counter Number*** every time the transmission is performed.
If the ***Detection Counter Number*** is not reset, a total amount of detection counter is reported to the server.

Example:
Detection Counter Number set to 15.

PB868LRx will deliver an uplink message to the server only when the detection of pushes reaches 15 counts.
It is totally independent by time: based on application and installation you can get this message at a frequent rate or also at several days distance.

4.3.3 MODE 3

With this mode, the sensor sends messages at regular intervals called ***Periodic Interval Time*** as programmed from the server or via serial shell.
The ***Periodic Interval Time*** is expressed in seconds and must be between 15 and 864000 seconds (10 days).
The PB868LRx will deliver to the server an uplink message with the ***Detection Counter Number*** collected from the sensor, when the specific interval time is elapsed.
If desired, is possible to reset the ***Detection Counter Number*** every time the transmission is performed.
If the ***Detection Counter Number*** is not reset, a total amount of detection counter is reported to the server.

Another available feature is the possibility of transmitting a message when a programmed ***Detection Counter Number*** is reached, before the ***Periodic Interval Time*** is elapsed.
To do this, you can set the appropriate flag as shown in the chapter 4.4.4 pag. 15

Example:
Periodic Interval Time set to 10 minutes
Detection Counter Number set to 15

Th Sensor PB868LRx will deliver a message to the server every 10 minutes reporting the ***Detection Counter Number*** happened in this time lapse.

The sensor will deliver to the server a message reporting that the ***Detection Counter Number*** is 15 if this count is reach before the ***Periodic Interval Time*** of 10 minutes is elapsed, and if enable flag is set.
At every detection, a fast blinking of led can be observed.

4.3.4 MODE 4

This mode was developed for working as an occupancy sensor.
You need to set a temporal interval called ***guard time***, that must be between 30 and 7200 seconds (2 hours).
The ***guard time*** is the period that the sensor waits when not detecting pushes before indicating that the button is no more used. When the ***guard time*** is lapsed without detecting pushes the sensor will communicate that the button has not been pressed for the ***guard time*.**
*Note that if the **guard time** is abnormally high, when the sensor is no more pushed, the button will report to the server the "room empty" message when **guard time** the is elapsed.*

As soon as a push is detected, PB868LRx will send a message to the server for indicate that someone is present.
In that moment, PB868LRx will activate the ***guard time*** and will start the countdown.
If during the ***guard time***, the sensor will detect a new push, the ***guard time*** will re-start the countdown from the beginning. If during the ***guard time*** PB868LRx will not detect any pression, the sensor will deliver a new message to server for indicate that the room is now empty.
At every detection, a fast blinking of led can be observed.

Example:
guard time set to 900 (15 minutes)

At the first detection, the sensor will deliver a message to the server to indicate that someone is present setting the bit 0 in the event flag (see the chapter 4.4.1 Event flag pag.12)
At the same time, ***guard time*** will be activated, and it will start the countdown.
If in these 15 minutes of ***guard time***, the sensor PB868LRx will detect new pushes, ***guard time*** will restart the countdown.
As soon as for 15 minutes (***guard time*** set) sensor PB868LRx will not detect any button press, a new message will be sent to server to indicate the fact that now the guard time is elapsed, clearing the bit 0 in the event flag (see the chapter 4.4.1 Event flag pag.12)

If you keep pushing the button for 2 hours, PB868LRx will send 2 messages: first one at time 0 when the first person pushed the button, second one after 2 hours and 15 minutes, so after 15 minutes after the last button press.

In this mode, you can also set a parameter called ***repetition time*** that ask to the PB868LRx sensor to send message at a specific interval, set by the user, where it will tell you that the button has been pressed. This time can be set from minimum 20 seconds up to maximum value minor then ***guard time***.
You can choose if you want to have this repetition, function set or not. (see chapter 4.4.4 pag. 15)

Looking at above example, if you activate also ***repetition time*** at 10 minutes, in the 2 hours of button pressions you will get:
First message as soon as the first person pushes the button.
A message every 10 minutes that tell you that the button has been pressed
A last message after 2hrs and 15 minutes that tell you that the sensor has not been used for 15 minutes.

Be aware that repetition time send an uplink message to the server and if it is set with short time it can reduce battery life time and can cause the risk to not respect the 1% duty cycle of LoRaWAN protocol.

4.4 Parameters specifications

4.4.1 Event flag status

This byte is a bit flag mask that reports the status of the sensor and is transmitted in several ports and in various scenario.

Event Flag byte
1 Byte unsigned char.

	MSB 7	*6*	*5*	*4*	*3*	*2*	*1*	*0 LSB*
Bit value	**0**	**0**	**0**	**x**	**x**	**x**	**x**	**x**

[7:3]	reserved
[2]	**Battery status** 1 = low battery event 0 = battery OK
[1]	Reserved
[0]	**Button** 1 = Button Pushed 0 = Button not pushed

If Low battery, tamper or reed event occurred the sensor will transmit a message on port 40 (see ch 4) reporting the status.
More events are possible at the same time.

4.4.2 Button press Counter

This unsigned word (16 bits) report the number of times the button has been pushed.

OPCNT

	MSB 15	*14*	*13*	*12*	*11*	*10*	*9*	*8 LSB*
Bit value	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	MSB 7	*6*	*5*	*4*	*3*	*2*	*1*	*0 LSB*
Bit value	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

[15:0]	Number of pressions

16 bits unsigned counter.
Number of button pressions.
This counter can be reset at the end of the transmission using CNFGRGST.

CNTELPS (Counter Lapse)
16 bits unsigned counter.
Specifies the number of times the push button has to be pushed before send an uplink event.

TMELPS (Time Lapse)
24 bits unsigned seconds counter ONLY for MODE 3.
It specifies the sampling period on MODE 3.
Once TMELPS is elapsed, the sensor sends a message on port 40 to the server and then it is reset to 0, ready for a new counter period.
Programmed values below 15 secs will be forced to 15. Values above n*15 and below (n+1)*15 will be forced to (n+1)*15 [sec] so the minimum setting timing period is 15 seconds with multiple of 15 seconds.
Allowed TMELPS value is comprised from 15 to 16777215 secs (about 194 days)
Example:

- The server sends TMELPS at 10 seconds then the sensor will force this at 15 seconds
- The server sends TMELPS at 31 seconds then the sensor will force this at 45 seconds

4.4.3 Device configuration flag

These flags configure the device for operating in a certain mode.

