ADT1 - LoRa data communication protocol

1 Introduction ... 3
2 Frame format (General structure) ... 3
2.1 Network server information ... 3
2.1.1 Device identification EUI ... 3
2.1.2 Time information ... 3
2.1.3 Transmission counter (Frame Up / Down) ... 3
2.1.4 Confirmed and unconfirmed transmissions (“Cnf”/”UnCnf”) ... 3
2.2 Data bytes / Payload encode and decode ... 4
2.2.1 Port ... 4
2.2.2 Payload ... 4
3 Function codes ... 5
3.1 1 (0x01) Measured values in float format (4 Byte) ... 5
3.2 12 (0x0C) “Info” message ... 7
3.2.1 Type 1 ... 7
3.3 31 (0x1F) 1 Byte variable ... 8
3.4 32 (0x20) 1 Byte variable - stream ... 8
3.4.1 Definition of variable function 31/32 (data type: unsigned char) ... 8
3.5 51 (0x33) 4 Byte variable ... 10
3.6 52 (0x34) 4 Byte variable - stream ... 10
3.6.1 Definition of variable function 51/52 (data type: unsigned long) ... 10
3.7 61 (0x3D) Float variable ... 12
3.8 62 (0x3E) Float variable - stream ... 12
3.8.1 Definition of variable function 61/62 (data type: float) ... 13
3.9 71 (0x47) ASCII characters ... 14

3.10 72 (0x48) ASCII characters - stream ... 14

3.10.1 Definition of variable function 71/72 (data type: ASCII characters) ... 15

3.11 81 (0x51) KELLER Sensor information ... 16

3.11.1 Sensor type 1 / RS485 ... 16

3.11.2 Sensor type 2 / I^{2}C ... 16

3.12 90 (0x5A) Command / Configuration ... 17

3.12.1 Definition of commando function 90 (data type: unsigned char) ... 17

4 Device type overview ... 18

5 Revision History ... 21

1 INTRODUCTION

This document describes the communication between the LoRa device (node) and the network server/cloud and how to interpret the data packets.

Communication can take place in both directions. Sending measured data from the device to the network server (uplink), and send configuration changes from the network server to the (downlink) device.

2 FRAME FORMAT (GENERAL STRUCTURE)

This describes the general structure of a LoRa data package.

2.1 Network server information

2.1.1 Device identification EUI

The DevEUI (Unique Identifier), which is determined with each transmission, is used to uniquely identify the devices. This EUI cannot be changed and I preconfigured on the device.

2.1.2 Time information

The Time information is transferred with the payload when the message is sent, or the message is time-stamped. This time can therefore be used as the measurement time. The time information will also be transmitted in the status information.

2.1.3 Transmission counter (Frame Up / Down)

The transmission counter is incremented on both sides for each message sent or received. Thus it can be determined how many transmissions have worked and how many have not.

2.1.4 Confirmed and unconfirmed transmissions (“Cnf”/”UnCnf”)

A confirmed or unconfirmed transmission has no influence on the content of the message, the advantage of a confirmed transmission is that the sender recognizes whether the message has been transmitted or received.

The disadvantage is that the energy consumption is slightly higher, since the receipt of the confirmation consumes additional energy. In addition, with each confirmed transmission, the network server confirms receipt of the packet, which results in a downlink. Since the downlink should be limited to a minimum, this can lead to additional costs depending on the network server.

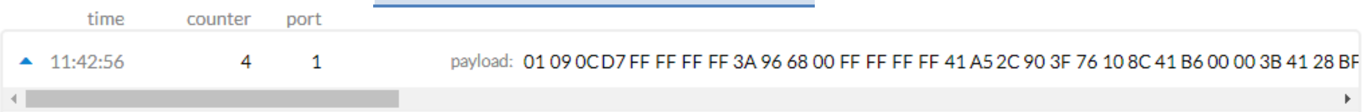
For these reasons we recommend unconfirmed transmissions. The basic configuration of the KELLER devices is “UnCnf” transmissions.

2.2 Data bytes / Payload encode and decode

Depending on the data type selected, the data is assigned to a specific port. The user data content is the payload.

This is an example from TTN (The Things Network)

Port	Payload (0…51 Bytes)
1…223	00x … FFx



Both port and payload are available in both directions (uplink and downlink).

2.2.1 Port

Depending on the data type selected, the data is assigned to a specific port. Thus the data type can be distinguished without interpretation of the payload.

Port	Meaning
1	Measurements
2	Alarm
3	Configuration
4	Info (Battery voltage, Humidity, Time …)
5	Answer on a request

2.2.2 Payload

The payload contains the user data. The size of the payload or the number of bytes transferred varies depending on the information or command. The maximum amount of data in one message is limited to 51 bytes.

The data bytes are transmitted in “ASCII” / “Hexadecimal” format. Groups of 2 “ASCII” characters form a byte in the data format Byte. This means that the payload (user content) is, for example, interpreted as follows:

Payload = 00AAF023 are 4 bytes that were transmitted.

- Byte 0 : 0x00 = 0
- Byte 1 : 0xAA = 170
- Byte 2 : 0x00 = 0
- Byte 3 : 0x23 = 35

Which information the transferred payload contains can be seen from the first data byte the “function code”.

