LoRa radio packet definitions

version 1.22

name	LoRa-radio-packet-definitions.docx			
project				
content	Definition of radio packets as sent by ZENNER LoRa devices			
version	**state**	**Changes**	**author**	**date**
1.0	Release	Initial version	Rha	2016-08-28
1.1	Release	Added SP4, modified SP3 packet	Rha	2016-08-31
1.2	Release	Added packets for two channels; SP5	Rha	2016-09-01
1.3	Release	Added AP2; fixed packet type error	Rha	2016-09-02
1.4	Release	Added new status codes	Rha	2016-10-03
1.5	Release	New subtype 0x03	Rha	2016-10-07
1.6	Release	Added SP0 packet Clarifications regarding date and time	Rha	2017-01-29

		stamp Clarifications regarding status code		
1.7	Release	Clarification of coding unknown values (section 1.2)	Rha	2017-02-16
1.8	Release	Different usage of subtypes in SP9 packets Added SP9.2 packet for static device information	Rha	2017-02-21
1.9	Release	Correct table for AP2 packet content New status code 0x0C battery EOL Correction of sample in section 4 (CP packets)	Rha	2017-03-2
1.10	Release	Overview over AP packets Added new packet AP2.1	Rha	2017-10-23
1.11	Release	Moved AP2.1 to AP2.0 (replacing old AP2.0 which was never used)	Rha	2017-10-24
1.12	Release	Correction of error in table Status code definitions in section 3.2.1	Rha	2018-01-12
1.13	Release	New packet SP9 subtype 0x03	Rha	2018-04-12
1.14	Release	Clarification of table “status codes” (3.2.1) regarding battery low, battery warning and battery end of life	Rha	2018-08-06
1.15	Release	New packet SP12 for hourly values	Rha	2018-10-15
1.16	Release	New subtype 0x04 for SP1-SP8, explanation of time shift via LoRa	Rha	2019-04-08

1.17	Release	SP 9.3: now using 2 reserved bytes for VIFE	Rha	2019-09-20
1.18	Release	Correction of error in SP9.3 in 1.17	Rha	2019-09-20
1.19	Release	Correction of error in description of SP12	Rha	2019-10-28
1.20	Release	Modification of status code 0x01	Rha	2019-11-05
1.21	Release	Clarification of SP12 fields	Rha	2019-11-22
1.22	Release	New section “firmware version coding”	Rha	2019-12-19

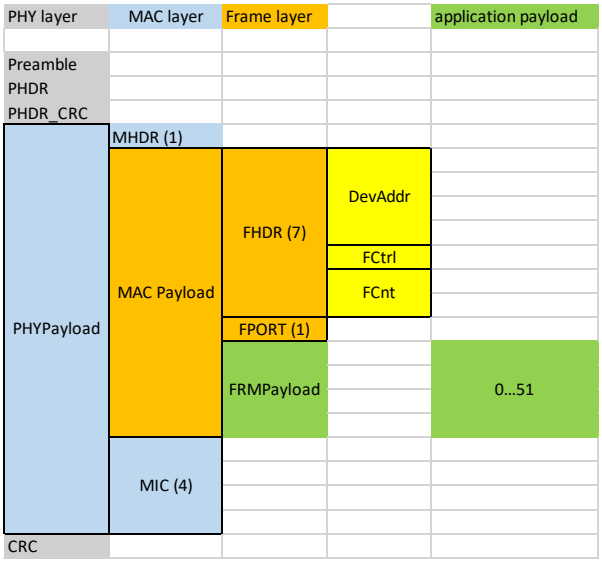
Contents

1 Introduction and definitions

This documents specifies the structure of the radio packets that a sent resp. received and understood by the devices of ZENNER that are using LoRa-WAN radio communication.

It defines the packet structure and some general rules. For a complete definition of all details the device specific definitions as given in the requirement specification resp. the userguide will be needed.