DEVICE CONFIGURATION FLAGS

	MSB 15	*14*	*13*	*12*	*11*	*10*	*9*	*8 LSB*
Bit value	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**x**

	MSB 7	*6*	*5*	*4*	*3*	*2*	*1*	*0 LSB*
Bit value	**0**	**0**	**0**	**0**	**0**	**0**	**x**	**x**

[15:9]	Unused - To be defined
[8]	Setting LED always OFF 1: always OFF. The LED never will be ON 0: Led will be ON when PB868LRH detects a button press or when the radio transmits a message **(This is valid only for the PB868LRH version, not for the industrial)**
[7:2]	Unused - To be defined
[1]	Setting Stop Blinking LED 1: stop blinking
[0]	Reboot firmware 1 = reboot request

4.4.4 MODE configuration flag

All the MODE, have associated some flags that configure the behaviour of the function.

MODE configuration Flag

	MSB 15	14	13	12	11	10	9	8 LSB
Bit value	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	MSB 7	6	5	4	3	2	1	0 LSB
Bit value	**0**	**0**	**0**	**0**	**x**	**x**	**x**	**x**

[15:4]	Unused To be defined
[3]	Transmit 'room busy' periodically at the ***repetition time*** interval (only for MODE 4) 1 = send a 'Room occupied' message periodically 0 = DO NOT sent (defaut value)
[2]	Only for mode 5: transmits if the reed contact is closedafter ***TMELPS Counter*** is elapsed (only for MODE 5). 1 = don't send the message if ***TMELPS*** is elapsed and the reed contact is closed. 0 = send the "contact closed" message if ***TMELPS*** is elapsed and the reed contact is reclosed (default value)
[1]	Transmit if ***Detection Counter Number*** if reached before ***Periodic Interval Time*** is elapsed (only for MODE 3) 1 = send ***Detection Counter Number*** is reached 0 = send only if ***Periodic Interval Time*** is elapsed (default value)
[0]	***Detection Counter Number*** reset (MODE 2 and MODE 3) 1 = reset request 0 = Not reset (default value)

ACK/NACK
3 bytes Char <ACK> and 4 bytes Char <NACK>)
Acknowledgement signal sent by the sensor to server. It can be Ack or Nack depending whether or not the message received from the server is fine. It also specifies on which protocol port the message has been received from the server. This message is sent every time the sensor receives a setting message from the server on ports 9, 12, 13, 30 and 31.

TMALIVE
16 bit unsigned seconds counter.
It specifies the number of seconds between two ALIVE events.
TMALIVE counter is set to 0 at the end of every transmission of the ALIVE event.
Programmed values below 15 secs, will be forced to 15
Values above n*15 and below (n+1)*15 will be forced to (n+1)*15 [sec] so the minimum setting timing period is 15 seconds with multiple of 15 seconds.
Allowed TMALIVE value is comprised from 15 to 65536 seconds
Default TMALIVE value is 50 minutes (3000 sec)
E.g.:

- The server sets TMALIVE at 9 seconds then the sensor will force this at 15 seconds
- The server sets TMALIVE at 40 seconds then the sensor will force this at 45 seconds

CNFGRGST
16 bit total.
This parameter is used in order to reset the OPCNT counter and to send the OPCNT value if the CNTELPS threshold is reached (only MODE 3)

	MSB 15	*14*	*13*	*12*	*11*	*10*	*9*	*8 LSB*
Bit value	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	MSB 7	*6*	*5*	*4*	*3*	*2*	*1*	*0 LSB*
Bit value	**0**	**0**	**0**	**0**	**0**	**0**	**x**	**x**

[15:2]	Unused To be defined
[1]	OPCNT value (only for MODE 3) 1 = send OPCNT value if OPCNT=CNTELPS 0 = OPCNT value is NOT sent (defaut value)
[0]	OPCNT counter reset (MODE 2 and MODE 3) 1 = reset request 0 = OPCNT is not reset (default value)

4.5 Link Check security mechanism

In this kind of sensors is present a security mechanism based on link check. With this algorithm the sensor can be aware if is connected to a network or not. The time between link checks can be provided by the user (please see Chapter 6.4.8 for the downlink specification). Once the programmed time elapsed the sensor's next message will have a link check request. If the request is answered, then the sensor is aware that is connected to a network. If the request is not answered, then the sensor will ask for the next link check in half the programmed time. This for 6 times. If the sensor is never answered during this six attempts the sensor will reboot.

This mechanism has been implemented for security purposes, but can be disabled at any time via shell or via server message.

5 Technical specification

Specifications				
Parameters	**Min**	**Typ**	**Max**	**unit**
Frequency band PB868LRx	-	867.1 < f < 868.5	-	MHz.
PBUS915LRx	-	902 < f < 928	-	MHz
RF power (EU868)	2	-	14	dBm EIRP
RF power (EN915)	2		18	dBm EIRP
Modulation		LoRa ™		
Protocol	LoRaWan 1.0 Class A Client 3.4.1			
RX sensivity			-138	dBm
Battery	1pz AA 3.6V lithium-thionyl 2700mA By EVE P/N ER14505V			
Temperature range	-20°	+25°C	+55°C	degree
Antenna	-	PCB printed	-	
Power supply	2.1	3	3.6	Vdc
Consumption standby	13	15	17	uA
Consumption TX	60	70	80	mA
Dimension	97 x 38 x 25			mm
Reference standards	EN 60950-1:2006 + A11:2009 + A1:2010 + A12:2011 – EN 62311:2008 EN 301 489-1 V1.9.2 EN 301 489-3 V1.6.1 ; Part 3 EN 300 220 V2.4.1			

Table 1 Technical Specification

6 Payload

6.1 Overview

All the messages exchanged between sensors and server are reduced to the strictly necessary for transport of the information requested. No other types of protocols are used for transporting the information on top the LoRaWAN protocol; this is all is needed to ensure the correct relay of information.
All messages are transmitted in RAW format.
The maximum length of any message in uplink direction is limited to 11 bytes. This precaution is necessary to don't trespass the maximum length of 11 bytes in US915 band at DR0.
The message from sensor node to the server (uplink direction) don't expected any type of control how message length, checksum or any other type of error correction.
The messages from server to the sensor node (downlink direction) are completed with length and checksum control. This precaution is necessary to avoid that wrong messages transmitted from the server may put the sensors in unpredictable state.
The messages received from the sensors, are confirmed to the server by sending ACK to inform him of the correct receipt, or NACK in case of corrupted message. If no message is received from the sensor, this one does not respond anything.

The information that the sensor is able to transmit, are organized on several port numbers.
This mechanism is used both for uplink and downlink messages.

