LORATECH Combo MBRSS4VMTH-BV Manual (draft)

Author: Jan Bělohoubek (JB), Martin Úbl (MU)

Version draft from 6.10.2018

General Description

LORATECH Combo is the semi-universal sensing and remote metering platform supplied by RVTech s r.o.

LORATECH Combo 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 Combo MBRSS4VMTH-BV* device.

Main Features

- Communication: LoRaWAN Class A device
- Debug: Featured LORATECH Terminal
- HDC1080 RHT Sensor : Relative Humidity & Temperature
- MBUS : up to 3 slaves
- MODBUS : configurable ModBUS RTU master device
- S0 Counter : pulse counter (up to 4 devices)
- ADC : up to 100V continuous measurement
- Extensible: compatible with KETCube [1]
- Power supply:
  - 3V6 battery: LS 33600 (recommended battery)
  - 4 – 100V DC
  - 5V USB IN (optional assembly – on request only)

Contents

List of Figures

List of Tables

Revision History

Revision	**Date**	**Author**	**Note**
draft	6.10.2018	JB, MU	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
Supply Voltage (USB)	5V	-0.3	5	6	V
High Supply Voltage (SL1, SL5, SL6)	100V	-0.3	5 – 100	110	V
Supply Voltage (SL4)	3V3	-0.3	–	3.9	V
Storage Temperature		-40	25	90	°C
Storage Humidity		20	–	70	%RH
Input RF Level		–	–	10	dBm

1.2 Operating Conditions

LORATECH Combo Core Parts					
Parameters	**Cond.**	**MIN**	**TYP**	**MAX**	**UNIT**
Supply Voltage (USB)	USB charger or PC	-0.3	5	5.5	V
High Supply Voltage (SL1, SL5, SL6)	DC min. 200mW	-110	5 – 100	100	V
Supply Voltage (SL4, SL18)	LS 33600	2.2	3.6	3.65	V
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

HDC1080 RHT Sensor [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

ADC [3]					
Parameters	**Cond.**	**MIN**	**TYP**	**MAX**	**UNIT**
High Voltage (SL1)	100V	-110	1 – 100	110	V
Voltage Measurement Accuracy		–	± 0.5	± 1	V

MBUS					
Parameters	**Cond.**	**MIN**	**TYP**	**MAX**	**UNIT**
Voltage at 3V3 pin	at 20°C; no load	2.9 [a]	–	3.9	V
Input current	at 3.3V; with 1 MBUS slave; at 36V	–	< 90	–	mA
Output current	1 – 3 MBUS slaves; at 36V	–	15 – 20	50	mA
Device read time (incl. power-on)	1 MBUS slave	–	5.1	6.8	s

[a] Avoid operation at low temperatures (below 0°C), when used with battery – low temperature voltage drop may cause MBUS malfunction.

S0 Counter					
Parameters	**Cond.**	**MIN**	**TYP**	**MAX**	**UNIT**
Open Circuit Voltage	–	–	3.3	–	V
Short Circuit Current	S0 Inter-PIN resistance ≤ 1 Ω	–	12	–	mA
On Resistance (S0_0, S0_1, S0_2)	S0_3 NOT used	–	0 – 1k	–	Ω
On Resistance (S0_2, S0_3)	S0_3 used	–	0 – 500	–	Ω
Off Resistance	–	–	≥ 1M	–	Ω
S0 switching frequency	–	–	14[a]	30	Hz

[a] See the European Standard EN 62053-31:1998 [4]

Recommended Battery LS 33600 [5]					
Parameters	**Cond.**	**MIN**	**TYP**	**MAX**	**UNIT**
Nominal capacity	at 5 mA, +20 °C, 2.0V	–	17.0	–	Ah
Open circuit voltage	+20 °C	–	3.67	–	V
Nominal voltage	+20 °C	–	3.6	–	V
Nominal energy	+20 °C	–	61.2	–	Wh
Maximum recommended continuous current		–	–	250	mA
Operating temperature range		-60	–	+85	°C

2 Variant Naming

LORATECH Combo is supplied in number of variants depending on enabled features. The device name consist of the product family name "*LORATECH Combo* " and the device-specific string expressing included features:

LORATECH COMBO AA[BBCCDDEE]-X[XX](Y)

