LoRaWan sensor commands. Version 4.6

Revision 2.22

Contents.

Contents **1**
Commands description **2**
RF Module command list **3**
Detailed description of commands **4**
DATA_DAY 4
DATA_HOUR_DIF 4
LAST_EVENTS 4
DELTA_TIME 6
GET_CURRENT 6
TIME2000 6
NEW_STATUS 6
NEW_EVENT 7
DATA_DAY_MUL 8
DATA_HOUR_MUL 9
GET_CURRENT_MUL 10
ABS_DATA_DAY 11
Supported Downlink commands. Server-side request **12**
SET_TIME2000 12
SET_PARAMETERS 13
GET_PARAMETERS 16
GET_ARCHIVE_HOURS 17
GET_ARCHIVE_DAYS 17
GET_CURRENT 18
GET_ARCHIVE_EVENTS 18
CORRECT_TIME2000 19
GET_NEW_STATUS 19
SOFT_RESTART 21
GET_ARCHIVE_HOURS_MUL 21
GET_ARCHIVE_DAYS_MUL 22
CLEAR_PARAMETERS 23
Commands relevant for the 4PI module (4-input) **28**
Representing integers using the minimum bytes **29**

Commands description

The sensor sends and receives the messages in the format:

Command 1
Command 2
…
Command n
LRC

The LRC is calculated by performing an XOR operation on the content of the message with a start value 0x55.

Command's format.

1. Command with a one-byte header.

Description	7	6	5	4	3	2	1	0
Command's code. Command's data size.	CMD != 0			CMD_LENGTH				
Command's data.	CMD DATA							

2. Command with a two-bytes header.

Description	7	6	5	4	3	2	1	0
Command's code.	0			CMD				
Command's data size.	CMD LENGTH							
Command's data.	CMD DATA							

3. Command with a three-bytes header.

Description	7	6	5	4	3	2	1	0
	0			0x1f (EXTRA_CMD)				
Command's code.	CMD							
Command's data size.	CMD LENGTH							
Command's data.	CMD DATA							

RF Module command list

Uplink data from the rf module

CMD	Description
DATA_DAY = 0x20-0x3F	Pulse counter daily data on billing hour
DATA_HOUR_DIF = 0x40-0x5F	Pulse counter hourly data, data accumulated on an hourly basis and transmitted from jooby rf module on adjustable period
LAST_EVENTS = 0x60-0x7F	Last unread event
DELTA_TIME = 0x80-0x9F	Data on the time shift of the moment of frame transmission and the moment of hourly data recording
ABS_HOUR_DIFF = 0xA0-0xBF	Absolute hourly data, data accumulated on an hourly basis and transmitted from jooby rf module on adjustable period
ABS_DATA_DAY = 0xC0-0xDF	Absolute daily data on billing hour
GET_CURRENT=0x07	Current pulse counter value
TIME2000 = 0x09	Current rf module time (time 2000 format)
NEW_STATUS=0x14	RF Module status
NEW_EVENT=0x15	New event occurs
DATA_DAY_MUL=0x16	Pulse counter data of the multichannel sensor on billing hour.
DATA_HOUR_MUL=0x17	Pulse counter data of the multichannel sensor, accumulated on a half-hour basis.
GET_CURRENT_MUL=0x18	Current pulse counter value for multichannel devices
MTX_CMD = 0x1E	Read data from an electricity meter MTX

Supported **Downlink** commands. Server-side request.

CMD	Description
SET_TIME2000 = 0x02	Time correction command. 4bytes signed value in seconds Time2000 format
SET_PARAMETERS = 0x03	Set parameters command
GET_PARAMETERS = 0x04	Get parameters command
GET_ARCHIVE_HOURS = 0x05	Read pulse counter hourly archived data command
GET_ARCHIVE_DAYS = 0x06	Read pulse counter daily archived data
GET_CURRENT = 0x07	Read current pulse counter value
TIME2000 = 0x09	RF module current time
GET_ARCHIVE_EVENTS=0x0B	Read events archived data
CORRECT_TIME2000=0x0C	Command data consists of a sequence number and one byte signed correction value. Command accepted only if sequence number greater than value previously received by the module
GET_NEW_STATUS=0x14	Get rf module status
GET_CURRENT_MUL = 0x18	Request for reading current pulse for multichannel device
SOFT_RESTART = 0x19	Software reset
GET_ARCHIVE_HOURS_MUL=0x1A	Read pulse counter hours-based archived data of the multichannel sensor.
GET_ARCHIVE_DAYS_MUL = 0x1B	Read pulse counter archived daily data of the multichannel sensor on billing hour.
CLEAR_PARAMETERS = 0x1D	Reset to factory defaults settings
MTX_CMD = 0x1E	Read data from an electricity meter MTX

Detailed description of commands

DATA_DAY

DATA_DAY This command is generated by the sensor and sent within a specified period without a request from the server when it is configured to issue daily consumption data. The command gives the full data of the pulse counter for the specified day. The bit field Magnet=1 means a magnet influence on the specified day.

DATA_DAY = 0x20.							
7	6	5	4	3	2	1	0
0	0	1	CMD_LENGTH=6				
Year[7..0]							Month[3]
Month[2..0]			Date[4..0]				
Magnet	reserved	reserved	Hour[4..0]				
Counter[0][2]							
Counter[0][1]							
Counter[0][0]							

DATA_HOUR_DIF

DATA_HOUR_DIF. This command is generated by the sensor and sent within a specified period without a request from the server when it is configured to output hourly consumption data. The command gives the full value of the pulse counter for the specified hour and the hourly difference in readings, which is limited to 13 bits. The bit field Magnet=1 means that there was a magnet impact at the specified hour.

DATA_HOUR_DIF = 0x40.							
7	6	5	4	3	2	1	0
0	1	0	CMD_LENGTH				
Year[7..0]							Month[3]
Month[2..0]			Date[4..0]				
Magnet	reserved	reserved	Hour[4..0]				
Counter[0][2]							
Counter[0][1]							
Counter[0][0]							
…							
Magnet	reserved	reserved	Counter_dif[hours-1][12..8]				
Counter_dif[hours-1][7..0]							

LAST_EVENTS

LAST_EVENTS. This command will be added to all commands the module sends without a request. The **GET_NEW_STATUS** command will be an exception because the data is inside the command. The command transmits the sequence code of the last sensor event.

LAST_EVENTS = 0x60. Last unread event							
7	6	5	4	3	2	1	0
0	1	1	CMD_LENGTH				
EVENT_SEQUENCE							
EXTEND=1	STATUS[6..0]						
EXTEND=0	STATUS[13..7]						

STATUS bits for gas meters:

- 0 - BAT = «1» – battery voltage is below the set limit,
- 1 - MAGNET – «1» - there is an influence of a magnetic field (only for gas sensors),
- 2 - BUTTON – «0» - button is pressed "1" - the button is released (only for gas sensors, a pressed button indicates the removal of the sensor),
- 3 - DOWN – «1» - the sensor detected a loss of connection with the server,
- CHN0, CHN1 – «1» - connector is disconnected (ignore for gas sensors).
- 4..6 - RES - «0» - зарезервированы;
- 7 - EXTEND - extend bit.