Uplink message list:

From sensor to server (uplink)	Port #	Example	Payload raw (example)
Last sequence downlink	2	15	0000000F
Presentation	5	PB868LRH	50423836384C5248
Serial Number	6	Serial number of device	AA112233445566FF
FW release, library release, HW release	7	Firmware release 1.11.2639 LoRaWan stack Release 4.3.15 Hardware release B	010B0A4F04030F42
Battery level	8	Level in percentage, 94%	5E
Alive	9	Level in percentage, 94% Event flag	5E02
ACK	10	Ack on port 9	41636b09
Specific Sensor Information message (button press)	40	Event (1 byte), push counter (2 bytes)	010001

Downlink message list:

From server to sensor (Downlink)	Port	From server	Example Payload raw	Response from sensor
Request last downlink counter sequence number	2	Enq	05456E715F	The last SEQDN received
Request sensor model	5	Enq	05456E715F	PB868LRH
Request Serial Number	6	Enq	05456E715F	Serial number
Request FW, lib & HW release	7	Enq	05456E715F	Fw, lib & HW revision
Request Battery level	8	Enq	05456E715F	Battery level in percentage
Alive interval setting	9	Set the alive interval to 2 hours (7200 seconds)	041C2038	Ack or Nack
Mode settings	12	Set the scenario		Ack or Nack
Flag parameters settings	13	Set various flags	040F0803	Ack or Nack
Led Blink Behavior	17	Specify a certain blink behaviour for the button led (Home version ONLY)	0A0107D0006401F4024F	Ack or Nack
Linkcheck period setting	18	Set the linkcheck Period to 1440 minutes	0405A0A1	Ack or Nack

6.2 Uplink message specification

In this chapter will be analyzed in detail the message transmission on specific port.

6.2.1 Message on port 2. *Last downlink sequence number received*

This message is basically a service message and report to the server the last downlink counter value received from the sensor.
The downlink counter is part of authentication mechanism of LoRaWAN protocol and avoid that the previously message can be reproduced and retransmitted from the server to sensor in a typical "*man-in-the middle"* attack.
If a duplicated downlink counter is received, a transmission on port 2 is generated with the current downlink counter number. If the server is able to rebuild the message with the next counter number, the sensor will accept the message.
The Last Downlink number message is provided also if the server sends an "Enq" on port 2.
(ref. chapter 6.3.1)

6.2.2 Message on port 5. *Presentation*

The payload contains the sensor model string in ASCII format.
For this specific sensor, the string PB868LRx is transmitted in HEX format
(PB868LRH) 50423836384C5248
The presentation message is sent ONLY every time the sensor performs a reboot.
The reboot is caused by power-on reset or by a server command.
The presentation message is provided also if the server sends an "Enq" on port 5. (ref. chapter 6.3.2)

6.2.3 Message on port 6. *Serial Number*

The payload contains the serial number string in Hexadecimal format. 8 bytes length (e.g. AA112233445566FF)
The Serial Number message is provided ONLY if the server sends an "Enq" on port 6. (ref. chapter 6.3.3)

6.2.4 Message on port 7. *FW release, library release, HW release*

The payload contains the firmware release, the LoRaWAN Stack release and the Hardware revision.
This information is provided at the start-up of the sensor, and if the server sends an "Enq" on port 7. (ref. chapter 6.3.4)
The fields are not divided by separator char.

An example of message is: 010B0A4F04030F42
Where:

Value HEX	Description	width	range	meaning
01	Major release	1 byte	Binary from 0 to 0xFF Hex	Firmware release 1.11.2639
0B	Minor release	1 byte	Binary from 0 to 0xFF Hex	
0A 4F	Build	2 byte	Binary from 0 to 0xFFFF Hex	
04	Major rel. of LoRaWAN stack	1 byte	Binary from 0 to 0xFF Hex	LoRaWAN stack release 4.3.15
03	Minor rel. of LoRaWAN stack	1 byte	Binary from 0 to 0xFF Hex	
0F	Build rel. of LoRaWAN stack	1 byte	Binary from 0 to 0xFF Hex	
42	Hardware revision	1 byte	ASCII Format from A to Z	HW rel. B

6.2.5 Message on port 8. *Battery level*

The payload contains the battery charge percentage level.
The battery Level message is sent spontaneously on port 8 if the battery charging level is below 25% or if the server sends an "Enq" on port 8. (ref. chapter 6.3.56.3.2)

This message is automatically repeated every 6 hours when the battery is below the 25% of charge level.

An example of message is: 5E
Where:

Value HEX	Description	width	range	meaning
5E	Percentage of battery charge	1 byte	Binary from 0 to 0x64 hexadecimal	Level battery from 0% to 100%

6.2.6 Message on port 9. *Alive*

The payload contains the ALIVE message.
This type of message is intended ONLY to verify if the sensor is operating, by sending a message at regular interval.
The programmable interval is between 15 and 172800 seconds (48 hours)
In this message, in the payload are reported other information useful to know the sensor state.
In this message is also present the battery percentage level charge and the flags of the status of the sensor.

An example of message is: 5E010000
Where:

Value HEX	Description	width	range	meaning
5E	Percentage of battery charge	1 byte	From 0 to 0x64 Hex	Level battery from 0% to 100%
01	Event Flag (see chapter 4.4 pag.12)	1 byte	From 0 to 0x0F Hex	Report the status of the device
0000	Push Counter	2 bytes	From 0 to 0xFFFF Hex	Report the counter

6.2.7 Message on port 10. *Ack / Nack*

To inform the server that messages are received, the sensor sends back to the server a confirmation message of ACK or NACK, followed by the number of port where the message has been received.
This mechanism is used ONLY for messages that don't expected an explicit response with data from sensor.
For example, ALIVE interval setting do not respond with data, but confirms the reception with ACK or NACK.
The messages that expected an explicit response from sensor, do not response with ACK/NACK but directly with data. An example of that is the request of firmware revision.
ACK is transmitted if the received messages are correct in the length, checksum and semantic.
If the received message is corrupt, the sensor response is NACK; in either cases ACK and NACK is followed by the number of port where the message has been received.

An example of messages are:	41636b09	(Ack on port 9 for alive setting message ok)
	4e61636b09	(Nack on port 9 for alive setting message corrupted)

Where:

Value HEX	Description	width	range	meaning
41636b 4e61636b	Ack Nack	3 or 4 bytes	Not applicable	Ack or Nack
09	Port	2 bytes	0x01 to 0xDC Hex	Port of incoming message

6.2.8 PB868LRx Message on port 40. *Button event*

The sensor, basically is a device that sends messages when the PB sensor detects a button pression or a button release.

The payload is in raw format (3 Bytes)

Name	Type	Function
EVENT	Unsigned char (8 bits)	Event flag. See chapter 4.4.1 Event flag status pag.12
CONTER	Unsigned int (16 bits BIG ENDIAN)	***Detection Counter Number***

The sensor sends **spontaneously** a message with the above structure to the server if:

- If the tamper switch changes its status (from CLOSE to OPEN and vice versa)
- If the battery level reaches the 25% of the full charge.
- If **MODE 1** has been programmed, the sensor sends ***Pressions Counter*** value to the server every time a button pression or a button release is detected. ***Pressions Counter*** is never reset.
- If **MODE 2** has been programmed, the sensor sends a message to the server if ***Pressions Counter*** reach the programmed value.
- If **MODE 3** has been programmed, the sensor sends a message when ***Periodic Interval Time*** is elapsed or if ***Pressions Counter*** reach the programmed value
- If **MODE 4** has been programmed, the sensor sends a message to the server when the button is pressed the first time, and when the ***guard time*** is elapsed after the last press.