The packets described in the following (SP, AP, CP/CR) describe only the inner payload “FRMPayload” of a LoRa-WAN telegram as indicated in the following diagram (numbers in brackets denote the number of bytes, not the value of the field)



1.1 Definitions of used terms

1. “daily value” means the value of a meter/counter at the end of the previous calendar day
2. “monthly value” means the value of a meter/counter at the end of the previous calendar month
3. “half-monthly value” means the value of the end of the 15th day of a calendar month
4. “key date” means the day when the “key value” is recorded. Please note that in case of the “key date” the value of a meter/counter is recorded at the beginning of this calendar day, not at its end!

1.2 Data coding

1. Values that are not known or for other reasons not available to a device will be marked as 0xFF in all bytes of the value field
2. Byte order:
   If not specified differently, byte order will be MSB first.

2 Synchronous packets (SP)

Synchronous packets are typically sent based on a schedule, as opposed to AP packets which are sent only in case of an event

The following table gives an informative overview of all SP packet types. For implementation, refer to the detailed descriptions in the following sections.

packet type	packet name	subtype	short description	remark
0x00	SP0		current value	
0x01	SP1		day value, single channel	
0x02	SP2		monthly value, single channel	
0x03	SP3		monthly and half-monthly value, single channel	
0x04	SP4		key date and value at key date, single channel	
0x05	SP5		day value, two channels	two-channel version of SP1
0x06	SP6		monthly value, two channels	two-channel version of SP2
0x07	SP7		monthly and half-monthly value, two channels	two-channel version of SP3
0x08	SP8		key date and value at key date, two channels	two-channel version of SP4
0x09	SP9	0x00	current date and time	
		0x01	current date and time & status summary	
		0x02	static device information	
		0x03	information about channels	
0x0C	SP12		hourly values, one packet per channel	

SP packet overview

2.1 SP0

SP0			current value		
	no. of Bytes		**content**		**remark**
	0,5		packet type	0x00	
	0,5		packet subtype		
	4		current value		data format depending on subtype
	12		debug data		optional, data format depending on device type

SP0 packet

The SP0 packet format follows the structure in the table above. The interpretation of the optional 12 Bytes of debug data is device type specific.

The device type can either be deducted from the metadata of each device in the join server. Alternatively, it can be deducted from the DevEUI which follows a numbering scheme that allows deducting the basic device type.

The device type specific definitions are given in the requirement specification resp. the userguide of the device.

2.2 SP1/SP5

SP1			day value		
	no. of Bytes		**content**		**remark**
	0,5		packet type	0x01	
	0,5		packet subtype		
	4		day value		data format depending on subtype
overhead	13				
Total Bytes	**18**				
Time on Air (SF12)	1155 ms				

SP1 packet (one channel)

SP5	two channels		day value		
	no. of Bytes		**content**		**remark**
	0,5		packet type	0x05	
	0,5		packet subtype		
	4		day value channel 0		data format depending on subtype
	4		day value channel 1		data format depending on subtype
	2		status summary		
overhead	13				
Total Bytes	**24**				
Time on Air (SF12)	1318 ms				

SP5 packet (two channels)

SP1 and SP5 packets encode one or two day values

The timestamp of the value is not encoded in the packet and must be reconstructed from the reception time. SP1 and SP5 packets are only sent during the day after the day value was calculated (please refer also to the definitions in the previous section).

Example: The SP1 packet which contains the day value of January 20, 2017, will only be sent during January 21, 2017.

This means that the backend should verify that the internal clock of each device is still in sync with the real date and time. This verification can and should be done based on the monthly packets SP2 resp. SP6 which contain the current date (and in case of SP2 also time) of the device.

2.3 SP2/SP6

SP2			monthly value		
	no. of Bytes		**content**		**remark**
	0,5		packet type	0x02	
	0,5		packet subtype		
	4		time stamp		current date and time of the device
	4		monthly value		data format depending on subtype
	2		status summary		
overhead	13				
Total Bytes	**24**				
Time on Air (SF12)	1318 ms				

SP2 packet (one channel)

SP6	two channels		monthly value		
	no. of Bytes		**content**		**remark**
	0,5		packet type	0x06	
	0,5		packet subtype		
	2		date stamp		current date of device
	4		monthly value channel 0		data format depending on subtype
	4		monthly value channel 1		data format depending on subtype
overhead	13				
Total Bytes	**24**				
Time on Air (SF12)	1318 ms				

SP6 packet (two channels)

SP2 and SP6 packets encode one or two monthly values

The date resp. timestamp in the packet is the current date resp. time of the device, not the date/timestamp of the value. So some knowledge about the packet transmission schedule must be used on the receiver side: SP2 and SP6 packets always contain the value of the end of last month (related to the device clock). The date stamp in the packet allows determining which month that is. In addition it allows to check the device clock. If there is a significant difference this must be noticed and handled (annotation of the value, notification for monitoring purposes, re-synchronization of the device time via command packet).