STATUS bits 4-x channels radiomodule:

- 0 - BAT = «1» – battery voltage is below the set limit,
- 1,2 - Reserved «0».
- 3 - DOWN – «1» - the sensor detected a loss of connection with the server,
- 4 - CHN0, 5 - CHN1, 6 - CHN2 – «1» - connector is disconnected.
- 7 - EXTEND - extend bit.
- 8 - CHN3 – «1» - connector is disconnected
- 9..14 - «0» - Reserved
- 15- EXTEND - «0»

Format for MTX device:

There are two bytes of Event_Status in the command body after the EVENT_SEQUENCE byte for the MTX lorawan module.

LAST_EVENTS = 0x60. Last unread event							
7	6	5	4	3	2	1	0
0	1	1	CMD_LENGTH				
EVENT_SEQUENCE							
EXTEND=0	RES	CHN1	CHN0	DOWN	BUTTON	MAGNETE	BAT
Status_Event* (MTX)							
Status_Event_2* (MTX)							

STATUS bits MTX device:

- 0 - «1» - Case open;
- 1 - «1» - Detected electromagnetic field;
- 2 - «1» - Set parameters remotely
- 3 - «1» - Set parameters locally ;
- 4 - «1» - Restart software;
- 5 - «1» - Wrong password and blocking;
- 6 - «1» - Set time;
- 7 - «1» - Correction time;
- 8 - «1» - Fault meters;
- 9 - «1» - Case open: terminal area;
- 10 - «1» - Case open: communication area;
- 11 - «1» - Tariff plan changed;
- 11 - «1» - Get new tariff plan;
- 12..15 - «0» - Reserved.

DELTA_TIME

DELTA_TIME This command will be substituted before the **DATA_HOUR_DIF** command if the DELTA_TIME_EN parameter is set. Delta_Time[2] - is a 2-byte value of the time shift between the moment of data transmission and the moment of the last recording of hourly data.

Value in seconds. The most significant byte comes first. Possible value - 0 - 3599 seconds.

DELTA_TIME = 0x80.							
7	6	5	4	3	2	1	0
1	0	0	CMD_LENGTH=2				
Delta_Time[2]							

GET_CURRENT

GET_CURRENT. The current readings of the sensor pulse counter.

The sensor can periodically send this command without request if the type of reporting data (TYPE_PARAMETERS = 5) has been set to 2 - transmission of current data. In the body of the command, the current value of the pulse counter is transmitted.

GET_CURRENT = 0x07.							
7	6	5	4	3	2	1	0
0	0	0	CMD GET_CURRENT = 0x07				
CMD_LEGTH= 4							
Magnet	reserved	reserved	0				
Counter[2]							
Counter[1]							
Counter[0]							

TIME2000

TIME2000. Command transferred as uplink from the module indicates current module time. **TIME_SEQ** is the sequence number transmitted from the server by the last command to correct or set the time.

TIME2000 = 0x09							
7	6	5	4	3	2	1	0
0	0	0	CMD TIME2000 = 0x09				
CMD_LEGTH = 5							
TIME_SEQ							
TIME2000[3]							
TIME2000[2]							
TIME2000[1]							
TIME2000[0]							

NEW_STATUS

NEW_STATUS. The command to request status information from the sensor. New command version. The command is sent by the sensor without request once a day. LOW_CURRENT_VBATTERY - battery voltage with minimum load (SLEEP MODE), value in mV. HIGH_CURRENT_VBATTERY – battery voltage with load simulating

transmission mode, value in mV. INT_RESISTANT is the internal resistance of the battery in mΩ. If the voltage value is 4095 mV, then the value is unknown. If the value of the internal resistance is 65535 mOhm, then the resistance is unknown. TEMPERATURE is a signed number in degrees Celsius. REMINED_BATTERY_CAPACITY is the remaining battery capacity, where 254 is 100%. A value of 255 means that the remaining battery capacity is unknown. LAST_EVENT - last unread sensor event in 48 hours. This field will automatically reset after 48 hours. After reading the sensor status, this field is also reset.

NEW_STATUS = 0x14							
7	6	5	4	3	2	1	0
0	0	0	CMD GET_NEW_STATUS = 0x14				
CMD_LEGTH = 12							
SOFT_TYPE							
SOFT_VERSION							
HARD_TYPE							
HARD_VERSION							
LOW_CURRENT_VBATTERY[11..4]							
LOW_CURRENT_VBATTERY[3..0]				HIGH_CURRENT_VBATTERY[11..8]			
HIGH_CURRENT_VBATTERY[7..0]							
INT_RESISTANT[15..8]							
INT_RESISTANT[7..0]							
TEMPERATURE							
REMINED_BATTERY_CAPACITY							
LAST_EVENT							

NEW_EVENT

NEW_EVENT. This frame is formed immediately after the event occurs. To increase the delivery probability, a frame can be transmitted with an acknowledgement request (at the network layer).

Whether to transmit with a confirmation request or not is determined by the set parameters of the sensor operation.

NEW_EVENT = 0x15.							
7	6	5	4	3	2	1	0
0	0	0	CMD **NEW_EVENT** = 0x15				
CMD_LENGTH							
EVENTS							
EVENT_SEQUENCE							
PARAMETERS (TIME2000[4],VBAT[2] .. и т.д.)*							

*The format of the passed parameters depends on the event type. All possible event types and their corresponding parameters are listed in the table below.

MAGNET_ON=1, MAGNET_OFF=2, ACTIVATE=3, DEACTIVATE=4, CAN_OFF=6, INSERT=7, REMOVE=8, COUNTER_OVER=9, EV_OPTOLOW=15, EV_OPTOFLASH=16,EV_REJOJN=18
EVENT_SEQUENCE
TIME2000[4]
BATTERY_ALARM=5
EVENT_SEQUENCE
VBAT[2]
ACTIVATE_MTX=11

EVENT_SEQUENCE	
TIME2000[4]	
EXT_ADDR[8]	
CONNECT=12, DISCONNECT=13	
EVENT_SEQUENCE	
CHANNEL	
EXTEND=1	COUNTER[6..0]
EXTEND=1	COUNTER[13..7]
EXTEND=1	COUNTER[20..14]
EXTEND=0	COUNTER[27..21]
EV_ MTX=17	
EVENT_SEQUENCE	
STATUS_EVENT	
STATUS_EVENT2	

DATA_DAY_MUL

DATA_DAY_MUL. This command is generated by the multichannel sensor and sent within a specified period without a request from the server when configured to issue daily consumption data. The command gives the pulse counter's full value for the previous day's billing hour. Bit **EXTEND** in the fields **CHANNELS** and **COUNTER** is used for field extension. If **EXTEND** = «1», the next byte will be an extension of the field. If **EXTEND** = «0», it is the last byte of the field. **CHANNELS** is a bit field and describes the channels, which are the data sources for the command. If bits 0, 2 3 are set up, then data in the command will be in the following order - data from the 0 channel first, then data from the 2 channel and data from the 3 channel last. The **COUNTER** field is 32 bits in size. The **COUNTER** also contains the **EXTEND** bit. If **EXTEND** = «1», the next byte will contain the next high 7 bits of the pulse counter. If **EXTEND** = «0», it is the last byte of the pulse counter.