6.3 Downlink message specification

The sensor is fully configurable from application server.
The nature of LoraWAN class A permits to exchange messages only when the end-device transmits data to the server (uplink).
After sending the data, the end-device enable two RX windows to receive packets from the server.
In these windows, the server has the opportunity to send the data at the end node using specifics LoRaWAN protocol ports.
Downlink communications from the server at any other time different from the above mentioned RX windows, will have to wait until the next scheduled uplink occurs.

Depending on which port the message is received, the sensor performs different action; for example, the ***'Enq'*** message is the same for several requests, but assume different role based on which port the message is received.

Unlike upload messages, the downlink messages are completed with information as message length and checksum control. This is a precaution from sensor side to avoid that wrong messages from server causing unpredictable behaviour of the sensor.

Every message from server to sensor has 1 byte header that contains the total length of the message and 1 byte footer that contains the checksum.
The checksum is calculated doing a logical XOR of all the bytes on the message except the last one, which is the checksum itself.
Refer to ***Message Builder*** tool to easily build and verify the correct message to send to the sensor.

Header (1 Byte) Total message length	Payload contains the data	Footer (1 Byte) Checksum byte

6.3.1 Message on port 2. *Enquiry last downlink sequence number*

This message from server requests the last sequence counter received by the device.
When the sensor receives an ***'Enq'*** on port 2, the device replies with a message that contains a 32 bits dword with the last sequence counter received. The value is represented in BIG ENDIAN.

An example of message is: 05456E715F
Where:

Value HEX	Description	width	range	meaning
05	Message length	1 byte	From 0 to 0xFF HEX	Total length 5 bytes
456E71	Payload '*Enq*'	3 bytes	Enq	Enquiry command
5F	Message checksum (XOR)	1 byte	From 0 to 0xFF HEX	Checksum

The device replies with the last sequence counter received. (ref. chapter 6.2.1 pag.216.2.2)

6.3.2 Message on port 5. *Enquiry model*

This message from server, request the model of the sensor.
When the sensor receives an ***'Enq'*** on port 5, the device replies with a message containing his model.
Example 50423836384C5248 (PB868LRH)

An example of message is: 05456E715F
Where:

Value HEX	Description	width	range	meaning
05	Message length	1 byte	From 0 to 0xFF hexadecima	Total length 5 bytes
456E71	Payload '*Enq*'	3 bytes	Enq	Enquiry command
5F	Message checksum (XOR)	1 byte	From 0 to 0xFF hexadecima	Checksum

The device replies with model string. (ref. chapter 6.2.2 pag. 21)

6.3.3 Message on port 6. *Enquiry Serial number of device*

This message from server requests the serial number of device.
When the sensor receives an ***'Enq'*** on port 6, the device replies with a message containing his S/N.
Example AA112233445566FF

An example of message is: 05456E715F
Where:

Value HEX	Description	width	range	meaning
05	Message length	1 byte	From 0 to 0xFF hexadecima	Total length 5 bytes
456E71	Payload '*Enq*'	3 bytes	Enq	Enquiry command
5F	Message checksum (XOR)	1 byte	From 0 to 0xFF hexadecima	Checksum

The device replies with own serial number. (ref. chapter 6.2.3 pag.21)

6.3.4 Message on port 7. *Enquiry Firmware release, LoraWAN Lib release, HW revision*

This message from server requests the revision of device.
When the sensor receives, an ***'Enq'*** on port 7, the device transmits a message that contains:

- Firmware release
- LoRaWAN Library release
- Hardware revision

The Firmware release and LoRaWAN Library are identified by the scheme: MAJOR, MINOR and BUILD.
The MAJOR and MINOR values will be between 0 and 255 (0x0 to 0XFF); the BUILD will be a value from 0 to 65535 (0x0 to 0xFFFF).
Hardware revision will be only a literal value in the range from A to Z.

For example:
Firmware release 1.11.2639, LoRaWan stack Release 4.3.15, Hardware release B
Will be reported to the server with the message 010B0A4F04030F42.

An example of message is: 05456E715F
Where:

Value HEX	Description	width	range	meaning
05	Message length	1 byte	From 0 to 0xFF Hex	Total length 5 bytes
456E71	Payload '*Enq*'	3 bytes	Enq	Enquiry command
5F	Message checksum (XOR)	1 byte	From 0 to 0xFF Hex	Checksum

The device replies with firmware, library and Hardware revision. (ref. to chapter 6.2.4 pag. 21)

6.3.5 Message on port 8. *Enquiry Battery level*

This message from server requests the battery level of device.
When the sensor receives an ***'Enq'*** on port 8, the device replies with the value in percentage of the battery level, the value is in the range 0 to 100%.
The percentage is transmitted in Hexadecimal format.

An example of message is: 05456E715F
Where:

Value HEX	Description	width	range	meaning
05	Message length	1 byte	From 0 to 0xFF HEX	Total length 5 bytes
456E71	Payload '*Enq*'	3 bytes	Enq	Enquiry command
5F	Message checksum (XOR)	1 byte	From 0 to 0xFF HEX	Checksum

The device replies with battery level in percentage.

6.3.6 Message on port 9. *ALIVE setting interval value*

This message from server sets the value of the timeout for the ALIVE signal.
This value is expressed in seconds and is in the range from 60 to 172800 seconds (48 hours).
When a valid interval value is received the sensor transmits an ACK message to the server with the number of port where this message has received, if the interval is out of range or the message is corrupted, the sensor replies to the server with NACK. (ref. chapter 6.2.7).

The factory default value is: 3000 seconds (50 minutes)

An example of message is: 0400000BB8B7
Where:

Value HEX	Description	width	range	meaning
06	Message length	1 byte	From 0 to 0xFF Hex	Total length 10 bytes
00000BB8	Interval in seconds	4 bytes	From 60 to 172800 seconds	3000 seconds
B7	Message checksum (XOR)	1 byte	From 0 to 0xFF Hex	Checksum

The device replies with Ack or Nack. (ref. chapter 6.2.6 pag. 22)

6.3.7 Message on port 13. *Device configuration flags*

The Device configuration flags command allow to perform actions on the behavior of the sensor, or to command several actuators inside the device.