Byte – N°	1	2	…	51
Meaning	Functions-code	user data	user data	user data

Function code	Type of data
1 (0x01)	“Measurement” message
12 (0x0C)	“Info” message
31 (0x1F)	1 Byte variable
32 (0x20)	1 Byte variable (Stream)
51 (0x33)	4 Byte variable
52 (0x34)	4 Byte variable (Stream)
61 (0x3D)	Float variable
62 (0x3E)	Float variable (Stream)
71 (0x47)	ASCII sign
72 (0x48)	ASCII sign (Stream)
81 (0x51)	KELLER pressure sensor information

3 FUNCTION CODES

The interpretation of the different function codes is described in the following chapters of this document.

3.1 1 (0x01) Measured values in float format (4 Byte)

Different numbers of measured values can be transmitted. The number and which channels are transmitted can be seen from the two “Channel” byte and the selected “Connection type” (see table below). Due the limitation of the payload length to 51 Bytes it is only possible to send maximum 11 channels in one message. The other channels will be transmitted in a second message.

Byte – N°	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	...	...	...	48
Meaning	1 (0x01)	Device type	Channel H	L	Value 1				Value 2				Value 3				...			

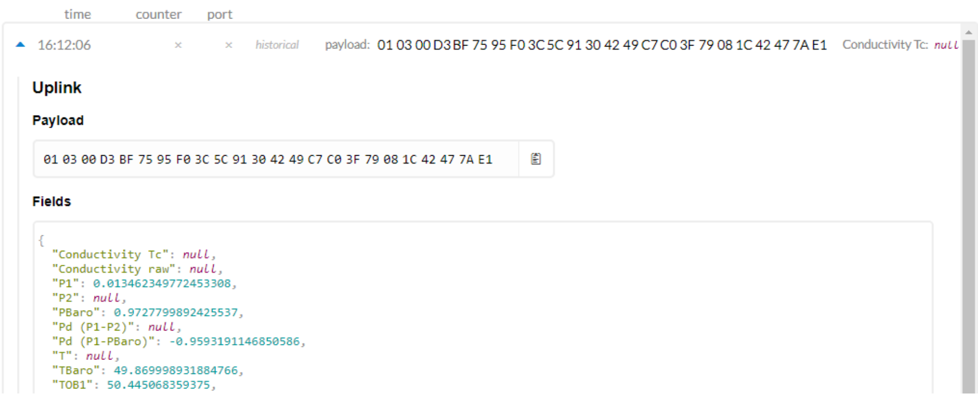
The **“Device type”** indicates the channel assignment. See "Device Type Overview".

The **“Channel”** shows you how many and which channels are transmitted in the message. Each bit stands for one channel, thus channel 1 … 16 are selectable.

Bit Position	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Channel	CH 16	CH 15	CH 14	CH 13	CH 12	CH 11	CH 10	CH 9	CH 8	CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1

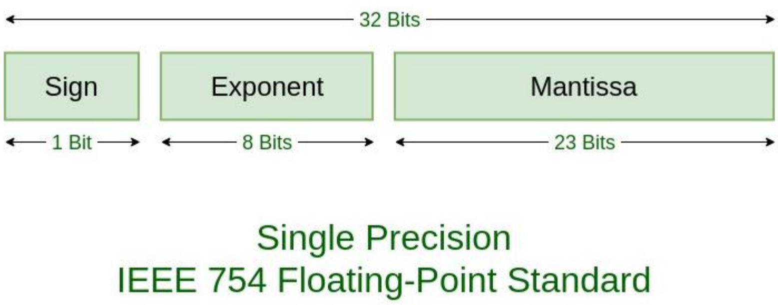
Example: Channel: 0000'0000 1000'0101 = Values 1 + 3 + 7 are contained in the message. The least significant bit of the channel is always the first value.

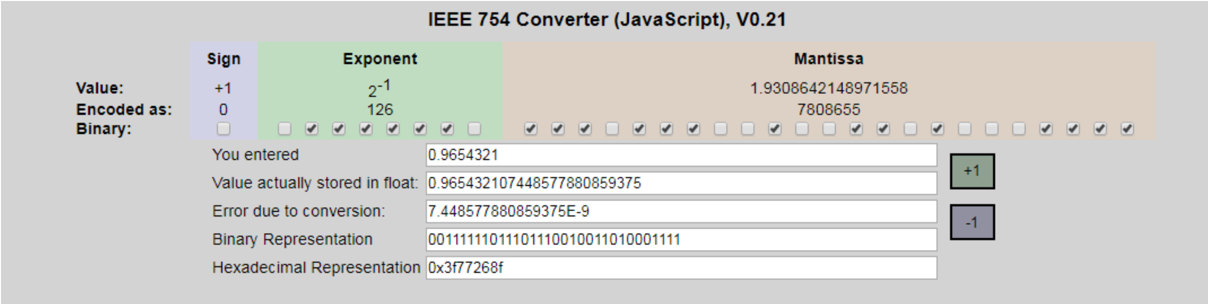
Example (TTN):



Payload:
010300D3BF7595F03C5C91304249C7C03F79081C42477AE1

The **“Value”** of the corresponding channels follows each other and each consists of 4 bytes and is in float IEE 754 format.




IEEE 754 Converter (JavaScript), V0.21
Sign
Exponent
Mantissa
Value:
Encoded as:
Binary:
+1
2-1
1.9308642148971558
0
126
7808655
You entered
0.9654321
Value actually stored in float:
0.965432107448577880859375
Error due to conversion:
7.448577880859375E-9
Binary Representation
00111111011101110010011010001111
Hexadecimal Representation
0x3f77268f
+1
-1

3.2 12 (0x0C) "Info" message

The "Info" message contains useful data of the device itself and actual status information of it.

