LORATECH MultiSense
Manual (pre) !! Preliminary – Confidential !!

Author: Jan Bělohoubek (JB), Martin Úbl (MU), Ondřej Růžička (OR)

Version pre from 10.3.2020

General Description

LORATECH MultiSense is the sensing and remote metering platform supplied by RVTech s r.o.

LORATECH MultiSense is the commercial and proprietary extension of KETCube platform developed by RVTech s r.o.

KETCube platform is developed by the Department of Technologies and Measurement (KET), University of West Bohemia in Pilsen [1].

This document describes the *LORATECH MultiSense* device family.

Main Features

- Communication: LoRaWAN Class A or C device
- Debug: Advanced LORATECH Terminal
- Remote terminal fuctionality
- RHT Sensor (HDC1080): Relative Humidity & Temperature
- S0 Counter: pulse counter (up to 4 devices)
- Extensible: compatible with KETCube [1]
- Power supply:
  - 3V6 battery: LS 14500 (recommended battery)
  - 8V – 25V IN (external adapter)
  - 5V USB IN (optional assembly – on request only)

Contents

1 Specifications **1**

2 Product Family Naming **3**

3 Board Layout **5**

4 Firmware Upgrade **6**

5 Operation Basics **9**

6 Debug and Configuration – LORATECH Terminal **10**

7 LoRa Configuration **13**

8 RHT Sensor (HDC1080) Configuration **15**

9 ModBUS Configuration **16**

10 S0 Counter Configuration **19**

11 Brownout Detection **22**

12 Maxbotix MaxSonar MB7040 **23**

13 Test Button Configuration **24**

List of Figures

1 Physical Layout of the *LORATECH MultiSense* 5
2 STM32 ST LINK Utility 7
3 LORATECH Terminal in Putty terminal emulator 11

List of Tables

Absolute Maximum Ratings 1
Operating Conditions 1
Operating Conditions – RHT Sensor (HDC1080) 1
Operating Conditions – S0 Counter 2
Operating Conditions – LS 14500 2
Product Family Members 3
Feature Marking 3
Power Supply Marking 4
Boxing Marking 4
Peripheral Marking 4
Board Layout 5
Main Board SWD 7

Revision History

Revision	Date	Author	Note
pre	10.3.2020	JB, MU, OR	pre-release internal version

1 Specifications

1.1 Absolute Maximum Ratings

Parameters	Symbol	MIN	TYP	MAX	UNIT
Supply Voltage (Logic – KETCube PINs)	3V3	-0.3	–	3.9	V
	Vref	-0.3	–	3V3 + 0.4	V
	GPIO	-0.3	–	3.9	V
Storage Temperature		-40	25	90	°C
Storage Humidity		20	–	70	%RH
Input RF Level		–	–	10	dBm
Supply Voltage (AA Battery)	3V3	-0.3	–	3.9	V
External Supply Voltage	VCC	-0.3	8 – 25	30	V

1.2 Operating Conditions

KETCube					
Parameters	**Cond.**	**MIN**	**TYP**	**MAX**	**UNIT**
Operating Temperature		-40	25	85	°C
Input current	3V, low-power mode	–	< 10	–	μA
Input current	3V, Tx: 14dBm	–	< 50	–	mA
LoRa range (Recommended 1/4 wave antenna)		–	0.5–10	–	km
Supply Voltage (USB)	USB charger or PC	-0.3	5	5.5	V
Supply Voltage (AA Battery)	LS 14500	2.2	3.6	3.65	V

RHT Sensor (HDC1080)[2]					
Parameters	**Cond.**	**MIN**	**TYP**	**MAX**	**UNIT**
Operating Humidity		0	–	100	%RH
Operating Temperature (functional)		-20	–	85	°C
RH Measurement Accuracy		–	± 2	–	%RH
Temperature Measurement Accuracy		–	± 0.2	± 0.4	°C

S0 Counter					
Parameters	**Cond.**	**MIN**	**TYP**	**MAX**	**UNIT**
Open Circuit Voltage	–	–	3.3	–	V
Allowed ON Resistance[a]	–	–	0 – 800	1k5	Ω
Required OFF Resistance	–	–	≥ 1M	–	Ω
Standard S0 Inputs					
S0 switching frequency	–	–	14[b]	30	Hz
Short circuit current	S0 Inter-PIN resistance ≤ 1 Ω	–	–	12	mA
Noise immunity	–	60	–	–	Hz
Low-Power S0 Inputs					
S0 switching frequency	–	–	14[c]	30	Hz
Short circuit current	up to 100 ms after switch	–	–	12	mA
Short circuit current	100 ms after switch	–	–	50	μA
Noise immunity	–	60	–	–	Hz
Ultra Low-Power S0 Inputs					
S0 switching frequency	–	–	14[d]	250	Hz
Short circuit current	up to 100 ms after switch	–	–	12	mA
Short circuit current	100 ms after switch	–	–	50	μA
Noise immunity	–	500	–	–	Hz

[a] The minimal current defined in [3] is achieved for MAX resistance only when VDD is above 2V9. The device is for MAX resistance operational downto 2V2 VDD
[b] See the European Standard EN 62053-31:1998 [3]
[c] See the European Standard EN 62053-31:1998 [3]
[d] See the European Standard EN 62053-31:1998 [3]