2.4 SP3/SP7

SP3			monthly and half-monthly value		
	no. of Bytes		**content**		**remark**
	0,5		packet type	0x03	
	0,5		packet subtype		
	2		date stamp		current date of device
	4		monthly value		data format depending on subtype
	4		half-monthly value		data format depending on subtype
overhead	13				
Total Bytes	**24**				
Time on Air (SF12)	1318 ms				

SP3 packet (one channel)

SP7	two channels		monthly and half-monthly value		
	no. of Bytes		**content**		**remark**
	0,5		packet type	0x07	
	0,5		packet subtype		
	4		monthly value channel 0		data format depending on subtype
	4		half-monthly value channel 0		data format depending on subtype
	4		monthly value channel 1		
	4		half-monthly value channel 1		
overhead	13				
Total Bytes	**30**				
Time on Air (SF12)	1482 ms				

SP7 packet (two channels)

SP3/SP7 packets encode each one or two half-monthly and monthly values

As with SP2/SP6 packets, the date resp. timestamp in the packet is the current date resp. time of the device, not the date/timestamp of the value.

As can be seen, in a SP7 packet there is no date stamp, due to size constraints of the packet. Similar to SP1/SP5 packets, it must be assumed that SP7 packets are only sent in the second half of the calendar month. So if a SP7 packet arrives between the 15th of a month and the last day of a month, it can be assumed that the half-monthly values encoded in the packet belong to the same month. A plausibility check is possible by comparing the monthly value in the SP7 with the monthly value in a previously received SP6 packet (if any).

From this it becomes clear that it is important to check the device clock based on all received SP2/SP6 packets.

2.5 SP4/SP8

SP4			key date value		
	no. of Bytes		**content**		**remark**
	0,5		packet type	0x04	
	0,5		packet subtype		
	2		date		key date (note: not current date of the device)
	4		value at key date		data format depending on subtype
	2		status summary		
	2		reserved		
overhead	13				
Total Bytes	**24**				
Time on Air (SF12)	1318 ms				

SP4 packet (one channel)

SP8	two channels		key date value		
	no. of Bytes		**content**		**remark**
	0,5		packet type	0x08	
	0,5		packet subtype		
	2		date		key date (note: not current date of the device)
	4		value channel 0 at key date		data format depending on subtype
	4		value channel 1 at key date		data format depending on subtype
overhead	13				
Total Bytes	**24**				
Time on Air (SF12)	1318 ms				

SP8 packet (two channels)

SP4/SP8 packets encode one or two key date values

Note that in this case the date does encode the date of the value, which is not necessarily the current date of the device.

2.6 SP9 subtype 0x00

SP9	subytpe 0		time		
	no. of Bytes		**content**		**remark**
	0,5		packet type	0x09	
	0,5		packet subtype	0x00	
	4		time stamp		current date and time of the device
overhead	13				
Total Bytes	**18**				
Time on Air (SF12)	1155 ms				

SP9 subtype 0x00 packet

A SP9 subtype 0x00 packets encodes current date and time of the device.

Devices will send a SP9 subtype 0x00 packet to enable the receiver side to check the device clock and eventually resynchronize it using a CP packet.

2.7 SP9 subtype 0x01

SP9	subtype 0x01		time & status		
	no. of Bytes		**content**		**remark**
	0,5		packet type	0x09	
	0,5		packet subtype	0x01	
	4		time stamp		current date and time of the device
	2		status summary		
	4		reserved		
overhead	13				
Total Bytes	**24**				
Time on Air (SF12)	1155 ms				

SP9 subtype 0x01 packet

A SP9 subtype 0x01 packets encodes current date and time of the device and the status summary. It is mainly needed in scenarios where a status summary is not transmitted in the regular SP packets.

