WRTU Client Protocol

Date: 18 December, 2014
Document Revision: 1.01B

BiPOM Electronics

Telephone : 1-713-283-9970
E-mail : info@bipom.com
Web : www.bipom.com

TABLE OF CONTENTS

1. Introduction 6

2. BiPOM Custom Protocol 7

3. General Commands 10
- 3.1 Read Device information 10
- 3.2 Write Device information 11
- 3.3 Read Device Time 12
- 3.4 Set Device Time 13
- 3.5 Set Default Configuration 14
- 3.6 Restart Device 15
- 3.7 Set Device Features 16
- 3.8 Get Device Features 17
- 3.9 Read Modem Model 18
- 3.10 Read Hardware Status 19

4. Hardware Commands 20
- 4.1 Read DATAFLASH Sector 20
- 4.2 Write DATAFLASH Sector 21
- 4.3 Read DATAFLASH CRC16 22
- 4.4 Read ADC Calibration Information 23
- 4.5 Calibrate ADC Channel 24
- 4.6 Read RS485 Port Configuration 25
- 4.7 Write RS485 Port Configuration 26
- 4.8 Read RS232 Port Configuration 27
- 4.9 Write RS232 Port Configuration 28
- 4.10 Read LCD Configuration 29
- 4.11 Write LCD Configuration 30
- 4.12 Read TLS2543 Selftest Values 31

5. Tags Commands 32
- 5.1 Read Tag List 32
- 5.2 Read Tag Configuration 33
- 5.3 Write Tag Configuration 36
- 5.4 Delete Tag 38
- 5.5 Delete All Tags 39
- 5.6 Read Values Map Item List 40
- 5.7 Read Values Map Item Configuration 41
- 5.8 Write Values Map Item Configuration 42
- 5.9 Delete Values Map Item 43
- 5.10 Delete All Values Map Items 44
- 5.11 Read Contact List 45
- 5.12 Read Contact Configuration 46
- 5.13 Write Contact Configuration 47

5.14 Delete Contact 48
5.15 Delete All Contacts 49
5.16 Read Message List 50
5.17 Read Message Configuration 51
5.18 Write Message Configuration 52
5.19 Delete Message 53
5.20 Delete All Messages 54
5.21 Read Live Value of Tag 55
5.22 Write Live Value of All Tags 56

6. Logger Commands 57
6.1 Set Logger State 57
6.2 Get Logger State 58
6.3 Initialize Log Reading 59
6.4 Read Log Records 60
6.5 Clear Log 62

7. Communication Commands 63
7.1 Read APN Configuration 63
7.2 Write APN Configuration 64
7.3 Read Data Pushing Configuration 65
7.4 Write Data Pushing Configuration 66
7.5 Read SMTP Configuration 67
7.6 Write SMTP Configuration 68

8. Constants 69
8.1 ADC Channels 69
8.2 Alarm Conditions 69
8.3 Alarm Types 69
8.4 Byte Orders 69
8.5 Modem Types 70
8.6 Data Pushing Output Format 70
8.7 Data Pushing Protocols 71
8.8 Device Reset Codes 72
8.9 Equation Types 72
8.10 Event Types 72
8.11 Hardware Types 72
8.12 Logger States 73
8.13 Modbus Register Types 73
8.14 Modbus Register Value Types 73
8.15 Data Record Types 73
8.16 Tag Types 74
8.17 Tag Value States 74
8.18 Event ID 74

9. Samples 77
9.1 Read device configuration 77
9.2 Write new device configuration 78
9.3 Read logged data 79

Appendix A: Internal Modbus Registers 80
Appendix B: Error codes 84
Appendix C: Event codes 89
Appendix D: Calculating Modbus CRC16 91
Appendix E: Calculating DATAFLASH and SD card CRC16 93
Appendix F: Protocol Limits 94
Appendix G: Tag Configuration Description 95
Appendix H: Calculating CRC8 99

1. Introduction

This document describes configuration and communications protocol of WRTU system.

This protocol is used by WRTU Client to configure device (read existing configuration or write new configuration) and also to read logged data.