Recommended Battery LS 14500 [4]					
Parameters	**Cond.**	**MIN**	**TYP**	**MAX**	**UNIT**
Nominal capacity	at 2 mA, +20 °C, 2.0V	–	2.6	–	Ah
Open circuit voltage	+20 °C	–	3.67	–	V
Nominal voltage	+20 °C	–	3.6	–	V
Nominal energy	+20 °C	–	9.36	–	Wh
Maximum recommended continuous current		–	–	50	mA
Operating temperature range		-60	–	+85	°C

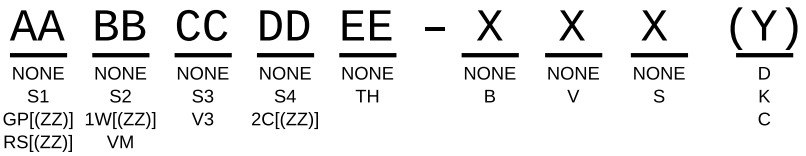
2 Product Family Naming

LORATECH MultiSense is the family of products comming with different embedded features. The products currently supplied by RVTECH are listed in the table below.

Name	Marking	Description
Ultrasonic	2C()TH-[B][V]	Low-cost and lower-precision distance metering
UltrasonicHQ	2C(MS)TH-[B][V]	Precise distance metering
RHT	1W()TH-[B][V]	Enviroment termperature and humidity sensor
ModBUS	RSTH-[B][V]	ModBUS/RS485 unit
Electricity	[S1][S2][S3][S4]TH-[B][V]	S0 counter input(s)

Variant Marking

LORATECH MultiSense is supplied in number of variants differing in enabled/available features and accessories. Each variant is marked depending on the enabled features. The device marking consist of the family name "*LORATECH MultiSense*" and the device-specific string expressing included features:



The above marking encodes the available features (AA – EE), power supply options (X), boxing type (Y) and connected peripherals (ZZ). The meaning of the codes is described in Tables below.

Feature Marking

#	Marking	Feature
AA	S1	S0 Counter input 1 (Ultra Low-Power input)
AA	GP	GPIO/UART
AA	RS	RS485/ModBUS
BB	S2	S0 Counter input 2 (Low-Power input)
BB	1W	1W interface
BB	VM	ADC (voltage measurement – up to 100V)
CC	S3	S0 Counter input 3
CC	V3	3V3 output
DD	S4	S0 Counter input 4
DD	2C	I2C BUS
EE	TH	RHT Sensor (HDC1080)

Power Supply

#	Marking	Power Supply
1	B	Battery (LS 14500 is recommended)
2	V	External Power Supply – DC Power (8 – 25V)
3	S	Solar Power Source

Boxing Variant

Marking	Boxing
D	Standard DIN-compatible box
K	Standard box
C	Standard box with an integrated DIN adapter

Peripheral Marking

Marking	Peripheral
MS	Maxbotix MaxSonar MB7040

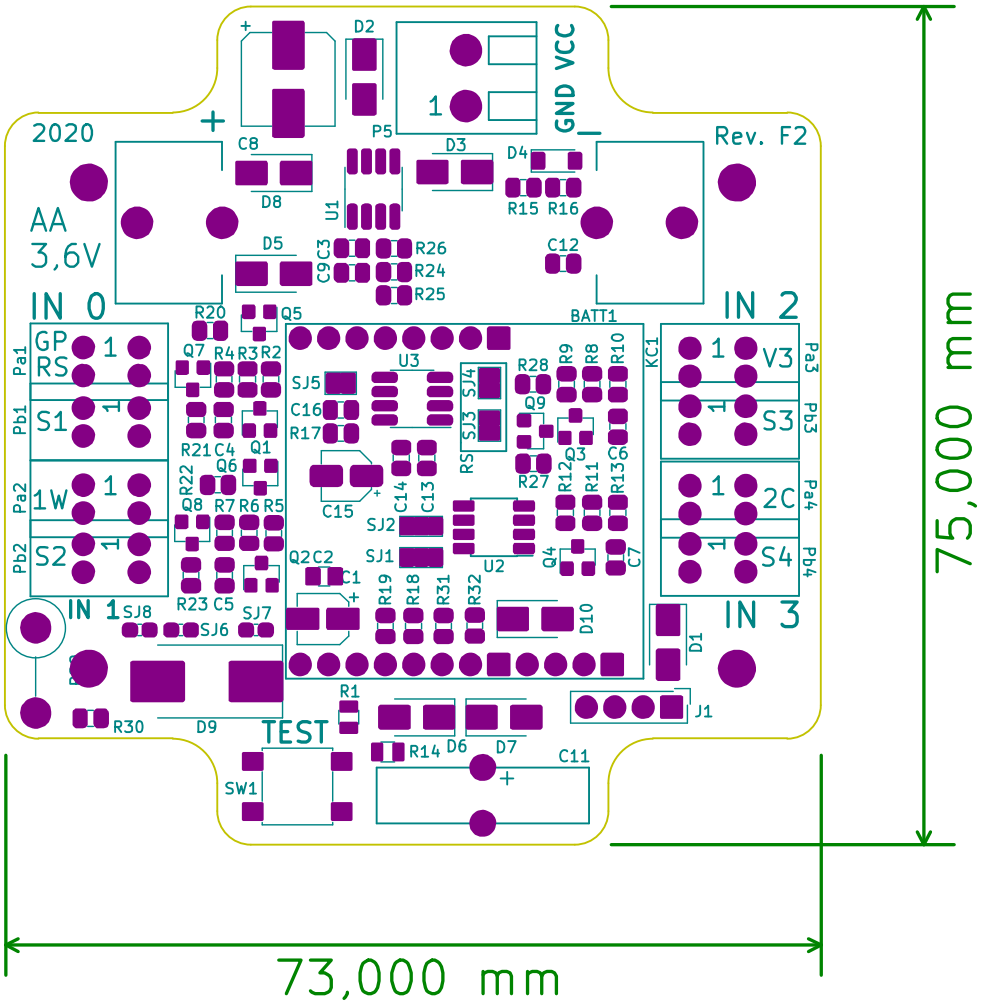
Example: 4xS0 Variant in the Standard Box Marking