DATA_DAY_MUL = 0x16.							
7	6	5	4	3	2	1	0
0	0	0	CMD **DATA_DAY_MUL** = 0x16				
CMD_LENGTH							
Year[7..0]							Month[3]
Month[2..0]			Date[4..0]				
EXTEND=1	CHANNELS[6..0]						
EXTEND=0	CHANNELS[13..7]						
EXTEND=1	COUNTER_1[6..0]						
EXTEND=1	COUNTER_1[13..7]						
EXTEND=1	COUNTER_1[20..14]						
EXTEND=0	COUNTER_1[27..21]						
EXTEND=1	COUNTER_2[6..0]						
EXTEND=1	COUNTER_2[13..7]						
EXTEND=0	COUNTER_2[20..14]						
…							
EXTEND=0	COUNTER_N[6..0]						

DATA_HOUR_MUL

DATA_HOUR_MUL. This command is generated by the multichannel sensor and sent within a specified period without a request from the server when configured to issue half-hour consumption data. The command contains the total value of the pulse counter for the specified hour and hourly difference, which is limited to 31-bit size. The **Hours**[2..0] field contains the count of hours in the command. «0» means the first hour (+1 to the value **Hours**[2..0]). Bit **EXTEND** in the fields **CHANNELS** and **COUNTER_X_DIF** is used for field extension. If **EXTEND** = «1», the next byte will be an extension of the field. If **EXTEND** = «0», it is the last byte of the field. **CHANNELS** is a bit field and describes the channels, which are the data sources for the command. If bits 0, 2 3 are set up, then data in the command will be in the following order - data from the 0 channel first, then data from the 2 channel and data from the 3 channel last. **COUNTER_X** field contains the total value of the pulse counter of the **Hour**[4..0] for the specified day. **COUNTER_X** also includes **EXTEND** bit. **COUNTER_X_DIF_N** has the hourly difference and **EXTEND** bit.

DATA_HOUR_MUL = 0x17.							
7	6	5	4	3	2	1	0
0	0	0	CMD **DATA_HOUR_MUL** = 0x17				
CMD_LENGTH							
Year[7..0]							Month[3]
Month[2..0]			Date[4..0]				
Hours[2..0]			Hour[4..0]				
EXTEND=0	CHANNELS[6..0]						
EXTEND=1	COUNTER_1[6..0]						
EXTEND=1	COUNTER_1[13..7]						
EXTEND=0	COUNTER_1[20..14]						
EXTEND=1	COUNTER_1_DIF_1[6..0]						
EXTEND=0	COUNTER_1_DIF_1[13..7]						
EXTEND=1	COUNTER_1_DIF_2[6..0]						
EXTEND=1	COUNTER_1_DIF_2[13..7]						
EXTEND=0	COUNTER_1_DIF_2[20..14]						
EXTEND=1	COUNTER_2[6..0]						
EXTEND=1	COUNTER_2[13..7]						
EXTEND=1	COUNTER_2[20..14]						
EXTEND=0	COUNTER_2[27..21]						
EXTEND=0	COUNTER_2_DIF_1[6..0]						
EXTEND=1	COUNTER_2_DIF_2[6..0]						
EXTEND=0	COUNTER_1_DIF_2[13..7]						
…							
EXTEND=1	COUNTER_N[6..0]						
EXTEND=0	COUNTER_N[13..7]						
EXTEND=1	COUNTER_N_DIF_1[6..0]						
EXTEND=1	COUNTER_N_DIF_1[13..7]						
EXTEND=1	COUNTER_N_DIF_1[20..14]						
EXTEND=0	COUNTER_N_DIF_1[27..21]						
EXTEND=1	COUNTER_2_DIF_2[6..0]						
EXTEND=0	COUNTER_N_DIF_2[13..7]						

GET_CURRENT_MUL

GET_CURRENT_MUL. This command is sent in response to the reading command of the current pulse for multichannel devices. This command can be sent periodically if parameter 5 is set to value 2.

In the command, the current value of the pulse counter is transmitted. The EXTEND bit in the CHANNELS and COUNTER field is used to extend the field. If EXTEND = "1", then this means that the next byte will contain the continuation of the data field. If EXTEND = "0", then this means that this is the last byte of this data field.

The CHANNELS field is a bit field that describes the channel's data from sensor per channel. If bits 0,1,2 and 3 are set in the CHANNELS field, this means that the current data for channels present in the message. The data for 0th channel comes first, then the 2nd, and finally from the 3rd channel.

The pulse counter is 32 bits unsigned integer . The pulse counter is transmitted using the EXTEND bit. The first byte of the data field contains the bit of the connection indicator of the channel connector and the 6 least significant bits of the counter.

GET_CURRENT_MUL = 0x18.							
7	6	5	4	3	2	1	0
0	0	0	CMD **GET_CURRENT_MUL** = 0x18				
CMD_LENGTH							
EXTEND=1	CHANNELS[6..0]						
EXTEND=0	CHANNELS[13..7]						
EXTEND=1	Counter1[6..0]						
EXTEND=1	Counter1[13..7]						
EXTEND=1	Counter1[20..14]						
EXTEND=1	Counter1[27..21]						
EXTEND=0	Counter1[31..28]						
EXTEND=1	Counter2[6..0]						
EXTEND=1	Counter2[13..7]						
EXTEND=0	Counter2[20..14]						
…							
EXTEND=1	CounterN[6..0]						
EXTEND=1	CounterN[13..7]						
EXTEND=1	CounterN[20..14]						
EXTEND=1	CounterN[27..21]						
EXTEND=0	CounterN[31..28]						

Example:

18 06 0F 83 01 08 0A 0C

18 06 - Command GET_CURRENT_MUL, len = 6

0F - EXTEND = 0 , CHANNELS[6..0] = 0x0F - counter for channel 1,2,3,4 present

83 01 - Counter channel 1 = 0x83 = (0x01<<7) + (0x83&0x7F) <-exclude EXTEND

08 - Counter channel 2 = 0x08

0A - Counter channel 3 = 0x0A

0C - Counter channel 4 = 0x0C

ABS_HOUR_DIFF

ABS_HOUR_DIFF. This command is generated by the sensor and sent with a specified period without a request from the server. The command sends absolute data that correspond to the impulse coefficient. This command will be issued instead of the DATA_HOUR_DIF command in the event that metering devices were transferred to the sensor at the moment of activation (installation) of the sensor. The IPK field is an impulse coefficient that perceives the consumed resource in 1 impulse. The METER field is the corrected value of the pulse counter, taking into account the initial data of the meter and sensor.

ABS_HOUR_DIF = 0xA0.							
7	6	5	4	3	2	1	0
1	0	1	CMD_LENGTH				
IPK							
Year[7..0]							Month[3]
Month[2..0]			Date[4..0]				
Magnet	reserved	reserved	Hour[4..0]				
METER[0][2]							
METER[0][1]							
METER[0][0]							
…							
Magnet	reserved	reserved	Meter_dif[hours-1][12..8]				
Meter_dif[hours-1][7..0]							

ABS_DATA_DAY

ABS_DATA_DAY. This command is generated by the sensor and sent with a specified period without a request from the server when it is configured to send daily consumption data. The command send the full data of the meter for the hour on the specified day. The bit field Magnet=1 means that there was a magnet impact on the specified day. The command send the daily consumption data of the metering device for the accounting hour. This command will be issued instead of the DATA_DAY command in the event that meter readings were transferred to the sensor at the time of activation (installation) of the sensor. The IPK field is the device's impulse coefficient, which determines whether the consumed resource corresponds to 1 impulse.The METER field is the corrected value of the pulse counter, taking into account the initial data of the meter and sensor.