3.2.1 Type 1

Byte – N°	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Meaning	12 (0x0C)	Type	Class / Group		SW Year / Week		Serial number				Time information			

15	16	17	18	19	20	21	22
Battery voltage				Battery capacity		Humidity	

The **"type"** is reserved for future use. Now it is always 1.

The **„Class / Group"** is represented by two hex values, where the first two chars are representing the class and the second two chars are representing the group.

The **„SW Year / Week"** is represented by two hex values, where the first two chars are representing the year and the second two chars are representing the week.

The **„Serial number"** is represented by four hex values.

The **"Time information"** represents the time in seconds from 1.1.2000 .Note that this time is a local time not UTC.

The **„Battery voltage"** is represented by four hex values, which consist of 4 bytes and are in float IEE 754 format.

The **„Battery capacity"** is represented by one byte.

The **„Humidity"** is represented by one byte.

Example (TTN):

Payload:
0C011300132F000000642576AFAA4098B368631D

Class.Group (Device Information):	19.00 (Class: 0x13 is 19, Group: 0x00 is 00)
SW-Version (Year.Week):	19.47 (Year: 0x13 is 19, Week: 0x2F is 47)
Serial number:	100 (0x00000064)
Time information (Device time):	2019-12-01 17:06:50 (in seconds after the year 1.1.2000 0:0:0 0x2576AFAA -> 628535210
Battery voltage:	4.7719 V (0x4098B368)
Battery capacity (calculated):	99 % (0x63)
Humidity:	16 % rel (0x1D)

3.3 31 (0x1F) 1 Byte variable

Different numbers of 1 byte values can be transmitted. There are always 2 byte packets, where the first byte is the index number “Index” and the second byte is the value “Val”

Byte – N°	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**…**	**51**
Meaning	31 (0x1F)	Index (n)	Val$_{Index}$ (n)	Index (y)	Val$_{Index}$ (y)	Index (z)	Val$_{Index}$ (z)	…	…

Index: Index number
Val: Value of the corresponding index

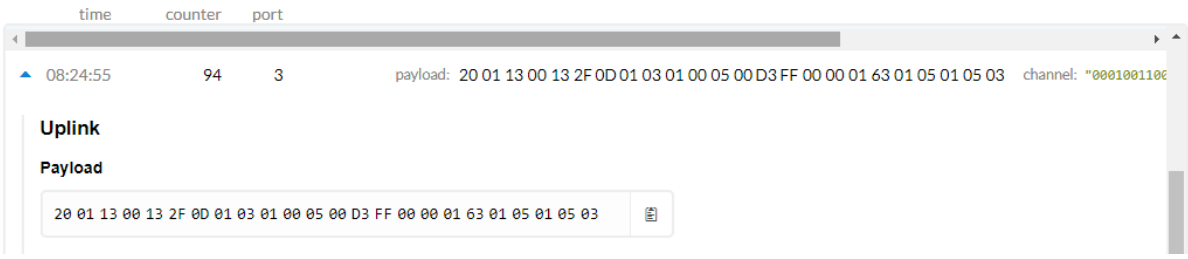
3.4 32 (0x20) 1 Byte variable - stream

A larger number of 1 byte values can be transmitted, whereby only the Index number “Index” of the first byte is specified, the following bytes are variable contents added in ascending order without gaps.

Byte – N°	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**…**	**51**
Meaning	32 (0x20)	Index (n)	Val$_{Index}$ (n)	Val$_{Index}$ (n+1)	Val$_{Index}$ (n+2)	Val$_{Index}$ (n+3)	Val$_{Index}$ (n+4)	…	…

Index: Index number (after each Byte increase index value by 1)
Val: Value of the corresponding index

Example (TTN):



Payload:
20011300132F0D010301000500D3FF000001630105010503

3.4.1 Definition of variable function 31/32 (data type: unsigned char)

Index	Description	Example / Range	**Adjustable / Type**	Firmware Version
1	Device Class	0x13 = 19	Ø	19.45
2	Device Group	0x00 = 00	Ø	19.45
3	SW Version Year	0x13 = 19	Ø	19.45
4	SW Version Week	0x2F = 47	Ø	19.45
5	supported “Device type” (defines how many Sensor types are supported from the firmware)	0x0D = 13	Ø	19.45
6	Uplink mode	0x01 = 1 (unconfirmed & OTAA)	✓	19.45

Bit Position	1	0
Mode	Type of message (unconfirmed / confirmed)	Join mode (ABP / OTAA)
	0 = unconfiormed 1 = confiormed	0 = ABP 1 = OTAA

Index	Description	Example / Range	Adjustable / Type	Firmware Version
7	Device type (see description “Device type overview”)	0x03 = 3	✓	19.45
8	Power for external device 0-> Deactivated	0x01 = 1 (external device will be powered)	✓	19.45