LORATECH MultiSense S1S2S3S4TH-B(K)

4x S0 counter configuration in the Standard Box equiped with Relative humidity and temperature measurement. The device is powered from 3V6 battery.

3 Board Layout

Figure 1: Physical Layout of the *LORATECH MultiSense*



The placement and a brief description of connectors, headers and user-replaceable parts is provided in the following table:

Reference	Feature	Placement	Description
UART Debug (J1)	–	bottom edge, right	UART Debug and Configuration (KETCube Terminal)
SW1	–	left-bottom part	LORATECH MultiSense test button
KC1	–	center	KETCube mainBoard socket
BATT1	B	top center	Battery (LS 14500 is recommended)
EXT DC	V	top center	DC Power (8 – 25V)
IN0 (Pa1)	S0	left edge top	S0 Counter, counter #0
IN0 (Pb1)	RS	left edge top	ModBUS
IN1 (Pa2)	S1	left edge bottom	S0 Counter, counter #1
IN2 (Pa3)	S2	right edge top	S0 Counter, counter #2
IN2 (Pb3)	V3	right edge top	3V3 output
IN3 (Pa4)	S3	right edge bottom	S0 Counter, counter #3
IN3 (Pa4)	2C	right edge bottom	I2C interface

4 Firmware Upgrade

There are two ways to upgrade LORATECH MultiSense Firmware: (i) using the KETCube serial interface (UART) or (ii) using the SWD programmer.

4.1 Firmware Upgrade Through the KETCube Serial Interface

Prior to programming, connect the serial Cable to LORATECH MultiSense (described in Section 6).

If the cable is connected appropriately, the KETCube Terminal is available. To start the programming procedure, execute the following command in the KETCube Terminal:

```
>> set core startBootloader
```

KETCube is now in the *Bootloader Mode*: it's waiting for programmer connection. Now, close the serial terminal window on your PC to release the serial port and start the `STM32CubeProg`[1] tool. In terminal, execute the following command (replace `COM1` by name of the port used by LORATECH MultiSense):

```
$ ./STM32_Programmer.sh -c port=COM1 -w KETCube.bin 0x8000000
```

The response (in the case of success), should be similar to the output below:

```
   -------------------------------------------------------------------
                    STM32CubeProgrammer v2.4.0
   -------------------------------------------------------------------

Serial Port /dev/ttyUSB0 is successfully opened.
Port configuration: parity = even, baudrate = 115200, data-bit = 8,
                    stop-bit = 1,0, flow-control = off

Activating device: OK
Chip ID: 0x447
BootLoader protocol version: 3.1
Device name : STM32L07x/L08x/L010
Flash size  : 192 KBytes (default)
Device type : MCU
Device CPU  : Cortex-M0+

Memory Programming ...
Opening and parsing file: KETCube.bin
  File          : KETCube.bin
  Size          : 175240 Bytes
  Address       : 0x08000000


Erasing memory corresponding to segment 0:
Erasing internal memory sectors [0 1369]
Download in Progress:
[==================================================] 100%

File download complete
Time elapsed during download operation: 00:01:00.096
```

[1] `https://www.st.com/en/development-tools/stm32cubeprog.html`

4.2 Firmware Upgrade Through the SWD Programmer

Any STM32 devBoard (e.g. STM32 Nucleo Board[2]) or STM32-ready debug probe with SWD can be used to upgrade firmware.

The *STM32 ST LINK Utility*[3] is a handy tool for *LORATECH MultiSense* firmware upgrade.

4.2.1 Programming Connector – SWD

The KETCube board contains 1.27 SWD connector denoted H3.

H3 PIN	SWD Name	Description
1	VDD_TARGET	VDD from application
2	SWCLK	SWD clock
3	GND	Ground
4	SWDIO	SWD data in/out
5	NRST	Target MCU reset