ABS_DATA_DAY = 0xC0.							
7	6	5	4	3	2	1	0
1	1	0	CMD_LENGTH=6				
IPK							
Year[7..0]							Month[3]
Month[2..0]			Date[4..0]				
Magnet	reserved	reserved	Hour[4..0]				
METER[0][2]							
METER[0][1]							
METER[0][0]							

Supported Downlink commands. Server-side request.

REQ_ONDEMAND_RES. Request command to transfer response with changed parameters. This command is used in the MTX lorawan module. This command works only with electricity meter MTX. This command should be sent before command **MTX_CMD.** Command's size is one byte. Field CR (bit's 7..4) defines CODE RATE of response (0-4/5, 1 - 4/6, 2-4/7,3-4/8), TXDROFFSET defines the offset of the transfer rate of the current setup, see table below.

Current UL SF		TXDROFFSET (min 0, max 15 - bin4)															
SF		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SF12	0	0	1	2	3	4	5	5	5	0	0	0	0	0	0	0	0
SF11	1	1	2	3	4	5	5	5	5	0	0	0	0	0	0	0	0
SF10	2	2	3	4	5	5	5	5	5	0	0	0	0	0	0	0	1
SF9	3	3	4	5	5	5	5	5	5	0	0	0	0	0	0	1	2
SF8	4	4	5	5	5	5	5	5	5	0	0	0	0	0	1	2	3
SF7	5	5	5	5	5	5	5	5	5	0	0	0	0	1	2	3	4

where - SF12, 1 - SF11, 2 - SF10, 3 - SF9, 4 - SF8, 5 - SF7

SET_TIME2000

SET_TIME2000. Time setting command. Sent to the sensor to set the time. The **TIME_SEQ** parameter in the command must be different from the same parameter in the **TIME2000** command sent from the sensor. This is done to avoid double processing of the command. The new sensor time will equal the current sensor time plus the value passed in the **SET_TIME2000** command. The **TIME2000**[4] parameter in the command is signed. The sensor sends a response to the command.

SET_TIME2000 = 0x02 - Request							
7	6	5	4	3	2	1	0
0	0	0	CMD SET_TIME2000 = 0x02				
CMD_LEGTH = 5							
TIME_SEQ							
TIME2000[3]							
TIME2000[2]							
TIME2000[1]							
TIME2000[0]							

SET_TIME2000 = 0x02 - Response							
7	6	5	4	3	2	1	0
0	0	0	CMD SET_TIME2000 = 0x02				
CMD_LEGTH = 1							
STATUS*							

*STATUS= «1» – the time setting was successful, "0" - time setting failed (the TIME_SEQ parameter was not changed).

SET_PARAMETERS

SET_PARAMETERS. The command to set the parameters of the sensor. The list of possible parameters is given below. The sensor confirms the execution of the command.

SET_PARAMETERS = 0x03.							
7	6	5	4	3	2	1	0
0	0	0	CMD SET_PARAMETERS = 0x03				
CMD_LENGTH*							
TYPE_PARAMETERS*							
DATA_PARAMETERS[N]							

*The length of the command depends on the context (the type of parameter being set).

Description of possible parameters

TYPE_PARAMETERS = 1. Setting the reporting period (time)
TIME_SHIFT[2] – value defines the offset of the first data transmission starting from the beginning of the day. This is a 2-byte value with a resolution of 2 minutes. **Deprecated setting. The field is currently unused.**
RANDOM_PERIOD[1] – one byte determines the maximum size of the pseudo-random part of the period. This is a 1-byte value with a resolution of 600 seconds. **Deprecated setting. The field is currently unused.**
PERIOD[1] – value determines the frequency of sending data from the sensor. This is a 1-byte value with a resolution of 600 seconds plus a pseudo-random value of no more than 511 seconds. If the parameter is not set, the data sending period will equal 13320 seconds plus a pseudo-random value of no more than 551 seconds.

To set PERIOD 600sec. It's very fast and use only for example:

03 05 - Command SET_PARAMETERS len=5

01 - TYPE_PARAMETERS = 1

00 00 - Reserved

00 01 - PERIOD[1] = 600sec/600=0x1

TYPE_PARAMETERS = 4. Day report hour setting. The parameter defines the hour of the day by which the daily consumption is calculated.
0_HOUR[1] – value from 0 to 23. By default, checkout time is 0.

TYPE_PARAMETERS = 5. Type of reporting data
DATA_TYPE[1]. 0 – hourly consumption data, 1 - daily consumption data, 2 - current data. By default, hourly consumption data will be transmitted.

TYPE_PARAMETERS = 8. Method for delivering priority data. With confirmation or not. Such data may include transmitting events
TX_DATA[1]. 0 – data with confirmation of data delivery, 1 – data is transmitted without delivery confirmation.

TYPE_PARAMETERS = 9. Device LoRaWAN activation method
TX_DATA[1]. 0 (default value) – OTAA activation by sending a JOIN packet; "1" - ABP activation by writing all keys to the device without generating and sending a JOIN packet. OTAA by default.

TYPE_PARAMETERS = 15. PowerCFG. Only for the MTX lorawan module
Set the half hours data types. This is the bitwise field, where bit 0 – ACTIV active energy for half hour (A+), bit 1 – VARI positive (capacitivy) reactive energy for half hour (A+R+), bit 2 - VARE negative (inductive) reactive energy for half hour (A+R-), бит 3 – ACTIVE_EXP active energy for half hour (A-), бит 4 – VARI_EXP positive (capacitivy) reactive energy for half hour (A-R+), bit 5 – VARE_EXP negative (inductive) reactive energy for half hour (A-R-). Bit equal to 1 means that the data for such type of energy will be transferred. By default value is equal to 1 and only data for ACTIV active energy will be transfered.

TYPE_PARAMETERS = 14. Data transmission schedule. Only for the MTX lorawan module.
Setup the schedule of the different data transmissions.
Type data, 1-byte value. 0 - half hour consumption data, 1 - daily consumption data, 2 - current data, 3 - module status.
Transmission period. 1-byte value. Equals 600 seconds plus random value less than 1020 seconds.
Half hour schedule. 3-bytes value. 24 bits бита defines which half hours schedule will be applied. Bit equals to 1 means that the correspondent schedule will be applied. First byte contains bits for the 23 - 16 hours, second byte 15 - 8 hours and final byte 7 - 0 hours.

TYPE_PARAMETERS = 0xE
DATA_TYPE = 0 (half hour consumption data)
TIME
HOURS[23..16]
HOURS[15..8]
HOURS[7..0]
DATA_TYPE = 1 (daily consumption data)
TIME
HOURS[23..16]
HOURS[15..8]
HOURS[7..0]
DATA_TYPE = 2 (current data)
TIME
HOURS[23..16]
HOURS[15..8]

HOURS[7..0]
DATA_TYPE = 3 (module status)
TIME
HOURS[23..16]
HOURS[15..8]
HOURS[7..0]

TYPE_PARAMETERS = 16. MULTICASTCFG. Only for the MTX lorawan module. There are 4 multicast group. All groups have the same set of encryption keys (mcstNetsKey[16], mcstAppsKey[16]).
MCST_GROUP, 1-byte value. Number of the multicast group.
MCST_ADDR[4], 4-bytes value. Address of the multicast group. The most significant byte is transmitted first.
minTime, 1 - byte value. Minimum delay in seconds before a response for the multicast request.
maxRNDTime[2], 2-bytes value. Maximum value of the random delay in seconds before a response for the multicast request. The most significant byte is transmitted first.
netsMCSTKey[16], 16-bytes value. Network encryption key of the multicast group.
appsMCSTKey[16], 16-bytes value. Application encryption key of tne multicast group.