The protocol is based on the standard Modbus protocol. It uses a new custom function (20 decimal, 0x14 hexadecimal). This custom command is used for implementing binary protocol which has its own commands set and command packet format. For checking data consistency and correctness used standard Modbus CRC16 method so internal underlying protocol does not have its own mechanism to check data correctness and validity and this is done when Modbus whole packet is checked.

Why custom binary protocol?

By default, any WRTU device can operate as a Modbus slave device that has a predefined set of Modbus registers which can be read by any Modbus client on any communication port (RS232 serial port, USB serial port, RS485 serial port, Ethernet port using Modbus TCP).

To avoid introducing a new protocol that would make the communications system complicated, a custom extension was added to existing Modbus protocols (Modbus RTU and Modbus TCP). Modbus protocol is simple and has features to check data validity and correctness and can be easily extended to be the transport protocol for any underlying protocol. So a Modbus packet can be used for transferring other types of packets.

2. BiPOM Custom Protocol

BiPOM Custom Protocol is based on standard Modbus protocol.
Information about standard Modbus protocol can be found here: www.modbus.com

Standard Modbus RTU packet is following:

<Slave ID> <Function> <Function Data> <CRC16>

Slave ID – 1 byte slave device address. The normally it is number from 1 to 247.
Function – 1 byte Modbus Function ID.
Function Data – additional information depending on **Function**.
CRC16 – 2 bytes CRC16 of whole Modbus packet.

BiPOM Protocol uses new custom Function : 0x14 (20 decimal).

So BiPOM custom command looks like following:

<Slave ID> **0x14** < Custom Command Length> <Custom Command Packet> <CRC16>

Custom Command Length - 2 bytes which keeps length of Custom Command Packet in bytes.
Custom Command Packet - custom command bytes.

Custom Command Length is 2 bytes unsigned integer value. Highest byte of value placed in first byte, the lowest byte – in second byte.

Example:

0x00 0x08 - this means that Custom Command Packet has 8 bytes. CRC16 is not included.
0x01 0x00 - this means that Custom Command Packet has 256 bytes. CRC16 is not included.

Custom Command Packet has following format:

<Command ID> <Command Data>

Command ID - 1 byte of Command ID.
Command Data - 1 or more bytes which contains command parameters or response data. Note, some commands does not have parameters, but they still should put 1 zero byte as command data.

Example of custom command sent to device with RTU 0 to set default configuration of device:

0x00 0x14 0x00 0x02 0x0C 0x00 0x94 0xD8

0x00 - Slave ID 0.
0x14 - Custom Modbus Function which mean that it is BiPOM Protocol command.
0x00 0x02 - Custom Command Length (2 bytes).
0x0C - Command ID 0x0C – Reset Device Configuration.
0x00 - Command Data – blank byte 0x00. It is because command 0x0C doesn't have parameters.
0x94 0xD8 - CRC16 of whole packet (0x00 0x14 0x00 0x02 0x0C 0x00).

NOTE: any WRTU device always will process request with address 0. It is broadcast address for WRTU device.

Bytes order

Any numbers transferred in underplayed BiPOM Protocol always placed in order from highest to lowest bytes.

Examples:

0xF1A0

0xF1 will be put in first byte
0xA0 will be put in second byte

0x11223344

0x11 will be put in first byte
0x22 will be put in second byte
0x33 will be put in third byte
0x44 will be put in fourth byte

Sample C++ code to put integer 16 bits value into data packet:

```
char    data[2];
int     v = 0x1122;

data[0] = v >> 8;
data[1] = v;
```

Sample C++ code to read integer 16 bits value from data packet:

```
int     v = 0;

v |= data[0] << 8;
v |= data[1];
```

Sample C++ code to put integer 32 bits value into data packet:

```
char    data[4];
long    v = 0x11223344;

data[0] = v >> 24;
data[0] = v >> 16;
data[0] = v >> 8;
data[1] = v;
```

Sample C++ code to read integer 32 bits value from data packet:

```
long    v = 0;

v |= data[0] << 24;
v |= data[0] << 16;
v |= data[0] << 8;
v |= data[1];
```

Real values can be also used. Only 4 bytes float value is used.