4.2.2 Firmware Upgrade Procedure

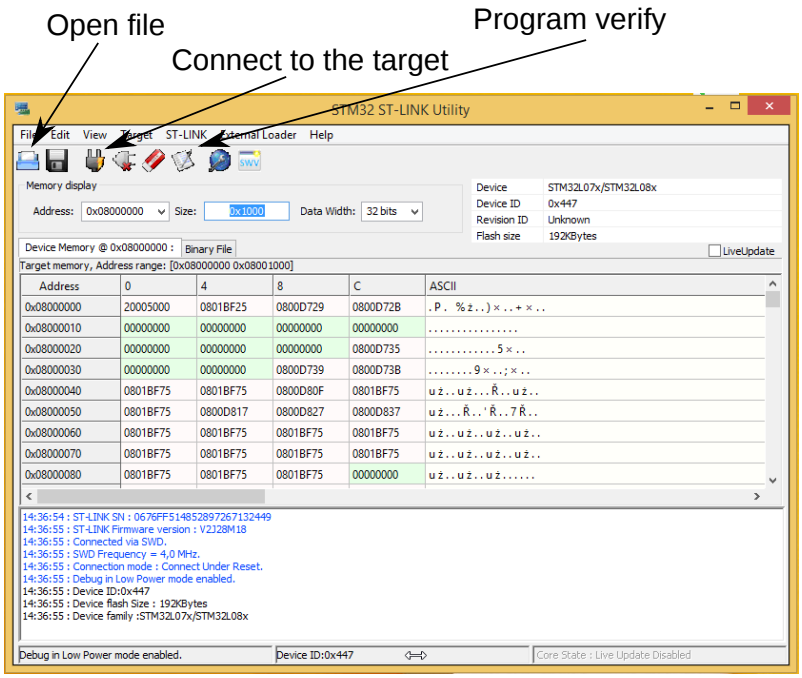


Figure 2: STM32 ST LINK Utility

1. Interconnect programmer (dev board) SWD with KETCube Programming Connector
2. Connect programmer (dev board) to PC
3. Run STM32 ST LINK Utility; the main window is in Figure 2
4. Click *Connect to the target* button – the device information should appear on the right side of main window and the device memory content should display in tabular part of the utility window

[2] https://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-eval-tools/stm32-mcu-eval-tools/stm32-mcu-nucleo/nucleo-f072rb.html
[3] https://www.st.com/en/development-tools/stsw-link004.html

5. Click *Open file* button – select supplied HEX file – the HEX file content should display in tabular part of the utility window
6. Click *Program verify* button – a small dialog window will display; here click *Start* button
7. Wait while programming is in progress

5 Operation Basics

LORATECH MultiSense is the modular device, which can be equipped with several sensors and actuators. It is able to communicate through LoRa network.

The device operation is driven by the *basePeriod* time value. This amount of time (configurable in milliseconds) determines the period between two sensing events. When this period elapses, *LORATECH MultiSense* performs all sensing-like actions (e.g. measure temperature, contact slaves to read -out data, ...) and transmits measurement results. The rest of the time, the device remains in low-power mode.

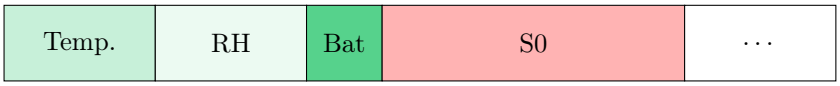
The *basePeriod* can be configured by `set core basePeriod` command (plus `reload` to apply configuration changes). The setting of *basePeriod* depends strictly on application and may vary from few seconds to several days.

The other configurable amount of time is *startDelay*. The *startDelay* determines the delay between the device power-on and the first sensing event (as defined by *basePeriod*).

The *startDelay* can be configured by `set core startDelay` command (plus `reload` to apply configuration changes). It is recommended to set *startDelay* to few seconds.

5.1 Sensor Data Transmission and Interpretation

As *LORATECH MultiSense* is modular, data from all available and enabled sensors are read sequentially. Thereafter sensor data are serialized given the module order, e.g:



The resulting frame is then transmitted using LoRa (if enabled).

To properly decode and interpret received data, you need to know:

- list of enabled modules and their order (can be obtained from command line – see Section 6)
- the module data length and format – see module configuration sections in this document

6 Debug and Configuration – LORATECH Terminal

When programmed by the supplied firmware, the *LORATECH Terminal* is available – see Figure 3.

Any 3V3 USB2Serial converter (e.g. *FTDI TTL-232R-3V3*) in connection with PC terminal program (e.g. Putty) can be used to configure and debug *LORATECH MultiSense*.

6.1 Connecting USB to Serial Converter

The USB to Serial Converter can be connected to *LORATECH MultiSense* on PIN header denoted `UART Debug (J1)` – see Section 3.

Connect TxD PIN of the *LORATECH MultiSense* to RxD of the *USB2Serial converter*, RxD of the *LORATECH MultiSense* to TxD of the *USB2Serial converter* and grounds (GND) – *Null-Modem*. Finally, connect the *USB2Serial converter* to PC.

If power supply is not provided from other source, you can power *LORATECH MultiSense* through 3V3 and GND PINs respectively.

!!! Remove other power sources before connecting power to 3V3 PIN! Note, that reverse polarity or over-voltage may damage the device !!!

See Absolute Maximum Ratings in Section 1.1.

6.2 LORATECH Terminal Settings

When physical connection is ready, configure your terminal program as follows:

- Speed: 9600 bps
- Data bits: 8
- Stop bits: 1
- Parity: No
- HW Flow control: No
- End-of-line: CR+LF or LF
- Port: depends on your system configuration (typically *COMx* for Windows, */dev/ttyUSBx* for Linux-based systems)

When configured, you can power-up *LORATECH MultiSense* and connect. You should see output similar to the Figure 3.