TYPE_PARAMETERS = 19. HOURS_OFFSET_CFG. Only for the MTX lorawan module for now. Setup the degree of the accuracy for the half hours data transmition.
OFFSET, 5-bit value. Percent of the data repetition defines by formula: 100*OFFSET/16. Therefore the maxumum data repetition persent is 31*100/16=193%. By default, it equals 0.

TYPE_PARAMETERS = 20. LAST_DAYCMD_CFG. For the MTX lorawan module. Allows the new option to request only active tariffs.
NEWCMD. If “0” - all tariffs, “1” - active tariffs only. By default, it equals to 0.

TYPE_PARAMETERS = 22. DELTA_TIME_EN. When setting the parameter, before transmitting data of the hourly values of the pulse counter, the DELTA_TIME command will be substituted, which contains the time elapsed since the last hourly value was recorded and the moment the frame was transmitted. By default, DELTA_TIME_EN = 0, and the DELTA_TIME command will not be substituted.

TYPE_PARAMETERS = 23. METER_BASE_DATA. Using this parameter, the initial consumption of the meter and its impulse coefficient are set. In the future, the module can transmit the absolute data of the meter.
METER_BASE_DATA[4] 4-byte value of the initial consumption of the meter, taking into account the impulse coefficient. The high byte comes first.
METER_IMPK[1] 1 byte value of the pulse coefficient of the metering device (corresponding to the consumed resource per 1 pulse).

TYPE_PARAMETERS = 24. ABSOLUTE_DATA_EN. Permission to transmit absolute data. If parameters “0” module will send impulse counter value, if parameters is “1” and setting parameters 23, then module send value of meter.

In firmware version >= 2.92 we have the possibility to get from device the absolute consumption value.
To get the absolute consumption value we need:

- set parameters 23 - initial consumption value and impulse coefficient
- and use parameter 24 to start to send absolute consumption value.

You can send it in one message.
Example:
initial meter value - 2.534 м3 -> you need to set in parameter 2543/10 = 254 = 0xFE
impulse coefficient - 0.01m3 (10 liter per pulse) -> 0x0A
As a result we need to send : 030617000000FE0A03021801+LRC
And expect in response: 0302170103021801+LRC

TYPE_PARAMETERS = 25. SERIAL_NUMBER. For modules with pulse input. The serial number of the meter is recorded. Dimension 6 bytes. The high byte comes first.

TYPE_PARAMETERS = 26. GEOLOCATION. It is possible to set the physical location of the sensor.
Latitude[4] - latitude 4 bytes
Longitude[4] - longitude 4 bytes
Attitude[2] - altitude 2 bytes

SET_PARAMETERS = 0x03. Response to a parameter setting command.							
7	6	5	4	3	2	1	0
0	0	0	SET_PARAMETERS = 0x03				
CMD_LEGTH = 2							
TYPE_PARAMETERS							
STATUS (1 - operation successful, 0 - error)*							

*In old version of firmware (before 91) module don't send result of operation.

GET_PARAMETERS

GET_PARAMETERS. The command to read the parameters set in the sensor. In the body of the command, you must specify the type of the parameter. In response, the sensor will transmit the current value of the requested parameter.

GET_PARAMETERS = 0x04. Command for requesting sensor operation parameters.							
7	6	5	4	3	2	1	0
0	0	0	CMD GET_PARAMETERS = 0x04				
CMD_LEGTH= 1							
TYPE_PARAMETERS*							

GET_PARAMETERS = 0x04. Response to the parameter request command.							
7	6	5	4	3	2	1	0
0	0	0	CMD GET_PARAMETERS = 0x04				
CMD_LEGTH*							
TYPE_PARAMETERS*							
DATA_PARAMETERS[N]							

GET_ARCHIVE_HOURS

GET_ARCHIVE_HOURS. Hourly consumption data archive query command. In the command data field, you must set the date and hour of the start of reading the archive. If there is no data in the archive, then 0xFFFFFFFF will be returned as the base value. Since the length of the transmitted data from the sensor is limited, not all the requested data will be transmitted.

GET_ARCHIVE_HOURS = 0x05							
7	6	5	4	3	2	1	0
0	0	0	CMD GET_ARCHIVE_HOURS = 0x05				
CMD_LEGTH= 4							
YEAR[7..0]							MONTH[3]
MONTH[2..0]			DATE[4..0]				
			HOUR[4..0] – starting hour				
NUMBER (number of samples)							

Response to the command to request the hourly data archive of the sensor pulse counter. If there has been magnet influence during the requested hour, then the Magnet bit will be set to 1.

GET_ARCHIVE_HOURS = 0x05. Response to the command to request the hourly data archive of the sensor pulse counter.							
7	6	5	4	3	2	1	0
0	0	0	CMD GET_ARCHIVE_HOURS = 0x05				
CMD_LEGTH= 2+4+hours*2							
YEAR[7..0]							MONTH[3]
MONTH[2..0]			DATE[4..0]				
Magnet	reserved	reserved	HOUR				
Counter[0][2]							
Counter[0][1]							
Counter[0][0]							
…							
Magnet	reserved	reserved	Counter_dif[12..8]				
Counter_dif[hours-1][7..0]							

GET_ARCHIVE_DAYS

GET_ARCHIVE_DAYS. Command for requesting the daily data archive of the sensor impulse counter. In the command data field, you must set the start date for reading the archive. If there is no data in the archive, then 0xFFFFFFFF is returned. Since the length of the transmitted data from the sensor is limited, not all the requested data will be transmitted.

GET_ARCHIVE_DAYS = 0x06.							
7	6	5	4	3	2	1	0
0	0	0	CMD GET_ARCHIVE_DAYS = 0x06				
CMD_LEGTH= 3							
YEAR[7..0]							MONTH[3]
MONTH[2..0]			DATE[4..0]				
NUMBER (number of samples)							

Response to the command to request the daily data archive of the sensor impulse counter. If magnet exposure occurred during the requested day, the Magnet bit would be set to 1.

GET_ARCHIVE_DAYS = 0x06.							
7	6	5	4	3	2	1	0
0	0	0	CMD GET_ARCHIVE_DAYS = 0x06				
CMD_LEGTH= 2+days*4							
YEAR[7..0]							MONTH[3]
MONTH[2..0]			DATE[4..0]				
Counter[0][2]							
Counter[0][1]							
Counter[0][0]							
…							
Magnet	reserved	reserved	0				
Counter[days-1][2]							
Counter[days-1][1]							
Counter[days-1][0]							

GET_CURRENT

GET_CURRENT. The command to request the current readings of the sensor pulse counter.

GET_CURRENT = 0x07.							
7	6	5	4	3	2	1	0
0	0	0	CMD GET_CURRENT = 0x07				
CMD_LEGTH= 0							

Response to the command to request the current readings of the sensor pulse counter. The sensor can periodically send this command without request if the type of reporting data (TYPE_PARAMETERS = 5) has been set to 2 - transmission of current data. In the body of the command, the current value of the pulse counter is transmitted.