	1-> +3.5 V			
9	Power Pre-On time (Gives power to the external sensor defined seconds before read out)	0x00 = 0 sec (is not needed for KELLER devices)	✓	19.45
10	Lock Timer Bit Pos. 0-> Measure Bit Pos. 1-> Alarm Bit Pos. 2-> Info	0x05 = 5 -> Measure, Info, is active	✓	19.45
11	Measure / Save channels 8 … 15	0x00 = 0 -> 0000`0000 no channels selected	✓	19.45
12	Measure / Save channels 0 … 7	0xD3 = 211 -> 1101`0011 (Channel: 0,1,4,6,7)	✓	19.45
13	Event Channel (Channel 0 … 15)	Not used (reserved)	✓	19.45
14	Event-Type	Not used (reserved)	✓	19.45
15	Alarm Channel (Channel 0 … 15)	0x00 = 0 -> channel 0 is selected	✓	19.45
16	Alarm Type 1-> On / Off 2-> Delta	0x01 = 1 -> On / Off alarm is selected	✓	19.45
17	Battery capacity in [%]	0x63 = 99	✓	19.45
18	ADR (adaptive data rate) */ ** 0-> ADR OFF 1-> ADR ON	0x01= 1 -> ADR ON	✓	19.45
19	DR (data rate)*/ ** 0-> SF12 / 125kHz 1-> SF11 / 125kHz 2-> SF10 / 125kHz 3-> SF9 / 125kHz 4-> SF8 / 125kHz 5-> SF7 / 125kHz	0x05= 5 -> SF7 / 125kHz *can be different for other regions (radio band selection)	✓	19.45
20	Power Index* / ** 0-> 16dBm 1-> 14dBm (Default) 2-> 12dBm 3-> 10dBm 4-> 8dBm 5-> 6dBm 6-> 4dBm 7-> 2dBm	0x01= 14 dBm *can be different for other regions (radio band selection)	✓	19.45
21	Radio Band (select Region) ** 0-> AS923 (Asia) 1-> AU915 (Australia) 5-> EU868 (Europe / Default) 6-> KR920 (Korea) 7-> IN865 (India) 8-> US915 (USA) 9-> US915-HYBRID (USA)	0x05= EU868 (Europe)	✓	19.45
22	LoRa module type** 0-> Unknown 1-> RN2483 2-> RN2903 3-> ABZ-093	0x03 = 3 -> ABZ-093	∅	19.45

** This values will not be changed until the user sends the confirmation (function 90 / Index 11 (Accept the configuration)).

3.5 51 (0x33) 4 Byte variable

Different numbers of 4 byte values can be transmitted. There are always 5 byte packets, where the first byte is the index number “Index” and the following four bytes are the value “Val”

Byte – N°	1	2	3	4	5	6	7	8	9	10	11	…
Meaning	51 (0x33)	Index (n)	Val_{Index} HH(n)	Val_{Index} HL(n)	Val_{Index} LH(n)	Val_{Index} LL(n)	Index (y)	Val_{Index} HH(y)	Val_{Index} HL(y)	Val_{Index} LH(y)	Val_{Index} LL(y)	…

Index: Index number
Val: Value of the corresponding index

3.6 52 (0x34) 4 Byte variable - stream

A larger number of 4 byte values can be transmitted, whereby only the index number “Index” of the first byte is specified, the following bytes are variable contents added in ascending order without gaps.

Byte – N°	1	2	3	4	5	6	7	8	9	10	…
Meaning	52 (0x34)	Index (n)	Val_{Index} HH(n)	Val_{Index} HL(n)	Val_{Index} LH(n)	Val_{Index} LL(n)	Val_{Index} HH(n+1)	Val_{Index} LH(n+1)	Val_{Index} LH(n+1)	Val_{Index} LL(n+1)	…

Index: Index number (after each Byte increase index value by 1)
Val: Value of the corresponding index

Example (TTN):

10:31:09 102 3 payload: 34 01 00 00 00 64 25 76 60 38 00 00 00 25 25 76 6B 51 25 4A D8 30 25 77 84 5F 00 00 00 00 00 00

Uplink

Payload

34 01 00 00 00 64 25 76 60 38 00 00 00 25 25 76 6B 51 25 4A D8 30 25 77 84 5F 00 00 00 00 00 00 0E 10 00 01 51 80 00 01 51 80 00 00 00 00 00 00 00 00

Payload:
3401000000642576603800000002525766B51254AD8302577845F000000000000E1000015180000151800000000000000000

3.6.1 Definition of variable function 51/52 (data type: unsigned long)

Index	Description	Example / Range	adjustable	Firmware Version
1	Serial number	0x00000064 = 100	Ø	19.45
2	Main time	0x25766038 = 628514872 -> 01.12.2019 11:27:52 in seconds after the year 1.1.2000 0:0:0 $Range: [0 \ldots 2^{32}]$	✓	19.45
3	Correct the “Main” Time **(data type: signed long)**	0x00000025 = 37 max. ±172800 sec. ±2 days	✓	19.45
4	Timer “Measure” (next measuring date)	0x25766B51 = 628517713 -> 01.12.2019 12:15:13 Range: *Main Time* + 0 … 5184000 sec (60 days) Note: Values outside the range will be automatically fitted to the next nearest value	✓	19.45
5	Timer “Alarm”	0x254AD830 = 625662000 -> 29.10.2019 11:00:00 Range: *Main Time* + 0 … 5184000 sec (60 days) Note: Values outside the range will be automatically fitted to the next nearest value	✓	19.45

6	Timer “Info”	0x2577845F = 628589663 -> 02.12.2019 08:14:23 Range: *Main Time* + 0 … 5184000 sec (60 days) Note: Values outside the range will be automatically fitted to the next nearest value	✓	19.45
7	Timer “Event Measure” (next event measuring date)	**Not used** (reserved)	✓	19.45
8	Interval “Measure”	0x00000E10 = 3600 -> 1 h Range: 60 … 2592000 sec (30 days) Note: Values outside the range will be automatically fitted to the next nearest value	✓	19.45
9	Interval “Alarm”	0x00015180 = 86400 -> 1 day Range: 60 … 2592000 sec (30 days) Note: Values outside the range will be automatically fitted to the next nearest value	✓	19.45
10	Interval “Info”	0x00015180 = 86400 -> 1 day Range: 60 … 2592000 sec (30 days) Note: Values outside the range will be automatically fitted to the next nearest value	✓	19.45
11	Interval “Event Check”	**Not used** (reserved)	✓	19.45
12	Interval “Event Measure”	**Not used** (reserved)	✓	19.45

