Protocol of data exchange with modem via USB interface

Version 2016.12.15

- Modem connects to USB-host as USB device of CDC class (virtual COM port in Windows, ttyUSB or ttyACM in Linux)
- Because real RS-232 is not used in this interface, parameters of serial port set on the host (baudrate, number of bits, parity, etc.) may be any
- Data is in binary format
- «Network address» of modem is **0xff**
- Multibyte numbers are transmitted starting from low byte (little endian format)

1. Reading the latest coordinates pack (firmware V5.13+)

Format of request frame (from host to modem)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of modem	0xff
1	1	uint8_t	Type of packet	0x03
2	2	uint16_t	Code of data in packet	0x4110
4	2	uint16_t	Access mode	0x0000
6	2	uint16_t	CRC-16 (see appendix)	

Format of answer frame (from modem to host)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of modem	0xff
1	1	uint8_t	Type of packet	0x03
2	1	uint8_t	Number of bytes of data transmitting	0x64
3	100 (0x64)	100 bytes	Data structure (see lower)	
103	2	uint16_t	CRC-16 (see appendix)	

Format of data field (100 bytes)

Offset	Size (bytes)	Description
0	96 (6*16)	Six last coordinates structures received by modem (see lower)
96	4	Reserved

Format of coordinates structure (16 bytes)

Offset	Size (bytes)	Description
0	1	Address of device
1	4	Coordinate X, mm (int32_t)
5	4	Coordinate Y, mm (int32_t)
9	4	Coordinate Z, mm (int32_t)
13	1	Byte of flags: Bit 0: 1 – no relevant coordinates (red mode in dashboard) Bit 1: 1 – temporary mobile beacon on frozen map (blue mode) Bit 2: 1 – beacon is used for hedgehog positioning
14	2	Reserved (0)

2. Reading/writing modem configuration

2.1 Reading modem configuration (firmware V5.30+)

Format of request frame (from host to modem)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of modem	0xff
1	1	uint8_t	Type of packet	0x03
2	2	uint16_t	Code of data in packet	0x5000
4	2	uint16_t	Access mode	0x0000
6	2	uint16_t	CRC-16 (see appendix)	

Format of answer frame (from modem to host)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of modem	0xff
1	1	uint8_t	Type of packet	0x03
2	1	uint8_t	Number of bytes of data transmission	0x30
3	0x30	structure	Data structure (see section 2.3)	
0x33	2	uint16_t	CRC-16 (see appendix)	

2.2 Writing modem configuration

Warning! To write modem configuration you must read configuration, setup the data fields described in following section, and then write it. Do not change any other bytes in structure, this may degrade the work of modem

Format of request frame (from host to modem)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of modem	0xff
1	1	uint8_t	Type of packet	0x10
2	2	uint16_t	Code of data in packet	0x5000
4	2	uint16_t	Access mode	0x0000
6	1	uint8_t	Number of bytes of data transmission	0x30
7	0x30	structure	Data structure (see section 2.3)	
0x37	2	uint16_t	CRC-16 (see appendix)	

Format of answer frame (from modem to host)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of device	0xff
1	1	uint8_t	Type of packet	0x10
2	2	uint16_t	Code of data	0x5000
4	2	uint16_t	reserved	
6	2	uint16_t	CRC-16 (see appendix)	

2.3 Structure of modem configuration data

Many fields of data structure are not explained. Do not change the fields! They are used for adjustment system from the Dashboard program; unauthorized changing may degrade the work of modem

Offset	Size (bytes)	Type	Description
0	21	21 bytes	Not explained
20	1	int8_t	Temperature of air setting Vt (signed). Temperature is (Vt+23)°C
21	1	uint8_t	Address of the beacon that should have map coordinates X=0,Y=0
22	4	4 bytes	Not explained
26	1	uint8_t	Address of the beacon that should have map coordinates X>0,Y=0
27	1	uint8_t	Address of the beacon that should have map coordinates with Y>0
28	1	uint8_t	Control flags: Bit 0: not explained Bit 1: 1 - enabled filtering of mobile beacons movement Bit 2: not explained Bit 3: 1 - high resolution mode (output coordinates in mm instead cm) Bit 4: not explained Bit 5: 1 = mirroring of all map Bit 6: 1= power save mode (power save works only when all of the submaps are frozen) Bit 7: not explained
29	2	2 bytes	Not explained
31	1	uint8_t	N, determines maximum frequency of retrieving hedgehog coordinates F(N)= 2^(N-1) Hz, N= 0…4, F(5)= 12 Hz, F(6)= 16 Hz, F(7)= 16+ (maximum)
32	16	16 bytes	Not explained