GET_CURRENT = 0x07.							
7	6	5	4	3	2	1	0
0	0	0	CMD GET_CURRENT = 0x07				
CMD_LEGTH= 4							
Magnet	reserved	reserved	0				
Counter[2]							
Counter[1]							
Counter[0]							

GET_ARCHIVE_EVENTS

GET_ARCHIVE_EVENTS. The command to request the sensor event archive. In the body of the command, the time starting from which to read the event archive is transmitted. If parameter TIME2000 = 0, then the oldest sensor events will be read. If TIME2000=0xFFFFFFFF, then the most recent sensor events will be read.

GET_ARCHIVE_EVENTS = 0x0B. The command to request the sensor event archive.							
7	6	5	4	3	2	1	0
0	0	0	CMD GET_ARCHIVE_EVENTS = 0x0B				
CMD_LEGTH = 5							
TIME2000[4] – time from which to watch events							
EVENTS (number of events)							

GET_ARCHIVE_EVENTS = 0x0B. Response to the request command for the sensor event archive.							
7	6	5	4	3	2	1	0
0	0	0	CMD GET_ARCHIVE_EVENTS = 0x0B				
CMD_LEGTH = (4+1+1)*EVENTS							
TIME_EVENT[4]							
EVENTS – event codes							
EVENT_SEQUENCE							

CORRECT_TIME2000

CORRECT_TIME2000. Time correction command. The **TIME_SEQ** parameter in the command must be different from the same parameter in the **TIME2000** command sent from the sensor. This is done to avoid double processing of the command. The new sensor time will equal the current sensor time plus the value passed in the **CORRECT_TIME2000** command. The **DIFF_TIME2000** parameter in the command is signed. The sensor sends a response to the command.

CORRECT_TIME2000 = 0x0C - Request							
7	6	5	4	3	2	1	0
0	0	0	CMD CORRECT_TIME2000 = 0x0C				
CMD_LEGTH = 2							
TIME_SEQ							
DIFF_TIME2000							

CORRECT_TIME2000 = 0x0C. Response							
7	6	5	4	3	2	1	0
0	0	0	CMD CORRECT_TIME2000 = 0x0C				
CMD_LEGTH = 1							
STATUS*							

*STATUS= «1» – the time setting was successful, "0" - the time setting did not occur (the TIME_SEQ parameter was not changed).

GET_NEW_STATUS

GET_NEW_STATUS. The command to request status information from the sensor. New command version.

The command is sent by the sensor without request once a day. LOW_CURRENT_VBATTERY - battery voltage with minimum load (SLEEP MODE), value in mV. HIGH_CURRENT_VBATTERY – battery voltage with load simulating transmission mode, value in mV. INT_RESISTANT is the internal resistance of the battery in mΩ. If the voltage value is 4095 mV, then the value is unknown. If the internal resistance value is 65535 mOhm, then the resistance is unknown. TEMPERATURE is a signed number in degrees Celsius.

REMINED_BATTERY_CAPACITY is the remaining battery capacity, where 254 is 100%. A value of 255 means that the remaining battery capacity is unknown. LAST_EVENT - last unread sensor event in 48 hours. This field will automatically reset after 48 hours. After reading the sensor status, this field is also reset.

GET_NEW_STATUS = 0x14 - Request							
7	6	5	4	3	2	1	0
0	0	0	CMD GET_NEW_STATUS = 0x14				
CMD_LEGTH = 0							

NEW_STATUS = 0x14 - Response							
7	6	5	4	3	2	1	0
0	0	0	CMD GET_NEW_STATUS = 0x14				
CMD_LEGTH = 12							
SOFT_TYPE							
SOFT_VERSION							
HARD_TYPE							
HARD_VERSION							
LOW_CURRENT_VBATTERY[11..4]							
LOW_CURRENT_VBATTERY[3..0]				HIGH_CURRENT_VBATTERY[11..8]			
HIGH_CURRENT_VBATTERY[7..0]							
INT_RESISTANT[15..8]							
INT_RESISTANT[7..0]							
TEMPERATURE							
REMINED_BATTERY_CAPACITY							
LAST_EVENT							

For the MTX lorawan module:

NEW_STATUS = 0x14 - Response							
7	6	5	4	3	2	1	0
0	0	0	CMD GET_NEW_STATUS = 0x14				
CMD_LEGTH = 20							
SOFT_TYPE							
SOFT_VERSION							
HARD_TYPE							
HARD_VERSION							
Time[4]							
CAUSE_RESET							
RSSI LAST DW FRAME							
SNR LAST DW FRAME							
CNT DWN REQ							
CNT DWN FRAGS REQ							
CNT UP RES							
CNT UP FRAGS RES							
MOTE-GW MARGIN							
GW-MOTE MARGIN							
NGWS							
DW QUALITY							
LAST_EVENT							

SOFT_RESTART

SOFT_RESTART. Command for software restart. Device restart in ~30 second with new LoraWAN parameters.

SOFT_RESTART = 0x19.							
7	6	5	4	3	2	1	0
0	0	0	CMD **SOFT_RESTART** = 0x19				
CMD_LEGTH=0							

Device confirm this message to send the uplink message:

SOFT_RESTART = 0x19.							
7	6	5	4	3	2	1	0
0	0	0	CMD **SOFT_RESTART** = 0x19				
CMD_LEGTH=0							

Example with LRC: 19 00 4C

GET_ARCHIVE_HOURS_MUL

GET_ARCHIVE_HOURS_MUL. The request for the archive of hours-based consumption data. The request must contain the date and the first hour of the archived data to be received in the response.. **CHANNELS** is a bit field and describes the channel set of the request. The **HOUR** field holds the count of the hour's sections - 1 (8 is the maximum). If there is no data in the archive, then 0xFFFFFFFF will be in the response as a reference value. The response's length is limited, so only part of the data may fit the response's size.

GET_ARCHIVE_HOURS_MUL = 0x1A							
7	6	5	4	3	2	1	0
0	0	0	CMD **GET_ARCHIVE_HOURS_MUL** = 0x1A				
CMD_LEGTH= 4							
YEAR[7..0]							MONTH[3]
MONTH[2..0]			DATE[4..0]				
HOURS			HOUR[4..0] – стартовый час				
EXTEND=0	CHANNELS						

Response to the **GET_ARCHIVE_HOURS_MUL** command.