The AA, BB, CC, DD or EE marking encodes the available features, available power sources and packaging as described in the following table:

#	Marking	Feature
1	MB	MBUS
2	RS	MODBUS RTU (over RS485)
3	S1, S2, S3, S4	S0 Counter input – 1 to 4 counters
4	VM	ADC (voltage measurement – up to 100V)
5	TH	HDC1080 RHT Sensor

The X, XX or XXX marking encodes the available power source options as described in the following table:

#	Marking	Feature
1	B	Battery (LS 33600 is recommended)
2	V	External DC power source or battery – 4 to 100V DC
3	S	Solar power source

The (Y) marking encodes the LORATECH Combo packaging as described in the following table:

#	Marking	Feature
2	K	Standard box
1	D	Standard DIN-compatible box
3	S	Standard box with an integrated DIN adapter

2.1 Example: Full-Featured Variant Marking

LORATECH COMBO MBRSS4VMTH-BV(K)

Full-featured configuration in the Standard Box. The device can be powered from battery or from an external DC power source.

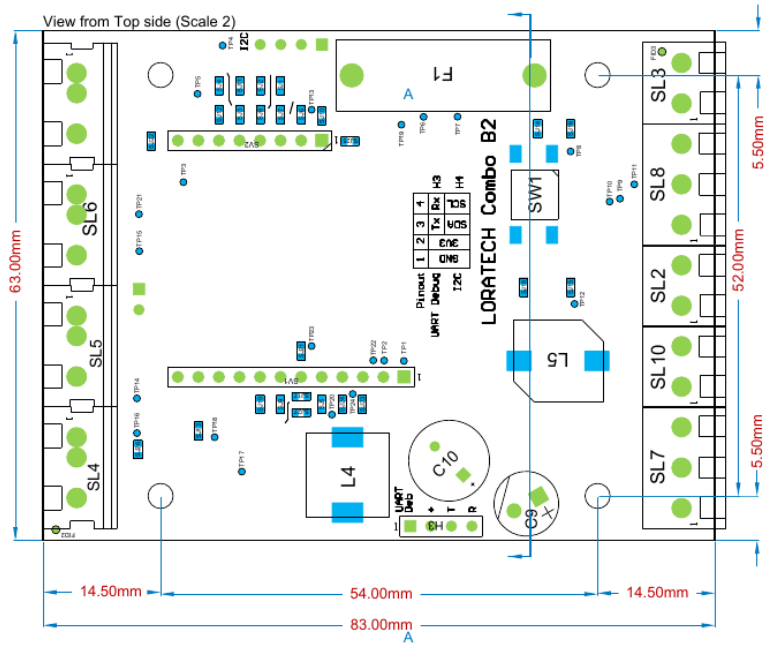
2.2 Example: MBUS Variant Marking

LORATECH COMBO MB-B(D)

MBUS-only configuration in the DIN-compatible Box. The device can be powered from battery battery.

3 Board Layout

Figure 1: Physical Layout of the *LORATECH Combo*



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

Connector/Header	Placement	Description
UART Deb (H3)	bottom edge, center	UART Debug and Configuration
SW1	right-top part	LORATECH Combo test button
SV1, SV2	left central part	KETCube mainBoard socket
F1	top edge, center	MBUS 50mA fuse
SL1	left edge	ADC and optionally DC Power (0 – 100V) common input
SL2	right edge	S0 Counter , counter #0
SL3	right edge	MBUS BUS
SL4	left edge	Battery (LS 33600 is recommended)
SL6	left edge	DC Power (0 – 100V)
SL7	right edge	S0 Counter , counter #2 and #3
SL8	right edge	MODBUS
SL10	right edge	S0 Counter , counter #1

4 Firmware Upgrade

Any STM32 devBoard (e.g. STM32 Nucleo Board[1]) or STM32-ready debug probe with SWD can be used to upgrade firmware.

The *STM32 ST LINK Utility*[2] is a handy tool for *LORATECH Combo* firmware upgrade.