3.7 61 (0x3D) Float variable

Different numbers of float values can be transmitted. There are always 5 byte packets, where the first byte is the index number “Index” and the following four bytes define the float “FloatVal”

Byte – N°	1	2	3	4	5	6	7	8	9	10	11	…
Meaning	61 (0x3D)	Index (n)	$FloatVal_{Index}$ (n)				Index (y)	$FloatVal_{Index}$ (y)				…

Index: Index number
FloatVal: Float value of the corresponding index

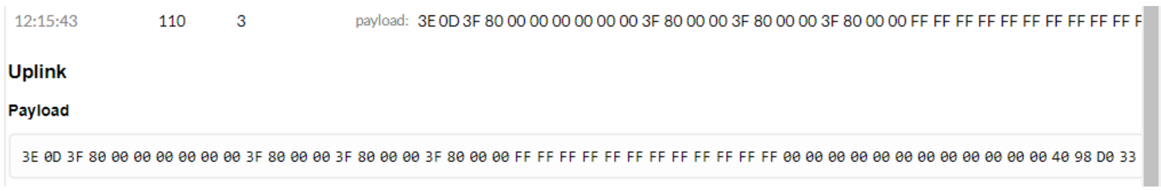
3.8 62 (0x3E) Float variable – stream

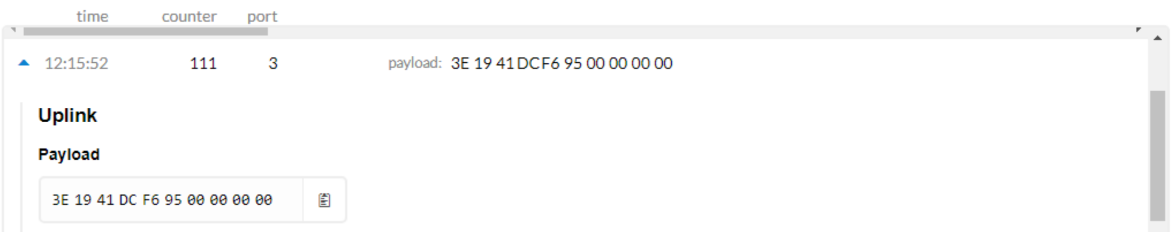
A larger number of float values can be transmitted, whereby only the index number “Index” of the first byte is specified, the following bytes are floats added in ascending order without gaps.

Byte – N°	1	2	3	4	5	6	7	8	9	10	…
Meaning	62 (0x3E)	Index (n)	$FloatVal_{Index}$ (n)				$FloatVal_{Index}$ (n+1)				…

Index: Index number (after each Byte increase index value by 1)
FloatVal: Float value of the corresponding index

Example (TTN):





Payload 109:
3E013F8000003F8000003F800000FFFFFFFFFFFFFFFFFFFFFFFF3F8000003F8000003F800000000000000044798CCD3F800000

Payload 110:
3E0D3F800000000000003F8000003F8000003F800000FFFFFFFFFFFFFFFFFFFFFFFF0000000000000000000000004098D033

Payload 111:
3E1941DCF69500000000

3.8.1 Definition of variable function 61/62 (data type: float)

Index	Description	Example / Range	adjustable	Firmware Version
1	Alarm On threshold	0x3F800000 = 1.0	✓	19.45
2	Alarm Off threshold	0x3F800000 = 1.0	✓	19.45
3	Alarm Delta threshold	0x3F800000 = 1.0	✓	19.45
4	Event On threshold	0xFFFFFFFF = Nan	✓	19.45
5	Event Off threshold	0xFFFFFFFF = Nan	✓	19.45
6	Event Delta threshold	0xFFFFFFFF = Nan	✓	19.45
7	Water Level Configuration Enable	0x3F800000 = 1.0	✓	19.45
8	Water Level Configuration Length (B)	0x3F800000 = 1.0	✓	19.45
9	Water Level Configuration Height (A)	0x3F800000 = 1.0	✓	19.45
10	Water Level Configuration Offset (f)	0x000000000 = 0.0	✓	19.45
11	Water Level Configuration Density	0x44798CCD = 998.2	✓	19.45
12	Water Level Configuration Width (b)	0x3F800000 = 1.0	✓	19.45
13	Water Level Configuration Angle (F)	0x3F800000 = 1.0	✓	19.45
14	Water Level Configuration Form factor (m)	0x000000000 = 0.0	✓	19.45
15	Water Level Configuration Minimum calculation Height (h)	0x3F800000 = 1.0	✓	19.45
16	Water Level Configuration Reserve	0x3F800000 = 1.0	✓	19.45
17	Water Level Configuration Reserve	0x3F800000 = 1.0	✓	19.45
18	Water Level Configuration Reserve	0xFFFFFFFF = Nan	✓	19.45
19	Water Level Configuration Reserve	0xFFFFFFFF = Nan	✓	19.45
20	Water Level Configuration Reserve	0xFFFFFFFF = Nan	✓	19.45
21	GPS coordinate as float (Longitude)	0x000000000 = 0.0	✓	19.45
22	GPS coordinate as float (Latitude)	0x000000000 = 0.0	✓	19.45
23	GPS coordinate as float (Altitude)	0x000000000 = 0.0	✓	19.45
24	Battery voltage in [V]	0x4098D033 = 4.775415	Ø	19.45
25	Rel. humidity in [%]	0x41DCF695 = 27.620401	Ø	19.45
26	Offset Barometer	0x000000000 = 0.0	Ø	19.45

3.9 71 (0x47) ASCII characters

Different numbers of ASCII characters can be transmitted. The length of the packets can vary depending on the ASCII characters, where the first byte is the start index number of the ASCII array and the second byte the index number “Index” which defines the chosen ASCII values followed by the ASCII content. The end is signalled with 0xFF.