7	6	5	4	3	2	1	0
0	0	0	CMD **GET_ARCHIVE_HOURS_MUL** = 0x1A				
CMD_LENGTH							
Year[7..0]							Month[3]
Month[2..0]			Date[4..0]				
Hours[2..0]			Hour[4..0]				
EXTEND=0	CHANNELS[6..0]						
EXTEND=1	COUNTER_1[6..0]						
EXTEND=1	COUNTER_1[13..7]						
EXTEND=0	COUNTER_1[20..14]						
EXTEND=1	COUNTER_1_DIF_1[6..0]						
EXTEND=0	COUNTER_1_DIF_1[13..7]						

EXTEND=1	COUNTER_1_DIF_2[6..0]
EXTEND=1	COUNTER_1_DIF_2[13..7]
EXTEND=0	COUNTER_1_DIF_2[20..14]
EXTEND=1	COUNTER_2[6..0]
EXTEND=1	COUNTER_2[13..7]
EXTEND=1	COUNTER_2[20..14]
EXTEND=0	COUNTER_2[27..21]
EXTEND=0	COUNTER_2_DIF_1[6..0]
EXTEND=1	COUNTER_2_DIF_2[6..0]
EXTEND=0	COUNTER_2_DIF_2[13..7]
…	
EXTEND=1	COUNTER_N[6..0]
EXTEND=0	COUNTER_N[13..7]
EXTEND=1	COUNTER_N_DIF_1[6..0]
EXTEND=1	COUNTER_N_DIF_1[13..7]
EXTEND=1	COUNTER_N_DIF_1[20..14]
EXTEND=0	COUNTER_N_DIF_1[27..21]
EXTEND=1	COUNTER_N_DIF_2[6..0]
EXTEND=0	COUNTER_N_DIF_2[13..7]

GET_ARCHIVE_DAYS_MUL

GET_ARCHIVE_DAYS_MUL. The request for the archive of daily data of the pulse counter multichannel sensor. The request must contain the first date of the archived data to be received in the response. If there is no data in the archive, then 0xFFFFFFFF will be in the response. The response's length is limited, so only part of the data may fit the response's size.

GET_ARCHIVE_DAYS_MUL = 0x1B.							
7	6	5	4	3	2	1	0
0	0	0	CMD **GET_ARCHIVE_DAYS_MUL** = 0x1B				
CMD_LEGTH= 4							
YEAR[7..0]							MONTH[3]
MONTH[2..0]			DATE[4..0]				
EXTEND=0	CHANNELS						
DAYS							

Response to the **GET_ARCHIVE_DAYS_MUL** command.

7	6	5	4	3	2	1	0
0	0	0	CMD **GET_ARCHIVE_DAYS_MUL** = 0x1B				
CMD_LEGTH							
YEAR[7..0]							MONTH[3]
MONTH[2..0]			DATE[4..0]				
EXTEND=0	CHANNELS[6..0]						
DAYS							
EXTEND=1	COUNTER_1_1[6..0]						
EXTEND=1	COUNTER_1_1[13..7]						
EXTEND=0	COUNTER_1_1[20..14]						
EXTEND=1	COUNTER_1_2[6..0]						
EXTEND=1	COUNTER_1_2[13..7]						

EXTEND=0	COUNTER_1_2[20..14]
…	
EXTEND=1	COUNTER_1_N[6..0]
EXTEND=1	COUNTER_1_N[13..7]
EXTEND=0	COUNTER_1_N[20..14]
EXTEND=1	COUNTER_2_1[6..0]
EXTEND=1	COUNTER_2_1[13..7]
EXTEND=0	COUNTER_2_1[20..14]
EXTEND=1	COUNTER_2_2[6..0]
EXTEND=1	COUNTER_2_2[13..7]
EXTEND=0	COUNTER_2_2[20..14]
…	
EXTEND=1	COUNTER_2_N[6..0]
EXTEND=1	COUNTER_2_N[13..7]
EXTEND=0	COUNTER_2_N[20..14]

CLEAR_PARAMETERS

CLEAR_PARAMETERS = 0x1D. Command to reset device parameters to default factory values.

CLEAR_PARAMETERS = 0x1D.							
7	6	5	4	3	2	1	0
0	0	0	CMD **CLEAR_PARAMETERS** = 0x1D				
CMD_LEGTH=0							

Response to the **CLEAR_PARAMETERS** command. Confirms the execution of a command.

CLEAR_PARAMETERS = 0x1D.							
7	6	5	4	3	2	1	0
0	0	0	CMD **CLEAR_PARAMETERS** = 0x1D				
CMD_LEGTH=0							

MTX_CMD = 0x1E. Read data from an electricity meter MTX

MTX_CMD = 0x1E.							
7	6	5	4	3	2	1	0
0	0	0	CMD **MTX_CMD = 0x1E**				
CMD_LEGTH							
SEQ							
LAST	FRAGS			RES	FRAG		
MSG_ID*							
PROTOCOL+ACCESS LEVEL*							
PROTOCOL+ACCESS LEVEL*							
CMD*							
CMD_LENGTH*							
DATA[CMD_LENGTH]*							

*Data for electricity meter encoded in MTX protocol data format exept fields FRM_TYPE, DST_ADDR, SRC_ADDR.

SEQ – sequence number, the same for all fragments of the MTX frame,

LAST – last fragment flag, «1» - is the last fragment,

FRAGS – fragments count,

ES – spare,

FRAG – fragment's index.

MTX_CMD command response. The data size could be bigger than frame size, fragmentation will be used then.

MTX_CMD = 0x1E.							
7	6	5	4	3	2	1	0
0	0	0	CMD **MTX_CMD = 0x1E**				
CMD_LEGTH							
SEQ							
LAST	FRAGS			RES	FRAG		
DATA_MTX[CMD_LENGTH-2]							

SEQ – sequence number, the same for all fragments of the MTX frame,

LAST – last fragment flag, «1» - is the last fragment,

FRAGS – fragments count,

RES – spare,

FRAG – fragment's index.

DATA_MTX:

MSG_ID*
PROTOCOL+ACCESS LEVEL*
PROTOCOL+ACCESS LEVEL*
CMD*
CMD_LENGTH*
DATA[CMD_LENGTH]*

Example of data separated into three fragments:

MTX_CMD = 0x1E.							
7	6	5	4	3	2	1	0
0	0	0	CMD **MTX_CMD = 0x1E**				
35							
SEQ							
0	3			1	1		
DATA_MTX[0..32]							

MTX_CMD = 0x1E.							
7	6	5	4	3	2	1	0
0	0	0	CMD **MTX_CMD = 0x1E**				
35							
SEQ							
0	3			1	2		
DATA_MTX[33..65]							

MTX_CMD = 0x1E.							
7	6	5	4	3	2	1	0
0	0	0	CMD **MTX_CMD = 0x1E**				
36							
SEQ							
1	3			1	3		
DATA_MTX[66..99]							

Command **GET_TIME** request. Get time of the electricity meter MTX.

MTX_CMD = 0x1E.							
7	6	5	4	3	2	1	0
0	0	0	CMD MTX_CMD = 0x1E				
CMD_LEGTH = 0x09							
SEQ							
1	001			0	1		
MSG_ID*							
PROTOCOL+ACCESS LEVEL = 0x10							
PROTOCOL+ACCESS LEVEL = 0x10							
GET_TIME=0x07							
CMD_LENGTH = 0							
CMD_END = 0							
LRC = 0 (MTX LRC!)							

Dump: 1e 09 23 91 23 10 10 07 00 00 00 d4

Commnand **GET_TIME** response:

MTX_CMD = 0x1E.							
7	6	5	4	3	2	1	0
0	0	0	CMD MTX_CMD = 0x1E				

CMD_LEGTH = 0x11			
SEQ			
1	001	0	1
MSG_ID*			
PROTOCOL+ACCESS LEVEL=0x10			
PROTOCOL+ACCESS LEVEL=0x10			
GET_TIME=0x07			
CMD_LENGTH = 0x08			
SummerFlag “0”-winter,”1”-summer			
Sec			
Min			
Hour			
Day (1-Sunday,2-Monday,..,7-Saturday)			
Date			
Month			
Year			
CMD_END			
LRC (MTX LRC!)			

Dump: 1e 11 ab 91 23 10 10 07 08 00 0c 21 0c 03 15 02 17 00 68 06

Command **SET_TIME** request**.** Setup time for the electricity meter MTX.