Sample C++ code to put float value into data packet:

```
char     data[4];
float    v = 1000;

unsigned char *pData = (unsigned char *)&v;
data[0] = pData[0];
data[1] = pData[1];
data[2] = pData[2];
data[3] = pData[3];
```

Sample C++ code to read float value from data packet:

```
float    v = 1000;

unsigned char *pData = (unsigned char *)&v;
pData [0] = data[0];
pData [1] = data[1];
pData [2] = data[2];
pData [3] = data[3];
```

3. General Commands

3.1 Read Device information

Read general device configuration from device: device unique number, device name, RTU number and other options. Please see **Response** table for more information.

Request

Byte #	Description
0	0x01 (Command ID)
1	Blank 0x00 byte

Response

Byte #	Description
0	0x01 (Command ID)
1,2	Error code
3	Command Version (latest version is 0x01)
4-7	UID, Unique device identifier.
8	RTU number
9 – 40	Device name (zero ended, always 32 bytes)
41, 42	SMS Time Limit (unsigned 16 bit integer)
43	RS232 <-> RS485 Bridge Mode. 0 – disabled, 1 – enabled
44	Log tags at aligned time period option state. 0 – disabled, 1 enabled
45, 46	Aligned Time Period in minutes (unsigned 16 bit integer)

3.2 Write Device information

Write general device configuration to device. The changes are written to DATAFLASH as soon as command is processed.

Request

Byte #	Description
0	0x02 (Command ID)
1	Command Version (latest version is 0x01)
2	RTU Number
3 – 34	Device name (zero ended, always 32 bytes)
35,36	SMS Time Limit (unsigned 16 bit integer)
37	RS232 <-> RS485 Bridge Mode. 0 – disabled, 1 – enabled
38	Log tags at aligned time period option state. 0 – disabled, 1 enabled
39, 40	Aligned Time Period in minutes (unsigned 16 bit integer)

Response

Byte #	Description
0	0x01 (Command ID)
1,2	Error code

3.3 Read Device Time

Read current RTC state from device.

Request

Byte #	Description
0	0x03 (Command ID)
1	Blank 0x00 byte

Response

Byte #	Description
0	0x03 (Command ID)
1,2	Error code
3	Command Version (latest version is 0x01)
4,5	YEAR
6	MONTH
7	DAY
8	HOUR
9	MINUTES
10	SECONDS

3.4 Set Device Time

Set specified Date/Time to device RTC.

Request

Byte #	Description
0	0x04 (Command ID)
1	Command Version (latest version is 0x01)
2,3	YEAR
4	MONTH
5	DAY
6	HOUR
7	MINUTES
8	SECONDS

Response

Byte #	Description
0	0x04 (Command ID)
1,2	Error code

3.5 Set Default Configuration

Reset device configuration and set all default values: removed all tags, contacts, messages, Values Map items.

Request

Byte #	Description
0	0x0C (Command ID)
1	Blank 0x00 byte

Response

Byte #	Description
0	0x0C (Command ID)
1,2	Error code

3.6 Restart Device

Cause to restart firmware on the device.

Request

Byte #	Description
0	0x0D (Command ID)
1	Blank 0x00 byte

Response

Byte #	Description
0	0x0D (Command ID)
1,2	Error code

3.7 Set Device Features

Enable/disable device features.

Request

Byte #	Description
0	0x19 (Command ID)
1	Command Version (latest version is 0x01)
2-5	Unsigned long integer bit mask of features. 1 – enabled, 0 - disabled

Features bit masks

0x01 Enable/disable cell modem communication.

Response

Byte #	Description
0	0x19 (Command ID)
1,2	Error code

3.8 Get Device Features

Read bit mask as unsigned long integer (32 bits). Each bit represents one feature: 1 – enabled, 0 – disabled.