4.1 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 Firmware Upgrade Procedure

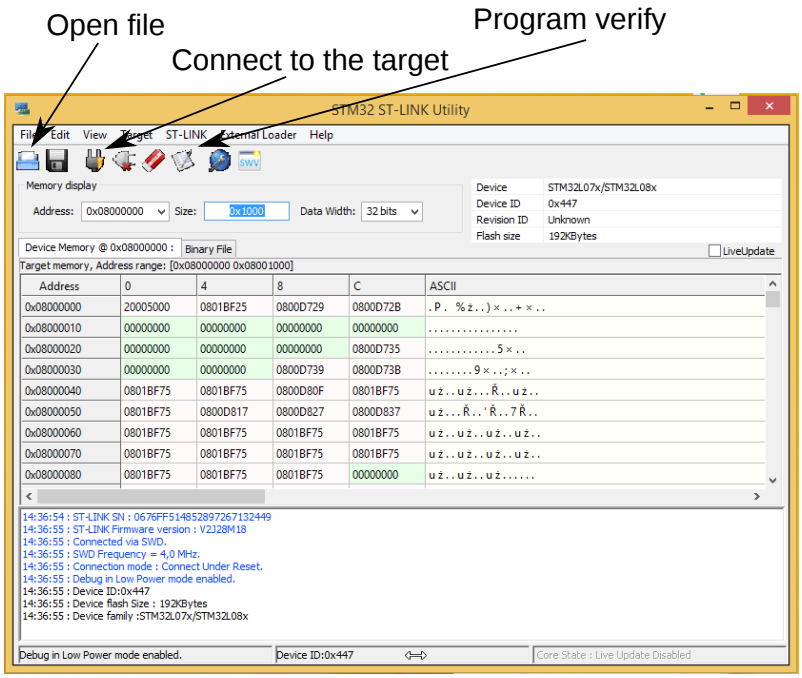


Figure 2: STM32 ST LINK Utility

- Interconnect programmer (dev board) SWD with KETCube Programming Connector
- Connect programmer (dev board) to PC
- Run STM32 ST LINK Utility; the main window is in Figure 2
- 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

[1] 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

[2] https://www.st.com/en/development-tools/stsw-link004.html

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

5 Operation Basics

LORATECH Combo 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. This amount of time (configurable in milliseconds) determines the period between two sensing events. When this period elapses, *LORATECH Combo* 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.

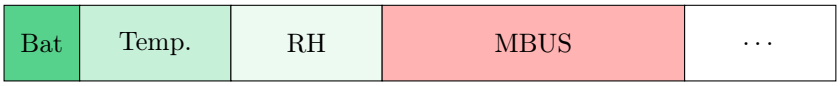
The *basePeriod* can be configured by `set core basePeriod` command (plus `reload` to apply configuration changes). The setting of *basePeriod* depends strictly on application and may vary from few seconds to several days.

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 Combo* 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 Combo* .

6.1 Connecting USB to Serial Converter

The USB to Serial Converter can be connected to *LORATECH Combo* on PIN header denoted `UART Deb (H3)` – see Section 3.

Connect TxD PIN of the *LORATECH Combo* to RxD of the *USB2Serial converter*, RxD of the *LORATECH Combo* 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 external source, you can power *LORATECH Combo* through 3V3 and GND PINs respectively. **Remove all other power sources before connecting power to 3V3 PIN! Note, that reverse polarity or over-voltage can 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 configuration (typically *COMx* for Windows, */dev/ttyUSBx* for Linux-based systems)

When configured, you can power-up *LORATECH Combo* and connect. You should see output similar to the Figure3.

Figure 3: LORATECH Terminal in Putty

```
Welcome to LORATECH Combo Command-line Interface
------------------------------------------------

Use [TAB] key to show build-in help for current command
Use [ENTER] key to execute current command
Use [+]/[-] keys to browse command hostory
                    I
List of commands:
       about   Print information about LORATECH Combo, Copyright, License, ...
       help    Print HELP
       disable Disable KETCube module
       enable  Enable KETCube module
       list    List available KETCube modules
       reload  Reload KETCube
       show    Show LoRa, SigFox ... parameters
       set     Set LoRa, Sigfox ... parameters

>> --- Core configuration load START ---
>>
>> KETCube Core base period set to: 30000 ms
>> KETCube Start delay set to: 2000 ms
>>
>> --- Core configuration load END ---
>>
```

6.3 Basic LORATECH Terminal Features

LORATECH Terminal allows to enable/disable and configure *LORATECH Combo* modules (e.g. HDC1080 RHT Sensor , batMeas , 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" and "show"). 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 RHT Sensor`). When module is enabled, it starts to perform defined operation (e.g. measure RH and Temperature and send results through LoRa).