For complete description of this flag, see chapter 4.4.3 pag. 14
An example of message is: 040F0803
Where:

Value HEX	Description	width	range	meaning
04	Message length	1 byte	From 0 to 0xFF Hex	Total length 4 bytes
0F08	Bit Mask Flag	2 bytes	From 0 to 65535	Refer to chapter 4.4.3
03	Message checksum (XOR)	1 byte	From 0 to 0xFF Hex	Checksum

The device replies with Ack or Nack. (ref. chapter 6.2.7 pag. 23)

6.3.8 Message on port 17. *Led blink behaviour*

This message is only valid for the home-styled push button. It won't have effect on the industrial version of this device.

This kind of message allow the user to specify a specific behaviour for the led located on top of the home-styled push button. The parameters that can be specified are:

- The period of the blinking cycle
- The time_on of the led
- The time_off of the led
- The number of blinks

The time_on parameter is the time for the led to remain on. The time_off parameter is the time for the led to remain off.

An example of message is: 0A0107D0006401F4024F
Where:

Value HEX	Description	width	range	meaning
0A	Message length	1 byte	From 0 to 0xFF Hex	Total length 10 bytes
01	Activation Byte	1 byte	00 or 01	Blink ON
07D0	Blink Period	2 bytes	From 0000 to FFFF Hex	Period of 2000 ms
0064	Time ON	2 bytes	From 0000 to FFFF Hex	Time_on of 100 ms
01F4	Time OFF	2 bytes	From 0000 to FFFF Hex	Time_off of 500 ms
02	Blinks number	1 byte	From 00 to FF Hex	2 blinks with the specified Rules, in the specified period
4F	Message checksum (XOR)	1 byte	From 0 to FF Hex	Checksum

6.3.9 Message on port 18. *LINKCHECK setting interval value*

This message from server sets the value of the Linkcheck request.
This value is expressed in minutes and is in the range from 720 (12 Hours) to 65535 minutes (1092.25 hours, 45.5 days).
When a valid interval value is received the sensor transmits an ACK message to the server with the number of port where this message has received, if the interval is out of range or the message is corrupted, the sensor replies to the server with NACK. (ref. chapter 6.2.7).
Note that after the configured period no message is generated, but the sensor will forward to the server a link check request with the next message. This for limiting the number of messages sent.

The factory default value is: 720 seconds (12 hours)

An example of message is: 0405A0A1

To disable this function: 04000004

Where:

Value HEX	Description	width	range	meaning
04	Message length	1 byte	From 0 to 0xFF Hex	Total length 4 bytes
05A0	Interval in minutes	2 bytes	From 720 to 65535 minutes	1440 minutes
A1	Message checksum (XOR)	1 byte	From 0 to 0xFF Hex	Checksum

The device replies with Ack or Nack.

7 Uart Interface

All of the module's settings and commands are transmitted over UART using the ASCII interface.
All commands need to be terminated with <CR><LF> and any replies they generate will be terminated by the same sequence.
The settings for the UART interface are 115200 bps, 8 bits, no parity, 1 Stop bit, no flow control. The baud rate cannot be changed.
Because the sensor operates in CLASS A in low power state, the UART is put in idle mode for reduce the consumption. For wake the processor is needed to send a break condition with length equal to 10mS.
The processor remain in wake condition until the UART continues to receive characters; if the UART doesn't receive data for above 15 seconds, the processor return in sleep mode and for wake it again another break condition has to be sent.

8 Command Syntax

To issue commands to the ACL868 module, the user sends keywords followed by optional parameters. Commands and parameters are case insensitive. Hex input data can be uppercase or lowercase.
String text data, such as **`ON`** or **`OFF`**, are case-insensitive.
Depending on the command, the parameter may expect values in either decimal or hexadecimal form; refer to the command description for the expected form. For example, when configuring the frequency, the command expects a decimal value in Hertz such as **`869525000`** (869.525 MHz).
Alternatively, when configuring the LoRaWAN device address, an hex value parameter is expected (as **`aabbccdd)`**. To enter a number in hex form, use the value directly. For example, the hex value **`0xFF`** would be entered as **`FF`**.

8.1 Command organization

There are four general command categories, as shown in Table 2 Command type

Command	Keyword	Description
General	*none*	Issues various type of command '*general purpose'*
LoRaWAN™ Protocol	<**mac**>	Issues LoRaWAN protocol network communication behaviours, actions and configurations commands.
Device	<**dev**>	Issues device specific configurations, directly accessing and updating the mode of function
Radio	<**rad**>	Issues radio specific configurations, directly accessing and updating the transceiver setup.
System	<**sys**>	Issues system level behaviour actions, gathers status information on the firmware and hardware version.

Table 2 Command type

All configuration must be saved in the flash memory to avoid configuration loss on the next reboot or reset.
The MAC command, involve the communication behaviours in the LoRaWAN network.
For use the new configuration, the module must be reset with a reboot command or power-on cycle.

9 Shell access level

To gain the access to the shell, is mandatory supply the password.
The shell is protected by two passwords with different level of privileges, in this document the passwords are named USER and SUPER-USER and are abbreviated with **US**(user) and **SU**(super-user) acronym.
The various shell prompt are:

>	when no password is entered. In this state the possible actions are very limited.
US>	when the user password in entered. In this state the actions are partially limited.
SU>	when the super user password in entered. In this state the actions are totally enabled.

Note

When the SAVE command is entered to store the parameters in flash memory, the access level will be revoked and the prompt return to **>** *automatically*

The command for switch trough the levels is **pw** followed by desired level.

To gain access with user privileges enter:	**pw user** followed by the password
To gain access with super-user privileges enter:	**pw suser** followed by the password

The default passwords are different for any sensor and are supplied by ASCOEL.
The passwords are modifiable by respective user or super-user.

10 General Command

General commands are used to execute actions without supply parameters.

Command	Password	Description
info	no	Show various information about the sensor. The informations depend from the device type.
reboot	no	Reboot the sensor immediately.
default	super-user	Load the default parameters. The parameters depend from the device type.
save	no	Store ALL parameters in flash memory.
pw	yes	To change password or gain access to the shell.

11 MAC Command

MAC commands are common for all type of the sensors and define the access behaviour to the LoRaWAN network.
MAC commands are very important for the correct operating of the sensors, an access with super-user password to the shell is needed for most of those parameters.
For showing some parameters, the user level is sufficient.

For access to the shell in super-user mode:
>pw suser <*suser password*> The prompt will switch to **SU>**

SU>mac <action> <parameter> <value>

11.1 DevEui Command

Parameter	action	Password	Description
deveui	set	Suser	This command sets the globally unique device identifier for the module. The module contains a pre-programmed unique EUI and can be retrieved using the **`mac get deveui`** command. This command is not available for the user.
	get	User	This command returns the globally unique end-device identifier, as set in the module. Default: 0000000000000000 Example: **`mac get deveui`** *Response example*: **`deveui 0E7E346401AB02CC`** 8-bytes hexadecimal number representing the device EUI.

