@author Kmakise	Modbus Protocol Weight Transmitter	Ver.1.0.0 Build：001

This Device is able to support the MODBUS RTU communication protocol on 2 wires RS485 serial line. With the MODBUS communication protocol you can read the data state on multiple online device and check them through software,standards supervisory software provided by third parties or through interface with equipment such as MODBUS PLC terminals and data processing. The MODBUS protocol is based on a Master-Slave architecture, where requests for interrogating are unidirectional and executed only by the master (usually a PC) to the Slave . Indeed, if the Slave does not questioned by the Master, do not send any signal. The Slave when interviewed by the Master meet on predefined rules (defined by the MODBUS protocol) and not ever generate messages on its own initiative while remaining in the passive state pending Master request. All slave devices must have a different address in order to be recognized by the Master, if not, the whole system may still have some problems during operation. RTU protocol is a binary code and is the most common, besides being fast, having a message length below by almost 50% compared to the ASCII protocol.To be able to converse with each other, the master and all slaves should have the same protocol (RTU), speed, stop bits and parity. For more information you can contact us.

1 MODBUS structure Protocol

The MODBUS protocol common structure that is independent of the communication type (serial, TCP/IP) is characterized by 2 communication fields which are: data and function code.
On serial line, however, the command string consists of 4 communication fields:

- Device address;
- Function code;
- Data;
- CRC.

1.1 Slave Address

The field address (Slave Address) serves to indicate what Slave is called by Master. The valid slave address can be between 1 and 128. Please, note that the Slaves must have different addresses. To communicate with a slave, the Master into the address field the value of the slave address, which in turn will use in the reply message. The values that the Master may enter inside the string are:

- 0 = address 0, or "Broadcasting", was sent to all slave who must not be answered;
- 1÷128 = free addresses for the Slave devices addressing;

1.2 Function Code

The function code used to indicate to the slave the request of the master, and then the operation type to be performed, if the slave could not make this request will send an error code. The codes that can be sent must be between 1 and 83.
To determine which functions are manageable refer to maps on the next page.

1.3 Data

The data field contains data sent from master to slave, or sent in response from Slave to Master. The data fields are multiples of 16-bit registers (1WORD = 2byte, 1byte = 8bit). Each WORD is always transmitted from the most significant byte. Depending on how the Slave records are set, you can view or change values in sequence, and it’ s possible if the records in question are adjacent to each other.

1.4 CRC

This field is used to verify the integrity of the received message. Is calculated and attached to the message by the transmitting station. The receiving station, as a first step, recalculate this field and compare it with that received. Is generated in the case of RTU (CRC Cyclic Redundancy Checking).

2 Introduction to MODBUS RTU

2.1 MODBUS RTU protocol

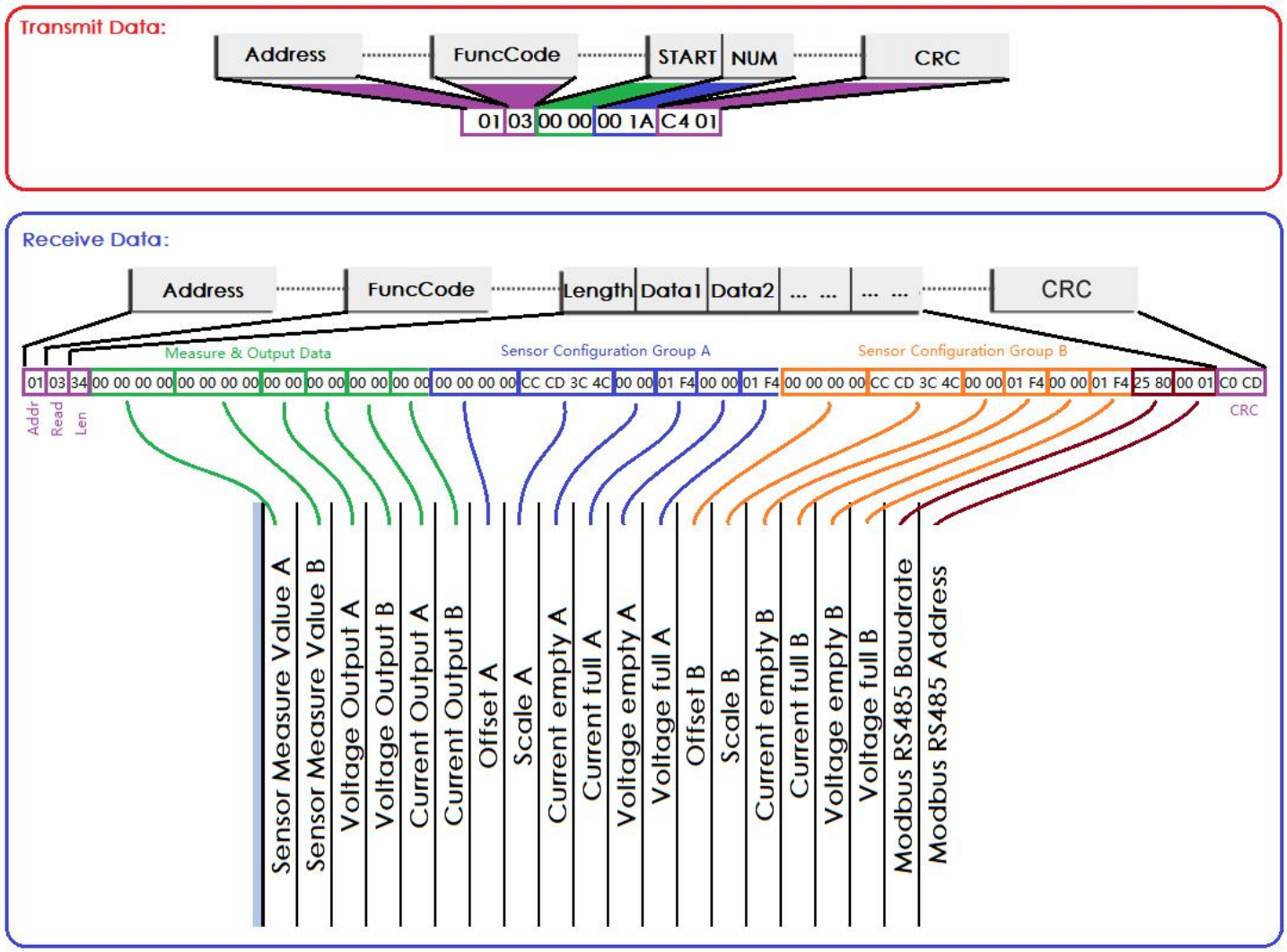
The string of communication consists of:

- T1 T2 T3 T4
- device address (1 byte)
- data (N x 2 byte; N = contiguous registers numbers to read or send)
- CRC (2 byte)
- T1 T2 T3 T4

"T1 T2 T3 T4"
indicates the time that must elapse before a subsequent communication to avoid collisions of messages, the overall structure of the byte is composed of:

- 1 start bit;
- 8 data bits (transmitted from the least significant bit);
- 1 parity bit + 1 stop bit, if there is no parity bit using 2 stop bits.

2.2 EXAMPLE



3 Mappings

To enable the "system integrator" to develop a levels management software, following the mapping is represented in the device series.

3.1 MODBUS possible functions

Modality: RTU
Parity: none (bit not sent)
Times: minimum waiting time between two successive transmissions: time of 1 character multiplied by 3.5
Read / write:
03, 04 – "read holding register"
06 – "write single register"
Diagnostics: unmanaged
Information Management: field data up to 100 bytes

DATA MAPPINGS					
Register Dec.	*Address Hex.*	*Dim. (16bit)*	*Description*	*unit*	*Function*
01	*00*	2	Sensor Measure Value A	kg	03h
03	*02*	2	Sensor Measure Value B	kg	03h
05	*04*	1	Voltage Output A	mV	03h
06	*05*	1	Voltage Output B	mV	03h
07	*06*	1	Current Output A	mA	03h
08	*07*	1	Current Output B	mA	03h
09	*08*	2	Offset A	kg	03h 06h
11	*A*	2	Scale A	kg/cnt	03h 06h
13	*C*	1	Current empty A	kg	03h 06h
14	*D*	1	Current full A	kg	03h 06h
15	*E*	1	Voltage empty A	kg	03h 06h
16	*F*	1	Voltage full A	kg	03h 06h
17	*10*	2	Offset B	kg	03h 06h
19	*12*	2	Scale B	kg/cnt	03h 06h

21	*14*	1	Current empty B	kg	03h 06h
22	*15*	1	Current full B	kg	03h 06h
23	*16*	1	Voltage empty B	kg	03h 06h
24	*17*	1	Voltage full B	kg	03h 06h
25	*18*	1	Modbus RS485 Baudrate	bps	03h 06h
26	*19*	1	Modbus RS485 Address		03h 06h

4. Combination

```
float Char_To_Float(void *p)
{
	float f;
	char *ch = (char *)p;
	char *pf = (char *)&f;
	pf[0] = ch[1];
	pf[1] = ch[0];
	pf[2] = ch[3];
	pf[3] = ch[2];
	return f;
}

int16_t Char_To_int16(void *p)
{
	char *ch = p;
	return ((int16_t)ch[0]<<8)+ch[1];
}
```