Request

Byte #	Description
0	0x1A (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x1A (Command ID)
1,2	Error code
3	Command Version (latest version is 0x01)
4-7	Unsigned long integer bit mask of features. 1 – enabled, 0 - disabled

3.9 Read Modem Model

Read modem model identifier.

Request

Byte #	Description
0	0x44 (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x44 (Command ID)
1,2	Error code
3	Modem model identifier

Modem models

Identifier Value	Description
0	Unknown
1	WMP50 2G Modem
2	HE-910D 3G Modem

Note: In case of CG-nanoWiPOM board modem model always set to 0.

3.10 Read Hardware Status

Read bit mask as unsigned long integer (32 bits). Each bit represents one hardware subsystem.
If bit is set this means that hardware is initialized successfully. If 0 – the hardware system was not initialized.

Request

Byte #	Description
0	0x1D (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x1A (Command ID)
1,2	Error code
3	Command Version (latest version is 0x01)
4-7	Unsigned long integer bit mask of hardware status

Bits maps

Bit #	Description
0	Core system
1	UART0
2	UART1
3	UART2
4	UART3
5	USB
6	I/O
7	ADC
8	RS232 Software Driver
9	RS485 Software Driver
10	Cell Modem Software Driver
11	USB Software Driver
12	LCD DISPLAY
13	TLC2543
14	Cell Modem

4. Hardware Commands

4.1 Read DATAFLASH Sector

Read one 512 bytes sector from DATAFLASH chip placed on a device.

Request

Byte #	Description
0	0x0E (Command ID)
1-4	Sector address

Response

Byte #	Description
0	0x0E (Command ID)
1,2	Error code (0 success)
3-514	512 bytes read from DATAFLASH sector

Errors

1002 – DATAFLASH was not initialized.
1009 – DATAFLASH read operation failed.
1010 – DATAFLASH write operation failed.
1014 – incorrect sector number (outside of available sectors).

4.2 Write DATAFLASH Sector

Read one 512 bytes sector from DATAFLASH chip placed on a device.

Request

Byte #	Description
0	0x0F (Command ID)
1-4	Sector address
5-516	512 bytes data to write to sector

Response

Byte #	Description
0	0x0F (Command ID)
1,2	Error code (0 success)

Errors

1002 – DATAFLASH was not initialized.
1009 – DATAFLASH read operation failed.
1010 – DATAFLASH write operation failed.
1014 – incorrect sector number (outside of available sectors).

4.3 Read DATAFLASH CRC16

Force firmware to calculate CRC16 of specified sectors on DATAFLASH.

Request

Byte #	Description
0	0x10 (Command ID)
1-4	Sector address
5,6	Number of sectors to calculate

Response

Byte #	Description
0	0x10 (Command ID)
1,2	Error code (0 success)
3,4	CRC16 value

Errors

1002 – DATAFLASH was not initialized.
1009 – DATAFLASH read operation failed.
1010 – DATAFLASH write operation failed.
1014 – incorrect sector number (outside of available sectors).

4.4 Read ADC Calibration Information

Read stored calibration for analog inputs AIN1 and AIN2. This calibration defines ZERO OFFSET which used when calculates RMS values on analog outputs.

Request

Byte #	Description
0	0x14 (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x14 (Command ID)
1,2	Error code (0 success)
3-6	AIN1 zero offset in volts (float 4 bytes value)
7-10	AIN2 zero offset in volts (float 4 bytes value)

Note: please see **Bytes Order** to correctly convert 4 bytes to float value.

4.5 Calibrate ADC Channel

Request firmware to store current ADC input value as ZERO OFFSET. This value will be used for calculating RMS value on this analog input.

Request

Byte #	Description
0	0x15 (Command ID)
1	Channel number. 0 – AIN1, 1 – AIN2

Response

Byte #	Description
0	0x15 (Command ID)
1,2	Error code (0 success)
3-6	New zero offset in volts (float 4 bytes value) for calibrated channel

4.6 Read RS485 Port Configuration

Read RS485 serial port communication settings.