11.2 Devaddr Command

Parameter	action	Password	Description
devaddr	set	Super-user	4-byte hexadecimal number representing the device address, from 00000000 – FFFFFFFF. The address must be unique in the current network and is used only for ABP (activation by personalization) devices. For OTAA (over-the-air activation) this parameter is useless. Default: 0000000000000000 Example: **`mac set devaddr 01AB02CC`** *Response*: **`devaddr OK`** *if address is valid* **`devaddr error: invalid value`** *if address is out of range*
	get	User	This command will return the current end-device address of the module. Default: 00000000 *Example*: **`mac get devaddr`** *Response example*: **`devaddr 01AB02CC`** 4-bytes hexadecimal number representing the device address.

11.3 Appeui Command

Parameter	action	Password	Description
appeui	set	Super-user	8-byte hexadecimal number representing the application EUI. This command sets the application identifier for the module. The application identifier should be used to identify device types (sensor device, lighting device, etc.) within the network. Default: 0000000000000000 Example: **`mac set appeui A456FFC199A123F5`** *Response*: **`appeui OK`** *if value is valid* **`appeui error: invalid value`** *if value is out of range*
	get	User	This command will return the current application EUI of the module. Default: 0000000000000000 *Example*: **`mac get appeui`** *Response example*: **`appeui A456FFC199A123F5`** 8-bytes hexadecimal number representing the application EUI.

11.4 NwkSkey Command

Parameter	action	Password	Description
nwkskey	set	Super-user	16-byte hexadecimal number representing the network session key. The key should remain the same until the communication session between devices is terminated. Default: 2B7E151628AED2A6ABF7158809CF4F3C Example: **`mac set nwkskey A456FFC199A123F501FA1345CF34F516`** *Response*: **`nwkskey OK`** *if value is valid* **`nwkskey error: invalid value`** *if value is out of range*
	get	Super-user	This command will return the current network session key. Default: 2B7E151628AED2A6ABF7158809CF4F3C *Example*: **`mac get nwkskey`** *Response example*: **`nwkskey 2B7E151628AED2A6ABF7158809CF4F3C`** 16-bytes hexadecimal number representing the network session key.

11.5 AppSkey Command

Parameter	action	Password	Description
appskey	set	Super-user	16-byte hexadecimal number representing the application session key. This key is unique, created for each occurrence of communication, when the network requests an action taken by the application Default: 2B7E151628AED2A6ABF7158809CF4F3C Example: **`mac set appskey A456FFC199A123F5`** *Response*: **`appskey OK`** *if value is valid* **`appskey error: invalid value`** *if value is out of range*
	get	Super-user	This command will return the current application session key. Default: 2B7E151628AED2A6ABF7158809CF4F3C *Example*: **`mac get appskey`** *Response example*: **`appskey 2B7E151628AED2A6ABF7158809CF4F3C`** 16-bytes hexadecimal number representing the application session key.

11.6 Appkey Command

Parameter	action	Password	Description
appkey	set	Super-user	16-byte hexadecimal number representing the application key. The application key is used to identify a grouping over module units which perform the same or similar task Default: 2B7E151628AED2A6ABF7158809CF4F3C Example: **`mac set appkey A456FFC199A123F5`** *Response*: **`appkey OK`** *if value is valid* **`appkey error: invalid value`** *if value is out of range*
	get	Super-user	This command will return the current application key. Default: 2B7E151628AED2A6ABF7158809CF4F3C *Example*: **`mac get appkey`** *Response example*: **`appkey 2B7E151628AED2A6ABF7158809CF4F3C`** 16-bytes hexadecimal number representing the application session key.

11.7 Pwridx Command (in current FW revision this command is disabled)

Parameter	action	Password	Description
pwridx	set	Super-user	Decimal number representing the index value for the output power, from 0 to 5 for 433 MHz frequency band and from 1 to 5 for 868 MHz frequency band. Default: 5 Example: **`mac set pwridx 5`** *Response*: **`pwridx OK`** *if value is valid* **`pwridx error: invalid value`** *if value is out of range*
	get	User	This command will return the current application session key. Default: 5 *Example*: **`mac get pwridx`** *Response example*: **`pwridx 5`** 1-bytes decimal number representing the RF power.

11.8 Dr Command (in current FW revision this command is disabled)

Parameter	action	Password	Description
dr	set	Super-user	Decimal number representing the data rate, from 0 and 7, but within the limits of the data rate range for the defined channels. This command sets the data rate to be used for the next transmission. Please refer to the *LoRaWAN™ Specification* for the description of data rates and the corresponding spreading factors Default: 0 Example: `mac set dr 3` *Response*: `dr OK` *if value is valid* `dr error: invalid value` *if value is out of range*

	get	User	This command will return the current datarate.
			
			Default: 0
			
			Example:
			`mac get dr`
			
			Response example:
			`dr 3`
			1-bytes decimal number representing the data rate.

11.9 Adr Command

Parameter	action	Password	Description
adr	set	Super-user	This command turns ON or OFF the adaptive data rate (ADR) control. The server is informed about the status of the module's ADR in every uplink frame it receives from the ADR field in uplink data packet. If ADR is enabled, the server will optimize the data rate and the transmission power of the module based on the information collected from the network. Default: ON Example: `mac set adr ON` *Response*: `adr OK` *if value is valid* `adr error: invalid value` *if value is out of range*

	get	User	This command will return the state of ADR control.
			
			Default: ON
			
			Example:
			`mac get adr`
			
			Response example:
			`adr OFF`
			ON or OFF string.

11.10Confirmed/unconfirmed port Command

Parameter	action	Password	Description
confport	set	Super-user	A confirmed message will expect an acknowledgment from the server; otherwise, the message will be retransmitted by the number indicated by the command **`mac set retry <value>,`** whereas an unconfirmed message will not expect any acknowledgment back from the server. Please refer to the *LoRaWAN™ Specification* for further details. The **confport** command, permit to set up to 10 ports where the message will be treated as confirmed type. For example, if is desired that messages sent on port 20 to be confirmed type, the command is **`mac confport 20`**. Is possible set up to 10 ports for use as confirmed, Every time that **`mac confport`** ***`<port number>`*** is entered, the <*port number*> list will be overwritten. If the list is full or if the port number is invalid or already present, the processor answer with error message. To delete the whole list, the command is: **`mac set confport none`** Default: **`none`** (no confirmed port) Example: **`mac set confport 3`** *Response*: **`confport OK`** *if value is valid* **`confport error: invalid value`** *if value is out of range*
	get	User	This command will return the list of *confirmed ports*. Default: **`none`** *Example*: **`mac get confport`** *Response example*: **`Confport 8 20 31`** A list of decimal values of port number.