2.8 SP9 subtype 0x02

SP9	subtype 0x02		static device information		
	no. of Bytes		**content**		**remark**
	0,5		packet type	0x09	
	0,5		packet subtype	0x02	
	4		firmware version		including device type, according to ZENNNER device protocols "firmware version coding"
	3		LoRa_WAN-version		same value as in answer to FC0x35 EFC 0x01
	2		lora-command-version		same value as in answer to FC0x35 EFC 0x00
	1		Minol device type		as defined in "FunkIDnumberingscheme"
	4		meterID		assigned at production
	2		reserved		
overhead	13				
Total Bytes	**30**				
Time on Ai	1155 ms				

SP9 subtype 0x02 packet

A SP9 subtype 0x02 packet encodes static data of the device.

Typically, but depending on device implementation, it will be sent at startup of the device (after join accept) and from time to time.

2.8.1 Firmware version coding

Mask	**Meaning**	**Using**
0xff000000	MajorVersion	Incremented with major functionality changes
0x00ff0000	MinorVersion	Incremented by small enhancements
0x0000f000	Revision	Incremented with troubleshooting function without change
0x00000fff	DeviceIdentity	Fix for each device type or series. See userguide.
Example: 04013009		
Hex interpreted:		
0x04013009		
04_____ MajorVersion 4		
01_____ MinorVersion 1		
3_____ Revision 3		
009____ DeviceIdentity 9		
FirmwareVersion – xx xx xx xx - 4 byte firmware version. Low byte first.		

2.9 SP9 subtype 0x03

A SP9 subtype 0x03 packet encodes static data of attached devices in case of a split module.

Typically, but depending on device implementation, it will be sent some time after startup of the device (after join accept) and from time to time.

Please note that some devices like the heat meters are using the last 6 bytes differently, so for example the OBIS code must be evaluated to detect this.

SP9	subytpe 0x03		static device information		
	no. of Bytes		**content**		**remark**
	0,5		packet type	0x09	
	0,5		packet subtype	0x03	
	1		channel (0x00 denotes the device itself)		
	4		fabrication number ("serial number") of device at selected channel		
	2		manufacturer of device at selected channel		
	1		fabrication block ("generation") of device at selected channel		
	1		medium of device at selected channel		
	1		OBIS code of device at selected channel		
	3		VIF/VIFE of device at selected channel		
	3		reserved		
overhead	13				
Total Byte	**30**				
Time on A	1155 ms				

2.10 Subtype definitions

valid for: SP1 - SP8				note that for SP9 subtype is used differently
subtype		**definition**		**remarks**
0x0		value BCD coded		"integer" value, no colon. Physical unit must de determined out-of-band
0x01		value binary coded, unit scale		unit scale refers to HCA
0x02		value binary coded, product scale		product scale refers to HCA
0x03		higher 2 bytes: value[0], binary		only applicable where a single value is
0x04		bucket		used in T&H sensor
0x05				
0x06				
0x07				
0x08				

SP1-SP8 subtype definitions

2.11 Status summary definitions

The status summary bytes are defined differently from device to device. Therefore, they can be interpreted correctly only with the knowledge from which device type the telegram was sent.

The device type can either be deducted from the metadata of each device in the join server. Alternatively, it can be deducted from the DevEUI which follows a numbering scheme that allows deducting the basic device type.

The device specific definitions of the status summary bytes can be found in the requirement specification resp. the userguide of the device.

2.12 SP12

SP12	one channel		hourly values		
	no. of Bytes		**content**		**remark**
	0,5		packet type	0x0C	
	0,5		packet subtype		
	1		channel		
	1		number of first hour in this packet		0: first hourly value of the day (= value in the moment when the old day ends and the new day starts)
	4		first hour of this packet		data format depending on subtype
	4		second hour of this packet		data format depending on subtype
	4		third hour of this packet		data format depending on subtype
	2		reserved		
overhead	13				
Total Bytes	**30**				
Time on Air (SF12)	1482 ms				

SP12 packets encode 3 hourly values of the channel given in the “channel” field.