MTX_CMD = 0x1E.							
7	6	5	4	3	2	1	0
0	0	0	CMD MTX_CMD = 0x1E				
CMD_LEGTH = 0x11							
SEQ							
1	001			0	1		
MSG_ID							
PROTOCOL+ACCESS LEVEL = 0x10							

PROTOCOL+ACCESS LEVEL = 0x10
GET_TIME=0x08
CMD_LENGTH = 0x08
SummerFlag “0”-winter,”1”-summer
Sec = 0
Min = 0x3a (58d)
Hour = 0x0c (12d)
Day (1-Sunday,2-Monday,..,7-Saturday) 3
Date = 0x15 (21)
Month = 0x02
Year = 0x17 (23)
CMD_END = 0
LRC = 0 (MTX LRC!)

Dump: 1e1123912410100808000003a0c031502170000f9

Command **SET_TIME** response:

MTX_CMD = 0x1E.							
7	6	5	4	3	2	1	0
0	0	0	CMD MTX_CMD = 0x1E				
CMD_LEGTH = 0x09							
SEQ							
1	001			0	1		
MSG_ID							
PROTOCOL+ACCESS LEVEL = 0x10							
PROTOCOL+ACCESS LEVEL = 0x10							
SET_TIME=0x08							
CMD_LENGTH = 0							
CMD_END = 0							
LRC (MTX LRC!)							

Dump: 1e 09 89 91 24 10 10 08 00 00 4d 3b
MTX lorawan module also creates a frame with daily consumption data and sends it within a defined period without request from the server.

Commands relevant for the 4PI module (4-input)

Uplink data from the rf module

CMD	Description
LAST_EVENTS = 0x60-0x7F	Last unread event
TIME2000 = 0x09	Current rf module time (time 2000 format)
NEW_STATUS=0x14	RF Module status
NEW_EVENT=0x15	New event occurs
DATA_DAY_MUL=0x16	Pulse counter data of the multichannel sensor on billing hour.
DATA_HOUR_MUL=0x17	Pulse counter data of the multichannel sensor, accumulated on a half-hour basis.
GET_CURRENT_MUL=0x18	Command for output current data of the pulse counter a multichannel sensor

Supported **Downlink** commands. Server-side request.

CMD	Description
SET_TIME2000 = 0x02	Time correction command. 4bytes signed value in seconds Time2000 format
SET_PARAMETERS = 0x03	Set parameters command
GET_PARAMETERS = 0x04	Get parameters command
TIME2000 = 0x09	RF module current time
GET_ARCHIVE_EVENTS=0x0B	Read events archived data
CORRECT_TIME2000=0x0C	Command data consists of a sequence number and one byte signed correction value. Command accepted only if sequence number greater than value previously received by the module
GET_NEW_STATUS=0x14	Get rf module status
GET_ARCHIVE_HOURS_MUL=0x1A	Read pulse counter hours-based archived data of the multichannel sensor.
GET_ARCHIVE_DAYS_MUL = 0x1B	Read pulse counter archived daily data of the multichannel sensor on billing hour.

List of parameters valid for the 4PI module

TYPE_PARAMETERS = 1. Setting the reporting period (time).
TYPE_PARAMETERS = 4.Day report hour setting.
TYPE_PARAMETERS = 5. Type of reporting data.
TYPE_PARAMETERS = 8. Method for delivering priority data.
TYPE_PARAMETERS = 9. Device LoRaWAN activation method.
TYPE_PARAMETERS = 10. Parameters of depassivation of battery.
TYPE_PARAMETERS = 11. Minimum time a day activity for battery (warehouse mode).

Representing integers using the minimum bytes

For the pulse counter, and not only, we use the format of the integer representation, which minimizes the number of bytes required for data transmission over communication channels. The EXTEND bit is used for this. If the EXTEND bit = "1", then this means that the next upper 7 bits of the pulse counter will be placed in the next byte. If EXTEND = "0" will mean that this is the last byte with pulse counter data.

Device stores the pulse counter in unsigned integer - 4 byte, 32 bit. We need from 1 to 5 bytes to transmit the pulse counters in messages.

EXTEND=1	COUNTER[6..0]
EXTEND=1	COUNTER[13..7]
EXTEND=1	COUNTER[20..14]
EXTEND=1	COUNTER[27..21]
EXTEND=0	COUNTER[31..28]

Example:

Counter per channel stored in device 0x0A - coded in 1 byte -> dump 0x0A ->
EXTEND bit = 0 (last byte), counter = 0x0A&0x7F= 0x0A= 0x00 00 00 0A -> = 10 in dec

Counter per channel 0x83 - coded in 2 bytes -> dump 0x83 0x01 ->
0x83:EXTEND bit = 1(not last byte), value1 = 0x83&0x7F= 0x03;
0x01:EXTEND bit =0 (last byte), value2 = 0x01;
Result counter = value1 + (value2<<7)=0x03+(0x01<<7)=0x83-> 0x00 00 00 83 = 137 in dec

Counter per channel stored in device 0x5503 - coded in 3 bytes -> dump 0x83 0xAA 0x01 ->
0x83:EXTEND bit = 1(not last byte), value1 = 0x83&0x7F= 0x03;
0xAA:EXTEND bit = 1(not last byte), value2 = 0xAA&0x7F= 0x2A;
0x01:EXTEND bit =0 (last byte), value3 = 0x01
Result counter = value1 + (value2<<7)+(value3<<14)=
0x03+(0x2A<<7)+ (0x01<<14)=0x5503-> 0x00 00 55 03 = 21 763 in dec

Counter per channel stored in device 0x2A81BFF - coded in 4 bytes -> dump 0xBF 0x83 0xAA 0x01 ->
0xBF:EXTEND bit = 1(not last byte), value1 = 0xBF&0x7F= 0x3F;
0x83:EXTEND bit = 1(not last byte), value2 = 0x83&0x7F= 0x03;
0xAA:EXTEND bit = 1(not last byte), value3 = 0xAA&0x7F= 0x2A;
0x01:EXTEND bit =0 (last byte), value4 = 0x01
Result counter = value1 + (value2<<7)+(value3<<14)+(value4<<21)=
0x3F+ (0x03<<7)+(0x2A<<14)+ (0x01<<21)=0x2A81BFF-> 0x2 A8 1B FF = 2 785 727 in dec

Counter per channel stored in device 0xFFFFFFFF - coded in 5 bytes -> dump 0xFF FF FF FF FF ->
0xFF:EXTEND bit = 1(not last byte), value1 = 0xFF&0x7F= 0x7F;
0xFF:EXTEND bit = 1(not last byte), value2 = 0xFF&0x7F= 0x7F;
0xFF:EXTEND bit = 1(not last byte), value3 = 0xFF&0x7F= 0x7F;
0xFF:EXTEND bit = 1(not last byte), value4 = 0xFF&0x7F= 0x7F;
0x0F:EXTEND bit = 0(last byte), value5 = 0x0F&0x7F= 0x0F;
Result counter = value1 + (value2<<7)+(value3<<14)+(value4<<21)+(value5<<28)=
0x7F+ (0x7F<<7)+(0x7F<<14)+ (0x7F<<21)+(0x0F<<28)=0xFFFFFFFF-> 0xFF FF FF FF = 4 294 967 295 in dec