11.11 Retry Command

Parameter	action	Password	Description
retry	set	Super-user	Decimal number representing the number of retransmissions for a confirmed uplink packet, from 1 to 8. Default: 1 Example: `mac set retry 3` *Response*: `retry OK` *if value is valid* `retry error: invalid value` *if value is out of range*

	get	User	This command will return the current retry number.
			
			Default: 0
			
			Example:
			`mac get retry`
			
			Response example:
			`retry 3`
			1-bytes decimal number representing the number of retransmissions, from 1 to 8.

11.12 Link Check Command

Parameter	action	Password	Description
linkchk	set	Super-user	Decimal number that sets the time interval in minutes for the link check process, from 720 to 65535. This command sets the time interval for the link check process to be triggered periodically. A `<value>` of '`0`' will disable the link check process. When the time interval expires, the next application packet that will be sent to the server will include also a link check MAC command. Please refer to the *LoRaWAN™ Specification* for more information on the Link Check MAC command Default: 720 Example: `mac set linkchk 1440` *Response*: `linkchk OK` *if value is valid* `linkchk error: invalid value` *if value is out of range*

	get	User	This command will return the current link check interval.
			
			Default: 720
			
			Example:
			`mac get linkchk`
			
			Response example:
			`linkchk 1440`
			Decimal number representing the interval in minutes for the next link check test.

11.13 Rx delay1 Command

Parameter	action	Password	Description
rxdelay1	set	Super-user	Decimal number representing the delay between the transmission and the first Reception window in milliseconds, from 0 to 65535. This command will set the delay between the transmission and the first Reception window to the `<rxDelay>` in milliseconds. The delay between the transmission and the second Reception window is calculated in software as the delay between the transmission and the first Reception window + 1000 (ms). Default: 1000 Example: **`mac set rxdelay1 1500`** *Response*: **`rxdelay1 OK`** *if value is valid* **`rxdelay1 error: invalid value`** *if value is out of range*
	get	User	This command will return the current RX1 delay. Default: 0 *Example*: **`mac get rxdelay1`** *Response example*: **`rxdelay1 1500`** Decimal number representing the RX1 delay in mS.

11.14 Rx2 frequency Command

Parameter	action	Password	Description
rx2freq	set	Super-user	Decimal number representing the frequency, from 863000000 to 870000000 Hz. This command sets the frequency for the second Receive window. The configuration of the Receive window parameters must be in concordance with the server configuration Default: 869525000 Example: **`mac set rx2freq 869525000`** *Response*: **`rx2freq OK`** *if value is valid* **`rx2freq error: invalid value`** *if value is out of range*
	get	User	This command will return the current second Receive window frequency. Default: 869525000 *Example*: **`mac get rx2freq`** *Response example*: **`rx2freq 869525000`** Decimal number representing the frequency in Hz.

11.15 Rx2 data rate Command

Parameter	action	Password	Description
rx2dr	set	Super-user	Decimal number representing the data rate for RX2, from 0 to 7. This command sets the data rate for the second Receive window. The configuration of the Receive window parameters must be in concordance with the server configuration Value / Spreading factor: 0 SF12; 1 SF11; 2 SF10; 3 SF9; 4 SF8; 5 SF7; 6 FSK; 7 TBD Default: 0 Example: **`mac set rx2dr 3`** *Response*: **`rx2dr OK`** *if value is valid* **`rx2dr error: invalid value`** *if value is out of range*
	get	User	This command will return the current datarate in RX2 window. Default: 0 *Example*: **`mac get rx2dr`** *Response example*: **`rx2dr 3`** Decimal number representing spreading factor used in RX2 window.

Value	Spreading factor
0	SF12
1	SF11
2	SF10
3	SF9
4	SF8
5	SF7
6	FSK
7	TBD

11.16Duty cycle Command

Parameter	action	Password	Description
dutycycle	set	Super-user	This command enables or disables the control of dutycycle. The duty cycle of radio devices is often regulated by government. If this is the case, the duty cycle is commonly set to 1%, but make sure to check the regulations of your local government to be sure. In Europe, duty cycles are regulated by section 7.2.3 of the ETSI EN300.220 standard. Please refer to the *LoRaWAN™ Specification* for further details. The control should be turned OFF **ONLY** for test purpose. Default: ON Example: `mac set dutycycle ON` *Response*: `dutycycle OK` *if value is valid* `dutycycle error: invalid value` *if value is out of range*

	get	User	This command will return the state of Duty Cycle control.
			Default: ON
			Example:
			`mac get dutycycle`
			Response example:
			`Dutycycle OFF`
			ON or OFF string.

11.17OTAA Command

Parameter	action	Password	Description
otaa	set	Super-user	This command chooses the method of provisioning the sensor on the network server. If OTAA is set to YES, automatically the ABP (Activation By Personalization) is switched OFF. Vice-versa if OTAA is set to NO, the ABP mode is enabled. Please refer to the *LoRaWAN™ Specification* for further details. Default: NO Example: **`mac set otaa ON`** *Response*: **`otaa OK`** *if value is valid* **`otaa error: invalid value`** *if value is out of range*
	get	User	This command will return the state of OTAA. Default: ON *Example*: **`mac get otaa`** *Response example*: **`Otaa YES`** YES or NO string.

11.18Class Command

Parameter	action	Password	Description
class	set	Super-user	This command choice the class of operation of the device. The available class are A or C. Please refer to the *LoRaWAN™ Specification* for further details. Default: A Example: **`mac set class C`** *Response*: **`class OK`** *if value is valid* **`class error: invalid value`** *if value is out of range*
	get	User	This command will return the selected Class. Default: ON *Example*: **`mac get class`** *Response example*: **`class C`** A or C string.

11.19 Password Command

Parameter	action	Password	Description
pw	none	Super-user or User	This command allows to change the current password or to gain the access to the shell. Every sensor is shipped with a specific set of passwords; these passwords are supplied by ASCOEL when the device is shipped. The passwords are freely modifiable by the *user* or *super-user* according with the own level access; to do this is mandatory to enter the own level password for set the new password. The max password length is 16 characters. Example: The current user password is '***user***' and the current super-user password is '***suser***' • To gain access to the shell in user level: *`pw user <current password>`* **`pw user user`** • To gain access to the shell in super-user level: *`pw suser <current password>`* **`pw suser suser`** • To change the user level password the command is: *`pw set user <current user pw> <new user pw>`* **`pw set user user`** ***`newpassword`*** • To gain access to the shell in super-user level: *`pw set suser <current super pw> <new super pw>`* **`pw set suser suser`** ***`newpassword`*** *Response*: **`user OK`** *if value is valid* **`user error`**