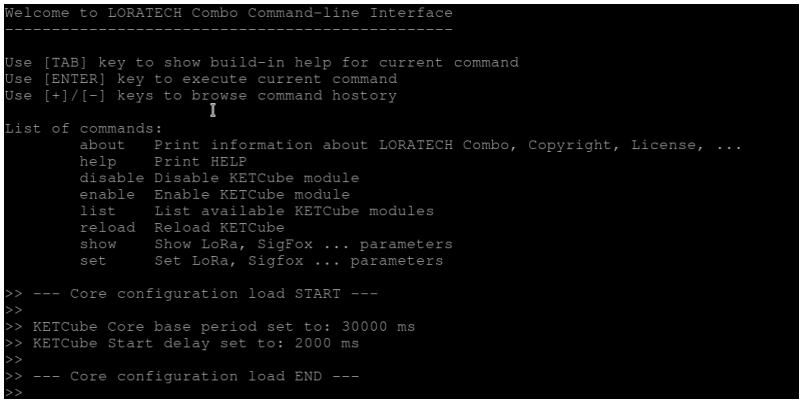


Figure 3: LORATECH Terminal in Putty terminal emulator

6.3 LORATECH TerminalBasics

LORATECH Terminal allows to enable/disable and configure *LORATECH MultiSense* modules (e.g. RHT Sensor (HDC1080), Battery Measurement, LoRa . . .) and module parameters (e.g. devEUI, appKEY, ... for LoRa module). The *LORATECH Terminal* is case-sensitive.

The LORATECH Terminal commands follow the hierarchical tree arrangement. The basic help including root commands is printed after device reset. The command `help` can be used anytime to display root commands.

Inline help is displayed when `[TAB]` key is pushed (e.g. write "s`[TAB]`" and all commands with leading "s" will be printed – these are: "set", "show", "setr" and "showr"). Inline help is useful especially for commands hidden deeply in the tree structure.

To display list of modules use `list` command. Commands `enable`/`disable` are used to turn ON/OFF modules (e.g. `enable HDC1080`). When module is enabled, it starts to perform defined operation (e.g. measure RH and Temperature and send results through LoRa).

Commands `showr`/`setr` are used to show/set *LORATECH MultiSense* or module RUNNING settings (e.g. `showr LoRa devEUIBoard`). The settings configured by the "setr" command are lost when device is reloaded. Commands "showr"/"setr" are usefull for device configuration testing.

Commands `show`/`set` are used to show/set *LORATECH MultiSense* or module PERSISTENT settings (e.g. `show LoRa devEUIBoard`). Settings are preserved in the on-chip EEPROM. The persistent settings are loaded during device start-up sequence. Use command `reload` to apply persistent settings.

The command history is available through `+` and `-` keys.

6.4 Debugging

Debug messages are useful when device is initially configured or in case, that unexpected behaviour occurred.

Debug messages are printed to serial terminal interface. They can be configured on the per-module basis by setting the *severity level* to the selected module or to the KETCube core.

Four severity levels are defined:

0 – NONE – no messages enabled

1 – ERROR – only error messages enabled (default severity)

2 – INFO – error and informational messages enabled

3 – DEBUG – previous message groups and debug information are enabled

The second (optional) parameter of the "enable" command is used to configure the *severity level* of the selected module (e.g. `enable HDC1080 3` enables debug messages for HDC1080 module). The command `set core severity` is used to configure *severity level* of the KETCube core and command `set driver severity` is used to configure *severity level* of the KETCube low-level drivers.

Note that when debug messages are produced, it may be difficult to write commands (terminal echo is foggy due to lot of debug messages produced by *LORATECH MultiSense*).

7 LoRa Configuration

Most of the LoRaWAN parameters can be set by using the *LORATECH Terminal* interface. The *LORATECH Terminal* basics are described in Section 6.

Write command `show LoRa` or `set LoRa` to display commands related to LoRa module:

```
>> show LoRa
appEUI               LoRa application EUI
appKey               LoRa application key
appSKey              LoRa app session Key
devAddr              LoRa device address
devEUIBoard          Board (boardID-based) LoRa device EUI
devEUICustom         Custom LoRa device EUI
enableABP            Enable ABP
enableCustomDevEUI   Custom (user-defined) LoRa device EUI
nwkSKey              LoRa network session Key

>> set LoRa
appEUI               LoRa application EUI
appKey               LoRa application key
appSKey              LoRa app session Key
devAddr              LoRa device address
devEUICustom         Custom LoRa device EUI
enableABP            Enable ABP
enableCustomDevEUI   Custom (user-defined) LoRa device EUI
nwkSKey              LoRa network session Key
```

7.1 LoRaWAN Node Activation

To select or check OTAA/ABP mode settings, the following commands are used: `set LoRa enableABP 0|1`/`show LoRa enableABP`. The "set" command parameter is boolean `0` or `1`. OTAA is the default mode.

7.2 LoRaWAN devEUI

User-defined devEUI (`set LoRa enableCustomDevEUI 1` or manufacturer's devEUI (`set LoRa enableCustomDevEUI 0`) can be used. The selected option can be checked via `show LoRa enableCustomDevEUI` command. The user-defined devEUI can be configured by `set LoRa devEUICustom` command. Manufacturer's devEUI is used by default.

Example: Set user-defined devEUI

```
>> set LoRa devEUICustom 1122334455667788
>> set LoRa enableCustomDevEUI 1
>> reload

>> show LoRa devEUICustom

Custom devEUI is displayed ...
```

Example: Use Manufacturer's devEUI

```
>> set LoRa enableCustomDevEUI 0
>> reload

>> show LoRa devEUIBoard

Board devEUI is displayed ...
```

7.3 LoRaWAN OTAA Parameters

The LoRaWAN appEUI and appKey can be set by using `set LoRa appEUI` and `set LoRa appKey` commands and checked by using `show LoRa appEUI` and `show LoRa appKey` commands.