3. Reading/writing submap configuration

3.1 Reading submap configuration (firmware V5.30+)

Format of request frame (from host to modem)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of modem	0xff
1	1	uint8_t	Type of packet	0x03
2	2	uint16_t	Code of data in packet	0x60XX where XX is number of submap
4	2	uint16_t	Access mode	0x0000
6	2	uint16_t	CRC-16 (see appendix)	

Format of answer frame (from modem to host)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of modem	0xff
1	1	uint8_t	Type of packet	0x03
2	1	uint8_t	Number of bytes of data transmission	0x50 (80)
3	80	structure	Data structure (see section 3.3)	
83	2	uint16_t	CRC-16 (see appendix)	

3.3 Structure of submap configuration data

Many fields of data structure are not explained. Do not change the fields! They are used for adjustment system from the Dashboard program; unauthorized changing may degrade the work of modem

Offset	Size (bytes)	Type	Description
0	1	uint8_t	Address of starting beacon for building submap
1	1	uint8_t	Control word: Bit 0: 1 - submap is frozen (freeze submap) Bit 1: 1 - beacons are higher than hedgehogs Bit 2...4: not explained Bit 5: 1 - mirroring submap Bit 6...7: not explained
2	1	uint8_t	Limitation of distances: Bit 0...6: manual limitation distances (if bit 7 = 1) Bit 7: 0 - automatic limitation, 1 = manual
3	13	13 bytes	Not explained
16	2	int16_t	X shift of submap, cm
18	2	int16_t	Y shift of submap, cm
20	2	uint16_t	Rotation of submap, centidegrees
22	58	58 bytes	Not explained

3.2 Writing submap configuration (firmware V5.30+)

Warning! To write modem configuration you must read configuration, setup the data fields described in following section, and then write it. Do not change any other bytes in structure, this may degrade the work of modem

Format of request frame (from host to modem)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of modem	0xff
1	1	uint8_t	Type of packet	0x10
2	2	uint16_t	Code of data in packet	0x60XX where XX is number of submap
4	2	uint16_t	Access mode	0x0000
6	1	uint8_t	Number of bytes of data transmission	0x50 (80)
7	80	structure	Data structure (see section 3.3)	
87	2	uint16_t	CRC-16 (see appendix)	

Format of answer frame (from modem to host)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of device	0xff
1	1	uint8_t	Type of packet	0x10
2	2	uint16_t	Code of data	0x5000
4	2	uint16_t	reserved	
6	2	uint16_t	CRC-16 (see appendix)	

4. Sleeping/waking up devices

Format of request frame (from host to modem)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of device	0x01…0x63
1	1	uint8_t	Type of packet	0x10
2	2	uint16_t	Code of data in packet	0xb006
4	2	uint16_t	Access mode	0x0000
6	1	uint8_t	Number of bytes of data transmission	0x08
7	1	uint8_t	Password, byte 0	0x2d
8	1	uint8_t	Password, byte 1	0x94
9	1	uint8_t	Password, byte 2	0x5e
10	1	uint8_t	Password, byte 3	0x81
11	1	uint8_t	Command: 0 – standard sleep 1 – deep sleep (wake only on HW reset) 2 – wake up from standard sleep 3…255 - reserved	0…2
12	3	3 bytes	reserved	
15	2	uint16_t	CRC-16 (see appendix)	

Format of answer frame (from modem to host)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of device	0x01…0x63
1	1	uint8_t	Type of packet	0x10
2	2	uint16_t	Code of data	0xb006
4	2	uint16_t	reserved	
6	2	uint16_t	CRC-16 (see appendix)	