Note: it is always taken care that the entire ASCII array fits into the payload, if not a new message is issued

Byte – N°	**1**	**2**	**3**	**4**	**…**	**…**	**n**	**n+1**	**n+2**	**n+3**	**…**	**…**	**n+z**
Meaning	71 (0x47)	start index ASCII Array	Index (x)	ASCII characters (x)			0xFF	start index ASCII Array	Index (y)	ASCII characters (x)			0xFF

Array $_{Index}$: start index number of the ASCII array ()
Index: Index number
ASCII characters: Float value of the corresponding index

3.10 72 (0x48) ASCII characters – stream

With function 72 the ASCII values are sent in a continuous stream. The index only appears at the beginning and is then automatically increased after an end character.

Note: it is always taken care that the entire ASCII array fits into the payload, if not a new message is issued

Byte – N°	**1**	**2**	**3**	**4**	**…**	**…**	**n**	**n+1**	**…**	**n+z**
Meaning	71 (0x47)	start index ASCII Array	Index (x)	ASCII characters (x)			0xFF	ASCII characters (x)		0xFF

Array $_{Index}$: start index number of the ASCII array ()
Index: Index number
ASCII characters: Float value of the corresponding index

Example (TTN):

	time	counter	port	
▲	14:47:48	124	5	payload: 48 00 07 30 46 36 39 32 46 45 31 43 45 44 46 37 37 42 46 32 38 38 37 33 39 44 39 36 45 44 45 35
▲	14:47:40	123	5	payload: 48 00 06 32 46 41 38 30 32 43 32 43 39 44 35 46 46 41 41 46 35 35 44 45 35 35 38 34 30 34 45 36
▲	14:47:31	122	5	payload: 48 00 04 33 31 43 34 39 34 43 37 44 46 44 38 37 46 30 33 33 35 35 46 44 45 30 41 37 31 45 38 42
▲	14:47:22	121	5	payload: 48 00 01 31 2E 30 2E 30 32 FF 30 30 39 44 36 42 30 30 30 30 43 35 44 33 38 32 FF 37 30 42 33 44
▼	14:47:18		3	payload: 5A 09 00 00

Payload 121:
480001312E302E3032FF30303944364230303030433544333832FF37304233443537454430303234364134FF

Payload 122:
4800043331433439344337444644383746303333353546444530413731453842353732FF3236303132314336FF

Payload 123:
4800063246413830324332433944354646414146353544453535383430344536363230FF

Payload 124:
4800073046363932464531434544463737424632383837333944393645444535344137FF

3.10.1 Definition of variable function 71/72 (data type: ASCII characters)

Index	Description	Example / Range	adjustable	Firmware Version
1	Firmware Version (short)	31 2E 30 2E 30 32 FF -> 1.0.02	Ø	19.45
2	Device EUI	30 30 39 44 36 42 30 30 30 30 43 35 44 33 38 32 FF -> 009D6B0000C5D382 Note: 16 ASCII characters	Ø	19.45
3	Application EUI*	37 30 42 33 44 35 37 45 44 30 30 32 34 36 41 34 FF -> 70B3D57ED00246A4 Note: 16 ASCII characters	✓	19.45
4	Application Key*	33 31 43 34 39 34 43 37 44 46 44 38 37 46 30 33 33 35 35 46 44 45 30 41 37 31 45 38 42 35 37 32 FF -> 31C494C7DFD87F03355FDE0A71E8B572 Note: 32 ASCII characters	✓	19.45
5	Device Address*	32 36 30 31 32 31 43 36 FF -> 260121C6 Note: 8 ASCII characters	✓	19.45
6	Network Session Key*	32 46 41 38 30 32 43 32 43 39 44 35 46 46 41 41 46 35 35 44 45 35 35 38 34 30 34 45 36 36 32 30 FF -> 2FA802C2C9D5FFAAF55DE558404E6620 Note: 32 ASCII characters	✓	19.45
7	App Session Key*	30 46 36 39 32 46 45 31 43 45 44 46 37 37 42 46 32 38 38 37 33 39 44 39 36 45 44 45 35 34 41 37 FF -> 0F692FE1CEDF77BF288739D96EDE54A7 Note: 32 ASCII characters	✓	19.45

* This values will not be changed until the user sends the confirmation (function 90 / Index 11 (Accept the configuration)).

Note: If an Index has no characters (empty) than only 0xFF will be appended

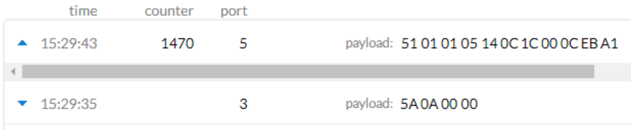
3.11 81 (0x51) KELLER Sensor information

With function 81 the information of the KELLER pressure sensor can be extracted. The content varies depending on the variant of the sensor (RS485 or I^{2}C).