Example: Set OTAA parameters

```
>> set LoRa appEUI 1122334455667788
>> set LoRa appKey 11223344556677881122334455667788
```

7.4 LoRaWAN ABP Parameters

The LoRaWAN appSKey and nwkSKey can be set by using `set LoRa appSKey` and `set LoRa nwkSKey` commands and checked by using `show LoRa appSKey` and `show LoRa nwkSKey` commands. The static device address i configured via `set LoRa devAddr` command and checked via `show LoRa devAddr` command.

Example: Set ABP parameters

```
>> set LoRa devAddr DEADBEEF
>> set LoRa appSKey 1122334455667788
>> set LoRa nwkSKey 11223344556677881122334455667788
```

7.5 LoRaWAN Period

The period of data transmission is given by the *basePeriod* – see Section 5.

7.6 LoRaWAN Ports

The periodical messages are transmitted by using LoRaWAN port 2. Asynchronous messages (particular modules can indicate errors asynchronously, push-button event) are transmitted by using LoRaWAN port 3 if and only if the *AsyncTx* module is enabled.

8 RHT Sensor (HDC1080) Configuration

RHT Sensor (HDC1080) is on-board *Relative Humidity* and *Temperature* sensor allowing RH+T measurement.

8.1 RHT Sensor (HDC1080) Module Configuration

RHT Sensor (HDC1080) module can be enabled in the same way as any other module – see Section 6.3.

The sample configuration procedure:

```
>> enable HDC1080
Executing command: enable

Setting module HDC1080:     success!
>> reload
Executing command: reload

Performing system reset and reloading KETCube configuration ...
```

Once reloaded, the *RHT Sensor (HDC1080)* module measures RH+T periodically.

For debugging purposes, set the severity level, as described in Section 6.4. The sample output for the severity level 2 (INFO) is shown below:

```
>> HDC1080 :: Temperature: 25, RH: 57
```

8.2 RHT Sensor (HDC1080) Module Output

The *RHT Sensor (HDC1080)* module output consist of 4 bytes:

Temperature is encoded in BigEndian as unsigned integer in additive code. The value units are degrees of Celsius.

To compute actual temperature, use the following equation:

$T = ((T_{MSB} << 8 \mid T_{LSB}) - 10000)/10\ [°C]$

Relative Humidity is encoded in BigEndian format as unsigned integer in additive code. The value units are % RH.

To compute actual RH, use the following equation:

$RH = (RH_{MSB} << 8 \mid RH_{LSB})/10\ [\%]$

An error for both temperature and humidity is indicated by out-of-the range values from the unsigned word (16-bit) range – see ranges in Section 1.2.

9 ModBUS Configuration

ModBUS module allows to use *LORATECH MultiSense* as a MODBUS Master device.

9.1 Physical Connection

To get it work, connect ModBUS slave to connector denoted IN0 (Pb1).

PIN#1 is RS485 signal B, PIN#2 is RS485 signal A.

9.2 ModBUS Module Configuration

ModBUS module is configured the same way as any other module – see Section 6.3. Basic transmission parameters can be configured by using the following commands: `baudrate`, `challenge address`, `challenge coil`, `challenge discreteinput`, `challenge inputregister` and `challenge holdingregister`. These commands define the communication speed and slave address as well as the set of commands, which will be used to read selected entities from connected ModBUS slave.

The slave response is always serialized in this order:

1. input registers
2. holding registers
3. coils (padded to bytes)
4. discrete inputs (padded to bytes)

ModBUS reads a list of configured entities and serializes their values into a response. Unlike other modules, where we may operate with cutting the response by bytes, we serialize the response data automatically. Input and holding registers are 2 bytes long, so they're put into a response as-is, but coils and discrete inputs are always padded to one byte, e.g. when only 2 coils are requested, another 6 zero bits are appended to fill the whole byte. Coils and discrete inputs are padded independently, as well, as every part of the list. If e.g. two coil segments are read, both of them should be padded to one byte.

When specifying list of entities to be read, values are delimited by comma. In given entity configuration, `start` list and `count` the list must have the same amount of items.

The sample configuration procedure:

```
>> set modbus baudrate 19200
Executing command: baudrate

Write success!
>> set modbus challenge address 01
Executing command: address

Write success!
>> set modbus challenge coil start 10,20
Executing command: start

Write success!
>> set modbus challenge coil count 02,05
Executing command: count
```

```
Write success!
>> set modbus challenge inputregister start 10
Executing command: start

Write success!
>> set modbus challenge inputregister count 1F
Executing command: count

Write success!
>> enable ModBUS
Executing command: enable

Setting module ModBUS:     success!

>> reload
Executing command: reload

Performing system reset and reloading KETCube configuration ...
```

The reading of given entity could be turned off by simply setting 00 to start and count.

When reloaded, the *ModBUS* module starts to read MODBUS slave with defined address periodically. Note, that when MODBUS reading is performed, the *LORATECH Terminal* will stuck temporarily.