3 Asynchronous packets (AP)

AP packets are typically sent only in case of an event

3.1 Overview

packet type	packet name	subtype	short description	remark
0x0A	AP1.0		status code, status data	
0x0B	AP2.0	0	serial no. of device and attached devices	

3.2 AP1

AP1			state change		
	no. of Bytes		**content**		**remark**
	0,5		packet type	0x0A	
	0,5		packet subtype	0x00	
	1		status code		
	3		status data		
overhead	13				
Total Bytes	**18**				
Time on Air (SF12)	1155 ms				

AP1 packet

An AP1 packet is used to asynchronously notify about state changes

3.2.1 Status code definitions

status code	explanation	status data			remarks
		Byte 0	Byte 1	Byte 2	
0x01	tamper	bit 0: (0=started, 1=ended)	date of event		
0x02	removal		date of event		
0x03	leak	channel (0 or 1)	date of event		
0x04	reverse flow	channel (0 or 1)	date of event		
0x05	battery warning		value under load		value is coded binary, physical unit mV; sometimes also named "battery low"
0x06	oversized	channel (0 or 1)	date of event		
0x07	undersized	channel (0 or 1)	date of event		
0x08	error		device specific error code		definition in req.spec./userguide of device (device type deducted from metadata of numbering sc
0x09	1F mode				
0x0A	blockage	channel (0 or 1)	date of event		
0x0B	burst	channel (0 or 1)	date of event		
0c0C	battery EOL				calculated battery end-of-life is reached, value is coded as in status code 0x05
0x0D	reserved				
0x0E	reserved				
0x0F	reserved				
0x10	battery prewarning		value under load		also sometimes called "battery forewarning"
0x11	reserved				
0x12	reserved				
0x13	smoke chamber pollution forewarning				
0x14	smoke chamber pollution warning				
0x15	push button failure				
0x16	horn drive level				
0x17	reserved				
0x18	test alarm released				
0x19	smoke alarm released				
0x1A	ingress apertures obstruction detection				
0x1B	LED failure				
0x1C	object in the sorrounding area detected				
0x1D - 0xFF	reserved				

Status code and status data

3.3 AP2.0

AP2.0			configuration change subtype 0x00		
	no. of Bytes		**content**		**remark**
	0,5		packet type	0x0B	
	0,5		packet subtype	0x00	
	1		channel (0x00 denotes the device itself)		
	4		fabrication number ("serial number") of device at selected channel		
	2		manufacturer of device at selected channel		
	1		fabrication block ("generation") of device at selected channel		
	1		medium of device at selected channel		
overhead	13				
Total Bytes	**23**				
Time on Air (SF12)	1155 ms				

An AP2.0 packet is used to notify about configurations in case of “split” devices (devices with separately attached meters), but can also be used just to notify about the FullSerialNumber of the device itself.

4 Command packets (CP)

Command packets (CP) are packets that are sent from the application via the LNS and a LoRa gateway to the device. In terms of LoRa-WAN specification, they are named “downlink messages”.

The structure is defined as follows in LoRa-WAN:

Downlink PHY:

Preamble	PHDR	PHDR_CRC	PHYPayload

LoRa-WAN does not require a CRC over the PHYPayload.

The ZENNER LoRa command packets are the content of the PHYPayload in downlink direction. The following rules apply for ZENNER LoRa command packets (CP) in downlink direction:

- All command packets (CP) follow this structure:

1 or 2 Bytes	0-n Bytes	2 Bytes (only if write)
FC; EFC (if any)	argument(s)	CRC

  This means they encapsulate the FC (function code) and EFC (extended function code) codes that are also used in IR communication.
- Command packets that implement a read request (backend intends to read one or more values from a device) are used without CRC
- Command packets that implement a write request (backend intends to change one or more values on the device) use a 2 Bytes CRC at the end of the command packet
- The device will silently discard a Command Packet with an invalid CRC (it may increment an internal counter, though)
- The response follows this structure:

	0-n Bytes
0xFE;FC; EFC (if any)	response data

Which FC and EFC in a LoRa Command Packet are supported is specified in the requirement specification and/or userguide of the device.