Byte – N°	1	2	3	4	…	…
Meaning	81 (0x51)	Sensor number	Sensor type	Sensor data		

Sensor number: Sensor count
Sensor type: Sensor type (0 = Unknown / 1 = RS485 / 2 = I^{2}C)
ASCII characters: Float value of the corresponding index

Example (TTN):



Payload:
51010105140C1C000CEBA1

3.11.1 Sensor type 1 / RS485

Description	Example / Range	**Type**	Firmware Version
Sensor Class	0x05 = 5	unsigned char	19.45
Sensor Group	0x14 = 20	unsigned char	19.45
SW Version Year	0x0C = 12	unsigned char	19.45
SW Version Week	0x1C = 28	unsigned char	19.45
Serial number	0x000CEBA1 = 846753	unsigned long	19.45

3.11.2 Sensor type 2 / I^{2}C

<table>
<tr><th>Description</th><th>Example / Range</th><th>Type</th><th>Firmware Version</th></tr>
<tr><td>Unique ID</td><td>0x04070144</td><td>unsigned long</td><td>19.45</td></tr>
<tr><td>Scaling 0</td><td>0x1574

200b00010|1010|11101|00:
2|10|29|0
=> Date: 29.10.2012,
Mode: PR</td><td>unsigned int</td><td>19.45</td></tr>
<tr><td>Pmin Val</td><td>0x00000000 = 0 bar</td><td>unsigned long</td><td>19.45</td></tr>
<tr><td>Pmax Val</td><td>0x41F00000 = 30 bar</td><td>unsigned long</td><td>19.45</td></tr>
</table>

3.12 90 (0x5A) Command / Configuration

The user has the possibility to query different parameters using the function 90. Only one command can be transmitted. As parameter (Para) a maximum of 2 bytes can be transferred with the command. Start value must be less than or equal to Stop value.

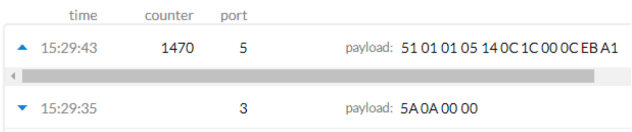
Byte – N°	**1**	**2**	**2**	**3**
Meaning	90 (0x5A)	Index	Para 1 (unsigned char)	Para 2 (unsigned char)

Index: define the function to be used
Para 1: defines the start index of the response message of the selected function
Para 2: defines the stop index of the response message of the selected function

Note: Para 1 and Para 2 correspond to the Index of the corresponding function (see function table). Is Para 1 and Para 2 equal to 0, then all variables are transferred.

Note: Only downlinks with port number 3 will be accepted. The response to a request is sent to port 5

Example (TTN):



3.12.1 Definition of commando function 90 (data type: unsigned char)

Index	Description	Para 1	**Para 2**	Firmware Version
1	Request measured values	0	0	19.45
2	Request small configuration	0	0	19.45
3	Request 1 Byte stream (Function code 32)	1	22	19.45
4	Reservec			
5	Request 4 Byte stream (Function code 52)	1	12	19.45
6	Request Float stream (Function code 62)	1	26	19.45
7	Request “Info” message (Function code 12)	0	0	19.45
8	Request big configuration	0	0	19.45
9	Request ASCII stream (Function code 72)	1	7	19.45
10	Request KELLER Sensor information (Function code 81)	1	5	19.45
11*	Accepting the configuration	0	0	19.45

* This Index is only needed to reconfigure the LoRa Keys and Settings. Here you have to know exactly what you’re doing.

4 DEVICE TYPE OVERVIEW

Type	Description	Channels
0	1x KELLER sensor with P_D (P_1-P_2) Interface: RS485(Bus address: 250) / I^2C (address: 0x40)	1: P_D (P_1-P_2) [bar] 2: P_1 [bar] 3: P_2 [bar] 4: T [°C] 5: TOB_1 [°C] 6: TOB_2 [°C]
1	1x KELLER sensor with P_D (P_1-P_2) Interface: RS485(Bus address: 250) / I^2C (address: 0x40)	1: P_D (P_1-P_2) [bar] 2: P_1 [bar] 3: P_2 [bar] 4: T [°C] 5: TOB_1 [°C] 6: TOB_2 [°C]
2	1x KELLER sensor with P_D (P_1-P_2) Interface: RS485(Bus address: 250) / I^2C (address: 0x40)	1: P_D (P_1-P_2) [bar] 2: P_1 [bar] 3: P_2 [bar] 4: T [°C] 5: TOB_1 [°C] 6: TOB_2 [°C] 7: P_{BARO} [bar] 8: T_{BARO} [°C]
3	1x KELLER sensor with P_D (P_1-P_{BARO}) Interface: RS485(Bus address: 250) / I^2C (address: 0x40)	1: P_D (P_1-P_{BARO}) [bar] 2: P_1 [bar] 3: P_2 [bar] 4: T [°C] 5: TOB_1 [°C] 6: TOB_2 [°C] 7: P_{BARO} [bar] 8: T_{BARO} [°C]
4	1x KELLER sensor with P_D (P_1-P_2) Interface: RS485(Bus address: 250) / I^2C (address: 0x40)	1: P_D (P_1-P_2) [bar] 2: P_1 [bar] 3: P_2 [bar] 4: T [°C] 5: TOB_1 [°C] 6: TOB_2 [°C] 7: P_{BARO} [bar] 8: T_{BARO} [°C] 9: - 10: -
5	1x KELLER sensor with P_D (P_1-P_{BARO}) Interface: RS485(Bus address: 250) / I^2C (address: 0x40)	1: P_D (P_1-P_{BARO}) [bar] 2: P_1 [bar] 3: P_2 [bar] 4: T [°C] 5: TOB_1 [°C] 6: TOB_2 [°C] 7: P_{BARO} [bar] 8: T_{BARO} [°C] 9: - 10: -
6	Up to 5 KELLER Sensor with 5x (P_1) Interface: RS485(Bus address: 1,2,3,4,5) /	1: P_D (P_1-P_2) (1) [bar] 2: P_1 (1) [bar] 3: P_2 (1) [bar] 4: T (1) [°C] 5: TOB_1 (1) [°C]