For debugging purposes, set the severity level, as described in Section 6.4. The sample output for the severity level 2 (INFO) is shown below:

```
>> MODBUS :: Tx data: DATA=01-01-00-10-00-02-BC-0E;
>> MODBUS :: Rx data: 01-01-01-03-11-89;
>> MODBUS CRC (8911) == rxCRC (8911)
>> MODBUS :: Tx data: DATA=01-01-00-20-00-02-BC-01;
>> MODBUS :: Rx data: 01-01-01-00-51-88;
>> MODBUS CRC (8851) == rxCRC (8851)
```

9.3 ModBUS Module Output

For error-free transaction, the ModBUS module output consist of 1 byte indicating the response length followed by the bytes of serialized responses, as shown below:

0x0A	0x15	0xF0	0x54	0x6C	0x19	0xB8	0x00	0x47	0x6C	0x23

If the leading byte is zero, it is followed by another one indicating error:

0x00	ERR.

Following error codes are defined:

- 0x01 – invalid function (request code)
- 0x02 – invalid data address
- 0x03 – invalid data value
- 0x04 – slave device failure
- 0x06 – slave device is busy

- 0x08 – slave detected parity error in its memory
- 0x0A – the slave replied with invalid address
- 0x0B – communication timeout
- 0x0C – the slave replied with unknown or unexpected code
- 0x0D – CRC failure
- 0x0E – the length of received data didn't match expected length

10 S0 Counter Configuration

S0 Counter module allows to use *LORATECH MultiSense* as a S0 pulse input device [3].

10.1 Physical Connection

Up to 4 S0 output devices can be connected to *LORATECH MultiSense*. One Ultra Low-Power input (S0), one Low-Power input (S1) and additional 2 inputs (S2 and S3) are available. Positive inputs are denoted as A+, negative as B- respectively. Lower S0 inputs should be preferred if less than 4 inputs are required.

Note, that the Ultra Low-Power input provides lowest power consumption and allows to count faster pulses than allowed by EN 62053-31:1998 [3]. Thus it may be useful for faster signal counting, but this input is also more noise sensitive, thus it should not be used in noisy enviroments.

10.1.1 S1 Connection

Connector is marked as IN0 (Pa1). PIN 2 is A+, PIN 1 is B-.

10.1.2 S2 Connection

Connector is marked as IN1 (Pa2). PIN 2 is A+, PIN 1 is B-.

10.1.3 S3 Connection

Connector is marked as IN2 (Pa3). PIN 2 is A+, PIN 1 is B-.

10.1.4 S4 Connection

Connector is marked as IN3 (Pa4). PIN 2 is A+, PIN 1 is B-.

10.2 S0 CounterModule Configuration

S0 Counter module is configured the same way as any other modules – see Section 6.3.

Each counter activation/deactivation is performed by using the following command: `OnX 0|1`, where `X` is the counter ID and the second parameter is boolean value setting the counter ON or OFF (e.g. `set S0 On0 1`).

LORATECH MultiSense allows to monitor the counter activity: the LoRa message is send in case of long inactivity. The inactivity period is configured by using the `timeoutX` command (`X` is the counter ID). The timeout accuracy is 1 minute. For timeout deactivation set timeout to 0 (e.g. `set S0 timeout0 0`).

The timeout requires *AsyncTx* module: if *AsyncTx* module is not active, the LoRaWAN message will not be transmitted even though the timeout will be detected by *S0 Counter* module (and displayed in *LORATECH Terminal* deppending on the severity level).

The counter value of timers can be checked from LORATECH Terminal by using the `show S0 valueX` command (`X` is the counter ID). The `set S0 valueX` command allows to set the default counter value (after reset).

The sample configuration procedure:

```
>> set S0 On0 1
Executing command: On0
Command execution OK
On0 returned: TRUE

>> set S0 timeout0 10
Executing command: timeout0
Command execution OK
timeout0 returned: 10

>> enable AsyncTx
Executing command: enable
Command execution OK

>> enable S0
Executing command: enable
Command execution OK

>> reload
Executing command: reload

Performing system reset and reloading KETCube configuration ...
```

When reloaded, the *S0 Counter* module starts to count events on active S0 counter lines.

The current configuration and counter states can be checked by using following commands: `OnOff`, `Timeout` and `Value`. The sample command sequence is given below:

```
>> show S0 OnOff
Executing command: OnOff
Counter 0 ON
Counter 1 OFF
Counter 2 ON
Counter 3 OFF

>> show S0 timeout0
Executing command: timeout0
Command execution OK
timeout0 returned: 10
>>
>> show S0 value0
Executing command: value0
Command execution OK
value0 returned: 155
>>
```

10.3 S0 Counter Module Output

The S0 Counter module produces the periodical messages. These messages are transmitted by using LoRaWAN port 2. The periodical message contains chained counter states – 4 bytes for each active counter. An example for 2 active counters (0 and 2) is shown below:

If no pulse is received during defined timeout and *AsyncTx* module is enabled, the asynchronous error message (2 bytes) is transmitted by using LoRaWAN port 3. An example error message is shown below:

The first byte – ID – indicates the module of asynchronous message origin. In this particular case, it is the ID of the *S0 Counter* module[4]. The second byte – ERR. – represents the error code itself.

Following error codes are defined:

- 0x00 – counter 0 timeout
- 0x01 – counter 1 timeout
- 0x02 – counter 2 timeout
- 0x03 – counter 3 timeout

[4] To get module ID, use command `list`: first module in the list – LoRa – has ID 0, the second module has ID 1, ...