Example:

example:							
command				**response**			
set key date to Aug 01	0x88 0x08 0x01 [CRC]			0xFE 0x88	if CRC check failed: silent discard		
read key date	0x88			0xFE 0x88 0x08 0x01			

Example “time shift command”:

The command “TimeShift” is defined as follows:

4.1.22 TimeShift (FC=8E)

←←←←

Master to slave:

Offset	Meaning	Bytes [hex]	Description
11	FC	8E	Function code
12,13	Time shift	?? ?? ??	See explanation below

Highest bit of Byte 13 gives the direction of the timeshift.

Direction Bit == 0: Device shifts its local time by adding the value of Byte 12 and Byte 13 (except highest bit) in seconds to its current local time (device time moves to the future)

Direction Bit == 1: Device shifts its local time by subtracting the value of Bytes 12 and Byte 13 (except highest bit) in seconds from its current local time (device time moves to the past)

Slave to master = ACK response frame. →→→→

A LoRa packet with the payload 8E1815A654 means:

- 8E is the command ID
- 0x1518 = 5400 sec., highest bit = 0 means device will add this to its local time
- 0x54A6 is CRC16, using polynome 0x8005 and start value 0xFFFF

The answer on this packet will just be FE8E as payload (acknowledgement of command reception).

A LoRa packet with the payload 8E1895A5D7 means:

- 8E is the command ID
- 0x9518 = 5400 sec., highest bit = 1 means device will subtract this from its local time
- 0xD7A5 is CRC16, using polynome 0x8005 and start value 0xFFFF

There are some important notes when using the time shift command:

- The command is not carried out by the device immediately, but can be delayed until 12:00 the next day. Before it is carried out, transmitted packets might still show the old device time.

- As acknowledgement can get lost in radio transmission or uplink transmission token might be used up, the backend should always wait until it receives a later packet (given the delay in the device: packet sent after the next day) with a time of the device before it resends a time shift command.
- The device does not implement any duplicate command detection.
- Note that shifts >10h are not possible in one command and much smaller time shifts are recommended.

5 Annex: external definitions

5.1 Date format

Date stamp coding according to EN13757-3:2013, Annex A, data type G:

Typ G: Zusammengesetzter Datentyp Compound CP16: Datum

Tabelle A.6 — Typ G: Datum (CP16)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8

Tag: UI5 [1 bis 5] <1 bis 31>
„0“: jeden Tag

Monat: UI4 [9 bis 12] <1 bis 12>
„15“: jeden Monat

Jahr: UI7 [6 bis 8, 13 bis 16] <0 bis 99>
„127“: jedes Jahr

Zur Kompatibilität mit alten Zählern mit umlaufender zweistelliger Datumsanzeige wird empfohlen, in jeder Master-Software die Jahre „00“ bis „80“ als die Jahre 2000 bis 2080 anzusehen. Ein Wert von FFh in beiden Bytes (d.h. FFh FFh) ist als ungültig zu interpretieren.

5.2 Date and time format

Date&Time stamp coding according to EN13757-3:2013, Annex A, data type F:

Typ F: Zusammengesetzter Datentyp Compound CP32: Datum und Uhrzeit

Tabelle A.5 — Typ F: Datum und Uhrzeit (CP32)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}

Min: UI6 [1 bis 6] <0 bis 59>; 63: jede Minute

Stunde: UI5 [9 bis 13] <0 bis 23>; 31: jede Stunde

Tag: UI5 [17 bis 21] <1 bis 31>; 0: jeden Tag

Monat: UI4 [25 bis 28] <1 bis 12>; 15: jeden Monat

Jahr: UI7 [22 bis 24 ; 29 bis 32] <0 bis 99>; 127: jedes Jahr

Hundertjahr-Format: UI2 [14 bis 15] <0 bis 3> dieses Jahr ist 1900 + 100 * Hundertjahr + Jahr

IV B1 [8] IV <0> = gültig; IV <1> = ungültig

SU B1 [16] IV <0> = Normalzeit; IV <1> = Sommerzeit

RES1 B1 [7] <0> reserviert für zukünftige Nutzung

Zur Kompatibilität mit alten Zählern mit umlaufender zweistelliger Datumsanzeige wird empfohlen, in jeder Master-Software die Jahre „00“ bis „80“ als die Jahre 2000 bis 2080 anzusehen.