5. Setting address of device

Format of request frame (from host to modem)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of device	0x01…0x63
1	1	uint8_t	Type of packet	0x10
2	2	uint16_t	Code of data in packet	0x0101
4	2	uint16_t	Access mode	0x0000
6	1	uint8_t	Number of bytes of data transmission	0x02
7	1	uint8_t	Code of data item (address)	0x00
8	1	uint8_t	New address of device	
9	2	uint16_t	CRC-16 (see appendix)	

6. Reading raw measured distances.

This command is accessible in two modes:

- With code of data 0x4000 – reading last eight distances. Answer frame contains last 8 measured distances from the moment of request
- With code of data 0x4001 – reading all distances frame by frame. Answer frame for every next request contains next 8 saved measured distances. When all table of distances is transmitted, it starts from the beginning

Format of request frame (from host to modem)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of modem	0xff
1	1	uint8_t	Type of packet	0x03
2	2	uint16_t	Code of data in packet	0x4000 or 0x4001
4	2	uint16_t	Access mode	0x0000
6	2	uint16_t	CRC-16 (see appendix)	

Format of answer frame (from modem to host)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of modem	0xff
1	1	uint8_t	Type of packet	0x03
2	1	uint8_t	Number of bytes of data transmitting	0x28
3	40 (0x28)	40 bytes	Data structure (see lower)	
43	2	uint16_t	CRC-16 (see appendix)	

Format of data field (40 bytes)

Offset	Size (bytes)	Description
0	32 (8*4)	Eight raw distances structures (see lower)
32	8	Reserved

Format of distance structure (4 bytes)

Offset	Size (bytes)	Description
0	1	Address of ultrasonic receiver
1	1	Address of ultrasonic transmitter
2	2	Measured distance between devices, mm (uint16_t)

7. Reading beacons's state (firmware V5.33+)

Format of request frame (from host to modem)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of device	0x01…0x63
1	1	uint8_t	Type of packet	0x03
2	2	uint16_t	Code of data in packet	0x0003
4	2	uint16_t	Access mode	0x0002
6	2	uint16_t	CRC-16 (see appendix)	

Format of answer frame (from modem to host)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of device	0x01…0x63
1	1	uint8_t	Type of packet	0x03
2	1	uint8_t	Number of bytes of data transmission	0x20
3	36	36 bytes	Data structure (see lower)	
39	2	uint16_t	CRC-16 (see appendix)	

Format of data field:

Offset	Size (bytes)	Type	Description
0	4	uint32_t	Time of work from reset or wake-up (seconds)
4	1	uint8_t	R, radio RSSI register value (received signal strength indicator). If R>128, RSSI(dBm) = ((R-256)/2) – 74 If R<=128) RSSI (dBm) = (R/2) - 74
5	1	uint8_t	Not explained
6	1	int8_t	Measured temperature Vt (signed). Temperature is (Vt+23)°C
7	2	uint16_t	Bit 0...11 : power supply voltage, mV Bit 12...13: not explained Bit 14: 1: low power, device will enter sleep after short time Bit 15: 1: very low power, device will enter deep sleep after short time
9	23	23 bytes	Not explained

8. Marvelmind robots control commands

8.1. Robot control command

Format of request frame (from host to modem)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of robot	0x01…0x63
1	1	uint8_t	Type of packet	0x10
2	2	uint16_t	Code of data in packet	0x1000
4	2	uint16_t	Access mode	0x0001
6	1	uint8_t	Number of bytes of data transmission	0x10
7	16 (0x10) bytes	uint8_t	Robot control data (see lower)	
23	2	uint16_t	CRC-16 (see appendix)	

Format of answer frame (from modem to host)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of robot	0x01…0x63
1	1	uint8_t	Type of packet	0x10
2	2	uint16_t	Code of data	0x1000
4	2	uint16_t	reserved	
6	2	uint16_t	CRC-16 (see appendix)	

Format of robot control data:

Offset	Size (bytes)	Type	Description
0	1	uint8_t	Mode of control: 0 - no control (wait mode) 1 - motors power control 2 - speed control 3 - writing movement program 4 - pause movement program 5 - continue movement after pause
1	1	uint8_t	Code of operation: 0 - move forward 1 - move backward 2 - rotate clockwise 3 - rotate counterclockwise 4 - pause for given time (for mode 3) 5 - repeat movement program from start (for mode 3) 6 - move to given point by coordinates (for mode 3) 7 - setup movement speed (for mode 3)
2	1	uint8_t	Control byte 1: For mode 1: power on motors, % For mode 2: speed of movement, % For mode 3: number of the program step (starting from zero)
3	2	Int16_t	Data for program (mode 3): Code of operation 0 or 1: distance of movement, cm Code of operation 2 or 3: angle of rotation, degrees

			Code of operation 4: time of pause, ms Code of operation 6: X coordinate of movement target, cm Code of operation 7: speed of movement, %
5	1	uint8_t	For mode 3: total number of steps in program. Maximum supported number of steps is 8.
6	2	int16_t	Additional data for program (mode 3): Code of operation 6: Y coordinate of movement target, cm
8	8	8 bytes	Reserved (0)

Some comments for this complicated command.

There are three main modes of robot control specified in byte 0 of robot control structure:

- power control (mode 1)
- speed control (mode 2)
- move by program (mode 3)

Mode 1 and mode 2 are generally used for test purposes. In mode 1 robot moves forward, backward, rotates left or right with selected power on motors. In mode 2 robot makes the same, but adjusting power to keep selected speed. The power or speed is set in byte 2 of structure, type of movement - in byte 1.

Mode 4 and mode 5 are special commands for pausing movement during program execution and continuing movement after pause.

The main mode for moving on complex trajectories is mode 3.

It allows to program to robot the sequence of primitive actions, which combination builds the trajectory. Each item of the sequence should be sent by one command of this type. Each command should contain the number of the current step in the byte 2 of robot control structure, and total number of steps in the byte 5 (maximum supported number of steps is 8).

In the byte 1 of robot control structure the type of primitive movement is specified. Parameters of the primitive movement are specified in fields "data for program" (bytes 3...4) and "additional data for program" (bytes 6...7).

So the following primitives are available:

- move forward by given distance;
- move backward by given distance;
- rotate clockwise by given angle;
- rotate counterclockwise by given angle;
- pause by given time;
- restart the movement program from first item (for looping moviments);
- move to given point (X, Y) in Marvelmind navigation system coordinates;
- change movement speed.

Robot begins execution of the program after receiving the sequence of primitives. After program execution, robot stops. But if the program contains item with code of operation 5 (repeat from start), the program repeats loop which will be executed forever, until receiving stop command or uploading new program..

8.2. Stop robot

Format of request frame (from host to modem)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of robot	0x01…0x63
1	1	uint8_t	Type of packet	0x10
2	2	uint16_t	Code of data in packet	0x403
4	2	uint16_t	Access mode	0x0001
6	1	uint8_t	Number of bytes of data transmission	0x04
7	4 bytes	4 bytes	Reserved (0)	0
11	2	uint16_t	CRC-16 (see appendix)	

Format of answer frame (from modem to host)

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of robot	0x01…0x63
1	1	uint8_t	Type of packet	0x10
2	2	uint16_t	Code of data	0x403
4	2	uint16_t	reserved	
6	2	uint16_t	CRC-16 (see appendix)	

This command simply terminates execution of any robot movement or program of movements. The robot stops and waits for new commands.

Appendix 1. Calculating CRC-16.

For checksum the CRC-16 is used. Last two bytes of N-bytes frame are filled with CRC-16, applied to first (N-2) bytes of frame. To check data you can apply CRC-16 to all frame of N bytes, the result value should be zero.

Below is the implementation of the algorithm in the 'C'.

```
typedef ushort ModbusCrc;//  ushort – two bytes

typedef union {
	ushort w;
	struct{
		uchar lo;
		uchar hi;
	} b;
	uchar bs[2];
} Bytes;

static ModbusCrc modbusCalcCrc(const void *buf, ushort length)
{
	uchar *arr = (uchar *)buf;
	Bytes crc;

	crc.w = 0xffff;

	while(length--){
		char i;
		bool odd;

		crc.b.lo ^= *arr++;
		for(i = 0; i < 8; i++){
			odd = crc.w & 0x01;
			crc.w >>= 1;
			if(odd)
				crc.w ^= 0xa001;
		}
	}

	return (ModbusCrc)crc.w;
}
```