11 Brownout Detection

Brownout Detection module allows to save volatile values in case of power loss. This is useful for devices powered by adapter or solar power source without battery backup.

When enabled, S0 counter values are backed-up in case of loss of external power. When powered-up again, the values are restored and are used to initialize the respective counters.

11.1 Brownout Detection Module Configuration

Brownout Detection module can be enabled in the same way as any other module – see Section 6.3.

The sample configuration procedure:

```
>> enable powerDownBackup
Executing command: enable

Setting module powerDownBackup:     success!
>> reload
Executing command: reload

Performing system reset and reloading KETCube configuration ...
```

Once reloaded, the *Brownout Detection* module protects the S0 volatile values.

For debugging purposes, set the severity level, as described in Section 6.4. The sample output for the severity level 2 (INFO) is shown below:

```
>> powerDownBackup :: BACKUP START!
>> powerDownBackup :: BACKUP DONE!
```

12 Maxbotix MaxSonar MB7040

Maxbotix MaxSonar MB7040 module allows to use *LORATECH MultiSense* as a ultrasound (sonar) distance metering device.

13 Test Button Configuration

Test Button module allows to transmit an asynchronous LoRa message. The LORATECH MultiSense test button is marked as `SW1`.

13.1 Test Button Module Configuration

Test Button module has no special configuration commands. It can be either enabled or disabled by using `enable` and `disable` commands respectively – see Section 6.3 for details.

The sample configuration procedure:

```
>> enable testButton
Executing command: enable

Setting module testButton:     success!
>> reload
Executing command: reload

Performing system reset and reloading KETCube configuration ...
```

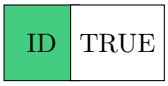
When reloaded, the *Test Button* module reacts to the push button events.

For debugging purposes, it may be useful to watch *Test Button* output in console. If required, activate the `DebugDisplay` module as described in Section 6.4. The debug output, related to *Test Button* module, should look similarly:

```
...
>> PushButton :: Push event
>> Module "AsyncTx" ProcessData()
>> AsyncSend :: from module = 10
...
```

13.2 Test Button Module Output

The *Test Button* module produces no periodical messages. If the test button is pushed and *AsyncTx* module is enabled, the asynchronous message (2 bytes) is transmitted by using LoRaWAN port 3. An example message is shown below:



The first byte – ID – indicates the module of asynchronous message origin. In this particular case, it is the ID of the *Test Button* module[5]. The second byte – TRUE – represents the logic *true* value (represented by *1*).

[5] To get module ID, use command `list`: first module in the list – LoRa – has ID 0, the second module has ID 1, ...

References

[1] University of West Bohemia in Pilsen, "KETCube Datasheet," 2018, -. [Online]. Available: https://github.com/SmartCAMPUSZCU/KETCube-docs

[2] Texas Instruments, "HDC1080 Datasheet," 2016, -. [Online]. Available: http://www.ti.com/lit/ds/symlink/hdc1080.pdf

[3] "IEC 62053-31:1998: Electricity metering equipment (a.c.) - Particular requirements - Part 31: Pulse output devices for electromechanical and electronic meters (two wires only)," TC 13 - Electrical energy measurement and control Committee, Standard, Jan. 1998.

[4] Saft Specialty Battery Group, "LS 14500 Primary Li-SOCl2 cell: High energy density 3.6 V AA-size bobbin cell," 2019, -. [Online]. Available: https://www.saftbatteries.com/

KETCube Platform License Information

Important Notice

LORATECH MultiSense is the commercial and proprietary extension of KETCube platform developed by RVTech s r.o.

LORATECH MultiSense is NOT developed or supported in any form by the University of West Bohemia in Pilsen.

Copyright

Developed by

The SmartCAMPUS Team
Department of Technologies and Measurement
Faculty of Electrical Engineering
www.smartcampus.cz/en | www.zcu.cz/en

License[6]

[6] *University of Illinois/NCSA Open Source License* (`https://opensource.org/licenses/NCSA`) – similar to *Modified BSD License*

LORATECH MultiSense Legal Notice

Copyright

Important Notice

RVTech s.r.o. (owner of LORATECH trademark) points out that all information in this document is given on an "as is" basis. No guarantee, neither explicit nor implicit is given for the correctness at the time of publication. RVTech s.r.o. reserves all rights to make corrections, modifications, enhancements, and other changes to its products and services at any time and to discontinue any product or service without prior notice. It is recommended for customers to refer to the latest relevant information before placing orders and to verify that such information is current and complete.

RVTech s.r.o. assumes no liability for the use of its products and does not grant any licenses for its patent rights or for any other of its intellectual property rights or third-party rights. It is the customer's duty to bear responsibility for compliance of systems or units in which products from RVTech s.r.o. are integrated with applicable legal regulations. Customers should provide adequate design and operating safeguards to minimize the risks associated with customer products and applications. The products are not approved for use in life supporting systems or other systems whose malfunction could result in personal injury to the user. Customers using the products within such applications do so at their own risk.

Any reproduction of information in datasheets of RVTech s.r.o. is permissible only if reproduction is without alteration and is accompanied by all given associated warranties, conditions, limitations, and notices. Any resale of RVTech s.r.o. products or services with statements different from or beyond the parameters stated by RVTech s.r.o. for that product/solution or service is not allowed and voids all express and any implied warranties.