11.20 Default Command

Parameter	action	Password	Description
default	none	Super-user	This command allows to load the factory default values for all parameters. This command is executed only with super-user privileges. <u>Example:</u> **`default`** <u>*Response*</u>: **`default OK`** *if value is valid* **`default error`**

11.21 Save Command

Parameter	action	Password	Description
save	none	none	This command stores all parameters in non-volatile flash memory. <u>Example:</u> **`save`** <u>*Response*</u>: **`save OK`** *if value is valid* **`save error`**

11.22 Reboot Command

Parameter	action	Password	Description
reboot	none	none	Performs a reboot of the sensor. <u>Example:</u> **`reboot`** <u>*Response*</u>: **`none`**

11.23 Info Command

Parameter	action	Password	Description
info	none	none	Show some data or parameters related to the sensor. The response format varies between firmware releases and sensors type. <u>Example:</u> **`info`** <u>*Response*</u>: **`none`**

12 PB868LRx specific shell command (DEV)

DEV commands are specific for every type of the sensor and define the operating behaviour for which the sensor was developed.
The DEV commands allow setting some vital parameters, to use these commands the shell level access can be of type USER or SUPERUSER.
For showing the DEV parameters, the user level is sufficient.

To access the shell in superuser mode:
>pw suser <*suser password*> The prompt will switch to **SU>**

To access the shell in user mode:
>pw user <*user password*> The prompt will switch to **US>**

12.1 Alive interval Command

Parameter	action	Password	Description
alive	set	Superuser	The alive interval allows you to set a time expressed in seconds that regularly transmits a data packet to the server. For the contents of the packet, refer to the specific device manual. Default: 3000 (min. 15 max 172800 seconds) Example: **`dev set alive 5000`** *Response*: **`alive OK`** *if value is valid* **`alive error: invalid value`** *if value is out of range*
	get	User	This command will return the value of alive interval. Default: 3000 *Example*: **`dev get alive`** *Response example*: **`alive 18000`** A decimal value of alive interval, in seconds.

12.2 Device Flag Command

Parameter	action	Password	Description
flag	set	User	The flag is a 16-bit wide register where the sensor store some information. Refer to specific sensor chapter for detailed explanation of each available flag. Default: 0 Example: **`dev set flag 15`** *Response*: **`flag OK`** *if value is valid* **`flag error: invalid value`** *if value is out of range*
	get	User	This command will return the bit mask of flag. Default: 0 *Example*: **`dev get flag`** *Response example*: **`flag 15`** A decimal values a bit mask flag.

12.3 MODE 1

Parameter	action	Password	Description
mode1	set	User	Setting of the **mode1** of the functionally of the PB sensor. <u>Syntax:</u> **`mode1 (`*`no parameter needed`*`)`** <u>Default:</u> **`nothing`** <u>Example:</u> **`dev set mode1`** *<u>Response</u>*: **`mode1 OK`** *if value is valid* **`mode1 error: invalid value`** *if value is out of range*

12.4 MODE 2

Parameter	action	Password	Description
mode2	set	User	Setting of the **mode2** of the functionally of the PB sensor. <u>Syntax:</u> **`mode2 <counter number> <flag>`** <u>Default:</u> **`counter number 1`** **`flag 0`** <u>Example:</u> **`dev set mode2 20 1`** *<u>Response</u>*: **`mode2 OK`** *if value is valid* **`mode2 error: invalid value`** *if value is out of range*

12.5 MODE 3

Parameter	action	Password	Description
mode3	set	User	Setting of the **mode3** of the functionally of the PB sensor. <u>Syntax:</u> **`mode3 <interval time> <counter number> <flag>`** <u>Default:</u> **`interval time 15 seconds`** **`counter number 1`** **`flag 0`** <u>Example:</u> **`dev set mode3 20 5 2`** *<u>Response</u>*: **`mode3 OK`** *if value is valid* **`mode3 error: invalid value`** *if value is out of range*

12.6 MODE 4

Parameter	action	Password	Description

mode4	set	User	Setting of the **mode4** of the functionally of the PB sensor.
			
			<u>Syntax:</u>
			`mode4 <guard time> <periodic interval> <flag>`
			
			<u>Default:</u>
			`guard time 600 seconds`
			`periodic interval number 15`
			`flag 0`
			
			<u>Example:</u>
			`dev set mode4 300 20 8`
			
			<u>Response</u>:
			`mode4 OK` *if value is valid*
			`mode4 error: invalid value` *if value is out of range*

12.7 Mode Command (get only)

Parameter	action	Password	Description
mode	get	User	This command will return the MODE currently programmed. _Example_: **`dev get mode`** _Response example for:_ **MODE1**: **`mode1`** **MODE2**: Syntax: **`mode2 <counter number> <flag>`** Example: **`mode2 15 2`** **MODE3**: Syntax: **`mode3 <interval time> <counter number> <flag>`** Example: **`mode3 300 20 2`** **MODE4**: Syntax: **`mode4 <guard time> <periodic interval> <flag>`** Example: **`mode4 600 60 8`**

12.8 Model Command (get only)

Parameter	action	Password	Description

model	get	User	This command will return the model of sensor.
			
			Default: any
			
			Example:
			`dev get model`
			
			Response example:
			`model PB868LRH`
			String that representing the model of the sensor.

12.9 Production date Command (get only)

Parameter	action	Password	Description

proddate	get	User	This command will return the production date of the sensor.
			
			Default: any
			
			Example:
			`dev get proddate`
			
			Response example:
			`proddate 06/07/17`
			String that representing the production date in the format dd/mm/yy.

12.10Serial number Command (get only)

Parameter	action	Password	Description

serial	Get	User	This command will return the serial number of the sensor.
			
			Default: none
			
			Example:
			`dev get serial`
			
			Response example:
			`Serial 363636365E368101`
			String that representing the serial number of the sensor.

12.11 Battery level Command (get only)

Parameter	action	Password	Description
battery	get	User	This command will return the current level of charge of the internal battery. The values reported represent the charge in several modes: Percentage, voltage, capacity, numeric and a flag for low battery threshold alert. Default: any *Example*: **`dev get battery`** *Response example*: **`battery 0000000000 100 3600 2400 254 0`** The response format is: **`battery <serial> <percentage> <voltage> <capacity> <level> <flag>`** where:

Value	Percentage
0 to 100	Percentage of the charge

Voltage
Voltage in mV

Capacity
Current in mAh

Value	level
0	External battery
1 to 253	Level charge
254	Fully charged
255	Unknown

Value	flag
0	Battery OK
1	Battery charge is below the 25%

`<serial>` is reserved for future implementation.

Contact Information

ASCOEL Srl
Via degli Artigiani 7 26025 Pandino -CR- ITALY
Phone: (+39)0373 970473
E-mail: info@ascoel.com
Internet: http://www.ascoel.it