Commands `show`/`set` are used to show/set *LORATECH Combo* or module settings (e.g. `show LoRa devEUI`). Settings are preserved in the on-chip EEPROM.

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

All settings are applied after device reset (use command `reload`).

6.4 Debugging

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

Debug messages can be switched on by typing `enable DebugDisplay` (and `reload`). To disable DebugDisplay, write `disable DebugDisplay` (and `reload`).

Note that when DebugDisplay is active it may be difficult to write commands (terminal echo is foggy due to lot of debug messages produced by *LORATECH Combo*) – you have to write without watching echo on the screen.

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
ABP           Is ABP enabled?
OTAA          Is OTAA enabled?
appEUI        Show LoRa application EUI.
appKey        Show LoRa application key.
appSKey       Show LoRa app session Key
devAddr       Show LoRa device address.
devEUI        Show LoRa device EUI.
devEUIType    Show LoRa device EUI type: custom (user-defined)
              or deviceID-based.
nwkSKey       Show LoRa network session Key.

>> set LoRa
ABP           Enable ABP.
OTAA          Enable OTAA.
appEUI        Set LoRa application EUI.
appKey        Set LoRa application key.
appSKey       Set LoRa app session Key
devAddr       Set LoRa device address.
devEUI        Set LoRa device EUI.
devEUICustom  Use custom (user-defined) LoRa device EUI
devEUIBoard   Use board (boardID-based) LoRa device EUI
nwkSKey       Set LoRa network session Key.
```

7.1 LoRaWAN Node Activation

To set OTAA/ABP mode or show which mode is enabled/disabled, use following commands: `show LoRa OTAA`/`show LoRa ABP` and `set LoRa OTAA`/`set LoRa ABP`.

7.2 LoRaWAN devEUI

User-defined devEUI (`set LoRa devEUICustom` and `set LoRa devEUI`) or manufacturer's devEUI (`set LoRa devEUIBoard`) can be used. The selected option can be checked via `show LoRa devEUIType` and `show LoRa devEUI` commands respectively.

Example: Set user-defined devEUI

```
>> set LoRa devEUI 1122334455667788
>> set LoRa devEUICustom
>> reload

>> show LoRa devEUI

Custom devEUI is displayed ...
```

Example: Use Manufacturer's devEUI

```
>> set LoRa devEUIBoard
>> reload

>> show LoRa devEUI

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 batMeas Configuration

batMeas module allows to monitor *LORATECH Combo MBRSS4VMTH-BV* battery.

8.1 batMeas Module Configuration

batMeas module can be enabled the same way as any other module – see Section 6.3.

Use the commands `list` and `bat` to select connected battery.

The sample configuration procedure:

```
>> show batMeas list
Executing command: list
Available batteries:
0)       CR2032 (Up to 560mAh battery)
1)       LS33600 (15 Ah battery)
>> set batMeas bat 1
Executing command: bat

Write success!

>> enable BatMeas
Executing command: enable

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

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

Once reloaded, the *batMeas* module measures battery voltage periodically.

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

```
>> Module "batMeas" GetSensorData()
>> batLevel :: 0
```

8.2 batMeas Module Output

The *batMeas* module output consist of a single byte only.

The *batMeas* output value (B) meaning is as follows as follows:

- 0x00: battery is (almost) discharged
- 0x01 – 0xFE: battery voltage scaled to 1 – 254
- 0xFF: RFU

To compute actual battery voltage (for recommended battery), use the following equation:

$V = (B/254 \cdot 0.7) + 2.9\ [mV]$

9 HDC1080 RHT Sensor Configuration

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

9.1 HDC1080 RHT Sensor Module Configuration

HDC1080 RHT Sensor module can be enabled 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 *HDC1080 RHT Sensor* module measures RH+T periodically.

For debugging purposes, it may be useful to watch RH+T output in console. If required, activate the `DebugDisplay` module as described in Section 6.4. The debug output, related to *HDC1080 RHT Sensor* module, should look similarly:

```
>> Module "HDC1080" GetSensorData()
>> HDC1080 :: Temperature: 25, RH: 57
```

9.2 HDC1080 RHT Sensor Module Output

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

T_{MSB}	T_{LSB}	RH_{MSB}	RH_{LSB}

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.

10 ADC Configuration

ADC module allows to monitor 1 – 100V external DC Voltage.

10.1 Physical Connection

LORATECH Combo ADC is designed to allow connection of up to 100V DC voltage source.

To get it work, connect monitored DC voltage source to connector denoted SL1.

The input is protected against wrong polarity – the device will not be destroyed, but measurement is not possible.

10.2 ADC Module Configuration

ADC module can be enabled the same way as any other module – see Section 6.3.

The sample configuration procedure:

```
>> enable ADC
Executing command: enable

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

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

Once reloaded, the *ADC* module measures connected voltage periodically.

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

```
>> Module "ADC" GetSensorData()
>> ADC :: Voltage@PA4: 133 mV
```

10.3 ADC Module Output

The *ADC* module output consist of one two-byte word (16-bit unsigned int):

Measured value represents the fraction of the measured Voltage. It is encoded in BigEndian format as unsigned integer.

To compute actual DC voltage connected to *ADC* input, use the following equation:

$V = 0,6 + U_{ADC} + \frac{U_{ADC} \cdot 10^3}{24}\ [mV]$

11 MBUS Configuration

MBUS module allows to use *LORATECH Combo* as a MBUS Master device.

11.1 Physical Connection

MBUS slaves are polarity independent. *LORATECH Combo MBUS* is physically designed to allow connection of up to 3 MBUS slaves, but current firmware supports only one MBUS slave.

To get it work, connect MBUS slave to connector denoted SL14.

11.2 MBUS Module Configuration

MBUS 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: `attempts`, `baudrate`, `challenge address` and `challenge class`. These commands define the BUS speed, number of attempts in case of error and the command, which will be used to read connected MBUS slave.

The slave response handling can be configured by using the following commands: `response offset` and `response length`.

MBUS handles with received data in the following way: the given number of bytes is cut from the response and further processed (e.g. transmitted though LoRa) – this data portion we call simply the *Cut*. The response *offset* states the first byte of the cut (starting from 0) and the *length* states the actual length of the cut.

The sample configuration procedure:

```
>> set MBUS attempts 3
Executing command: attempts

Write success!
>> set MBUS baudrate 9600
Executing command: baudrate

Write success!
>> set MBUS challenge address 00
Executing command: address

Write success!
>> set MBUS challenge class 2
Executing command: class

Write success!
>> set MBUS response offset 0
Executing command: offset

Write success!
>> set MBUS response length 10
Executing command: length

Write success!
>> enable MBUS
Executing command: enable

Setting module MBUS:     success!
```

```
>> reload
Executing command: reload

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

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

For debugging purposes, it may be important to be able to trace MBUS communication from console. If required, activate the `DebugDisplay` module as described in Section 6.4. The debug output, related to *MBUS* module, should look similarly:

```
>> Module "MBUS" GetSensorData()
>> MBUS :: sending data: DATA=10-40-00-40-16;
>> MBUS :: ACK recv.
>> MBUS :: Short Frame ACK recv.
>> MBUS :: sending data: DATA=10-7B-00-7B-16;
>> MBUS :: CTRL/Long Frame recv.
>> MBUS :: DATA recv.
>> MBUS :: slave data: DATA=68-33-33-68-08-00-72-35-37-32-69- ...
>> MBUS :: slave data cut: DATA=68-33-33-68-08-00-72-35-37-32;
```

11.3 MBUS Module Output

For error-free transaction, the MBUS module output consist of 1 byte indicating the cut length followed by the bytes of the cut, as shown below:

0x0A	0x68	0x33	0x33	0x68	0x08	0x00	0x72	0x35	0x37	0x32

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

0x00	ERR.

Following error codes are defined:

- 0x01 – communication timeout
- 0x02 – RSP_UD not received
- 0x03 – Slave does not support RSP_UD for given class
- 0x04 – 0xFF – RFU

12 MODBUS Configuration

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

12.1 Physical Connection

To get it work, connect ModBUS slave to connector denoted SL8.

PIN#1 is RS485 signal A, PIN#2 is RS485 signal B and PIN#3 is connected to ground to allow shield connection.

12.2 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, it may be important to be able to trace MODBUS communication from console. If required, activate the `DebugDisplay` module as described in Section 6.4. The debug output, related to *MODBUS* module, should look similarly:

```
>> Module "MODBUS" GetSensorData()
>> 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)
>> Module "MODBUS" SleepEnter()
```

12.3 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

13 S0 Counter Configuration

S0 Counter module allows to use *LORATECH Combo* as a S0 pulse input device [4].

13.1 Physical Connection

Up to 4 S0 output devices can be connected to *LORATECH Combo MBRSS4VMTH-BV* . One low-power input S0_0 and additional 3 inputs (S0_1, S0_2 and S0_3) 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.

13.1.1 S0_0 Connection

Connect S0 to connector denoted as SL2. PIN 1 is A+, PIN 2 is B-.

13.1.2 S0_1 Connection

Connect S0 to connector denoted as SL10. PIN 1 is A+, PIN 2 is B-.

13.1.3 S0_2 Connection

S0_2 and S0_3 counters share common A+ branch. Connect S0_2 inputs to connector denoted as SL7. PIN 2 is A+, PIN 3 is B-.

13.1.4 S0_3 Connection

S0_2 and S0_3 counters share common A+ branch. Connect S0_3 inputs to connector denoted as SL7. PIN 2 is A+, PIN 1 is B-.

13.2 S0 Counter Module Configuration

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

Each counter activation/deactivation is performed by using the following command: `OnOff`. The command parameter is an hexadecimal number (2 digits) representing the state of all available counters.

LORATECH Combo MBRSS4VMTH-BV allows to monitor the counter activity: the LoRa message is send in case of long inactivity. The inactivity period is configured by using the `Timeout` command. The timeout accuracy is 1 minute. For timeout deactivation set Timeout to 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* by the DebugDisplay module).

The current value of timers can be checked from LORATECH Terminal by using the `Value` command.

The sample configuration procedure:

```
>> set S0 OnOff 05
Executing command: OnOff

Write success!

>> set S0 Timeout 0 5
Executing command: Timeout
Write success!

>> set S0 Timeout 2 10
Executing command: Timeout
Write success!

>> enable AsyncTx
Executing command: enable

Setting module AsyncTx:     success!
>> enable S0
Executing command: enable

Setting module S0:     success!
>> 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 Timeout 0
Executing command: Timeout
Counter 0 timeout: 5 minutes
>>
>> show S0 Timeout 2
Executing command: Timeout
Counter 2 timeout: 10 minutes
>>
>> show S0 Value 0
Executing command: Value
Counter 0 current value: 0
>>
>> show S0 Value 2
Executing command: Value
Counter 0 current value: 0
```

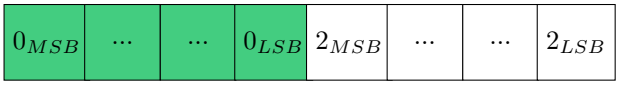
For debugging purposes, it may be useful to watch *S0 Counter* output in console. If required, activate the `DebugDisplay` module as described

in Section 6.4. The debug output, related to *S0 Counter* module, should look similarly:

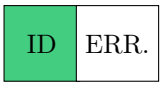
```
>> S0 :: CNT2 UP!
...
>> S0 :: CNT2 UP!
...
>> S0 :: CNT0 timeout!
>> Module "S0" GetSensorData()
>> S0 :: CNT0: 0
>> S0 :: CNT2: 2
```

13.3 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[3]. 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

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

14 Test Button Configuration

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

14.1 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 ...
```

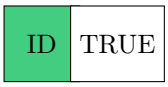
When reloaded, the *Test Button* module reacts to the push button events.

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
...
```

14.2 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[4]. The second byte – TRUE – represents the logic *true* value (represented by *1*).

[4] 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] University of West Bohemia in Pilsen, "KETCube appNote 003: Voltage Measurement up to 100V DC," 2018, -. [Online]. Available: https://github.com/SmartCAMPUSZCU/KETCube-docs/appNotes

[4] "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.

[5] Saft Specialty Battery Group, "Primary lithium battery LS 33600: 3.6 V Primary lithium-thionyl chloride (Li-SOCl2) High energy D-size bobbin cell," 2018, -. [Online]. Available: https://www.saftbatteries.com/

LORATECH Combo

Copyright

KETCube Platform

Copyright

Developed by

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

License[5]

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