	I^2C (address: 0x40, 0x41, 0x43, 0x47, 0x4F)	6: TOB_2 (1) [°C] 7: P_{BARO} [bar] 8: T_{BARO} [°C] 9: - 10: - 11: P_1 (2) [bar] 12: P_1 (3) [bar] 13: P_1 (4) [bar] 14: P_1 (5) [bar] 15: -
7	1x SDI-12 Sensor Interface: SDI-12(Bus address: 0) /	1: not used 2: P_{BARO} [bar] 3: T_{BARO} [°C] 4: - 5: - 6: - 7: - 8: - 9: - 10: - 11: - 12: - 13: - 14: - 15: -
8	Up to 5 KELLER Sensor with 5x (P_1+TOB_1) Interface: RS485(Bus address: 1,2,3,4,5) / I^2C (address: 0x40, 0x41, 0x43, 0x47, 0x4F)	1: P_1 (1) [bar] 2: TOB_1 (1) [°C] 3: P_1 (2) [bar] 4: TOB_1 (2) [°C] 5: P_1 (3) [bar] 6: TOB_1 (3) [°C] 7: P_1 (4) [bar] 8: TOB_1 (4) [°C] 9: P_1 (5) [bar] 10: TOB_1 (5) [°C] 11: - 12: - 13: P_{BARO} [bar] 14: T_{BARO} [°C] 15: -
9	1x KELLER sensor for CTD with P_D (P_1-P_2) Interface: RS485(Bus address: 250) / I^2C (address: 0x40)	1: P_D (P_1-P_2) [bar] 2: P_1 [bar] 3: P_2 [bar] 4: T (Conductivity) [°C] 5: TOB_1 [°C] 6: TOB_2 [°C] 7: P_{BARO} [bar] 8: T_{BARO} [°C] 9: - 10: - 11: Conductivity T_C [$\frac{mS}{cm^2}$] 12: Conductivity raw [$\frac{mS}{cm^2}$]
10	1x KELLER sensor for CTD with P_D (P_1-P_{BARO}) Interface: RS485(Bus address: 250) / I^2C (address: 0x40)	1: P_D (P_1-P_{BARO}) [bar] 2: P_1 [bar] 3: P_2 [bar] 4: T (Conductivity) [°C]

		5: TOB_1 [°C] 6: TOB_2 [°C] 7: P_{BARO} [bar] 8: T_{BARO} [°C] 9: - 10: - 11: Conductivity T_C [$\frac{mS}{cm^2}$] 12: Conductivity raw [$\frac{mS}{cm^2}$]
11	Up to 3 KELLER sensor for CTD Interface: RS485(Bus address: 1,2,3,) / I^2C (address: 0x40, 0x41, 0x43)	1: P_1 (1) [bar] 2: TOB_1 (1)[°C] 3: Conductivity T_C (1) [$\frac{mS}{cm^2}$] 4: T (Conductivity) [°C] (1) 5: P_1 (2) [bar] 6: TOB_1 (2)[°C] 7: Conductivity T_C (2) [$\frac{mS}{cm^2}$] 8: T (Conductivity) [°C] (2) 9: P_1 (3) [bar] 10: TOB_1 (3)[°C] 11: Conductivity T_C (3) [$\frac{mS}{cm^2}$] 12: T (Conductivity) [°C] (3) 13: P_{BARO} [bar] 14: T_{BARO} [°C] 15: -
12	1x KELLER sensor with P_D (P_1-P_{BARO}) Interface: RS485(Bus address: 250) / I^2C (address: 0x40)	1: P_D (P_1-P_{BARO}) [bar] 2: P_1 [bar] 3: P_2 [bar] 4: T (Conductivity) [°C] 5: TOB_1 [°C] 6: TOB_2 [°C] 7: P_{BARO} [bar] 8: T_{BARO} [°C] 9: - 10: - 11: - 12: - 13: - 14: - 15: -
13	Up to 2 KELLER Sensor with 5x (P_1 + P_2 + TOB_1 + TOB_2) Interface: RS485(Bus address: 1,2) / I^2C (address: 0x40, 0x41)	1: P_1 (1) [bar] 2: P_2 (1) [bar] 3: TOB_1 (1) [°C] 4: TOB_2 (1) [°C] 5: P_1 (2) [bar] 6: P_2 (2) [bar] 7: TOB_1 (2) [°C] 8: TOB_2 (2) [°C] 9: P_{BARO} [bar] 10: T_{BARO} [°C] 11: - 12: - 13: -

5 REVISION HISTORY

Version	Date	Author	Description
12/2019	03.12.2019	Pascal Schlegel (SPa)	Create document
02/2020	13.02.2020	Pascal Schlegel (SPa)	Update description in F51/52 Index 8-10