Request

Byte #	Description
0	0x1B (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x1B (Command ID)
1,2	Error code (0 success)
3	Command Version (latest version is 0x01)
4-7	Baudrate
8	Parity. 0 – NONE, 1 – ODD, 2 – EVEN
9	Stop bits (1 or 2)
10	Data bits (8 or 9)

4.7 Write RS485 Port Configuration

Write RS485 serial port communication settings.

Request

Byte #	Description
0	0x1C (Command ID)
1	Command Version (latest version is 0x01)
2-5	Baudrate
6	Parity. 0 – NONE, 1 – ODD, 2 – EVEN
7	Data bits (8 or 9)
8	Stop bits (1 or 2)

Response

Byte #	Description
0	0x1C (Command ID)
1,2	Error code (0 success)

4.8 Read RS232 Port Configuration

Read RS232 serial port communication settings.

Request

Byte #	Description
0	0x28 (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x28 (Command ID)
1,2	Error code (0 success)
3	Command Version (latest version is 0x01)
4-7	Baudrate
8	Parity. 0 – NONE, 1 – ODD, 2 – EVEN
9	Stop bits (1 or 2)
10	Data bits (8 or 9)

4.9 Write RS232 Port Configuration

Write RS232 serial port communication settings.

Request

Byte #	Description
0	0x29 (Command ID)
1	Command Version (latest version is 0x01)
2-5	Baudrate
6	Parity. 0 – NONE, 1 – ODD, 2 – EVEN
7	Data bits (8 or 9)
8	Stop bits (1 or 2)

Response

Byte #	Description
0	0x29 (Command ID)
1,2	Error code (0 success)

4.10 Read LCD Configuration

Read LCD display settings.

Request

Byte #	Description
0	0x42 (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x42 (Command ID)
1,2	Error code (0 success)
3	Command Version (latest version is 0x01)
4	Display pages switch delay in seconds

4.11 Write LCD Configuration

Write LCD display settings.

Request

Byte #	Description
0	0x42 (Command ID)
1	Command Version (latest version is 0x01)
2	Display pages switch delay in seconds

Response

Byte #	Description
0	0x42 (Command ID)
1,2	Error code (0 success)

4.12 Read TLS2543 Selftest Values

Read TLS2543 peripheral board self test values. TLC2543 has 3 analog inputs configured to return BOTTOM, MIDDLE and TOP points. Firmware can read and compare them to predefined values to check if TLC2543 peripheral board is presented and operational.

Request

Byte #	Description
0	0x2C (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x2C (Command ID)
1,2	Error code (0 success)
3	Command Version (latest version is 0x01)
4,5	BOTTOM point analog input value. Should be less or equal to 2
6,7	MIDDLE point analog input value. Should be in range [2046, 2050]
8,9	TOP point analog input value. Should be greater or equal to 4093.

5. Tags Commands

5.1 Read Tag List

Read ID of all configured tags. This command allow get list of ID of all configured tags and then requests the tag configuration or tag's live value using this ID value.

Request

Byte #	Description
0	0x05 (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x05 (Command ID)
1,2	Error code (0 success)
3	Number of tags in the list
4-N	Tag ID List

Note: Tag ID List is byte sequence. Every tag ID is unsigned 2 bytes integer. So number of bytes N depended on number of tags. For example:
- if number of tags is 1, then Tag ID List will contain only 2 bytes.
- if number of tags is 10, then Tag ID List will contain only 20 bytes.

5.2 Read Tag Configuration

Read single tag configuration parameters. See **Appendix G: Tag Configuration Description** for more information about Tag Configuration parameters.

Request

Byte #	Description
0	0x06 (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x06 (Command ID)
1,2	Error code (0 success)
3	Command Version (latest version is 0x03)
4,5	Tag ID
6	Tag type
7	Flag of logger state. 0 – this tag should not be logged. 1 – it should be logged
8	Flag of Values Map state. 0 – Values Map is not used. 1 – Values Map is used.
9-32	Tag name (always 24 bytes). Zero ended.
33,34	Internal Modbus register address to what tag is bind
35-38	Logging period in seconds
39-62	Units1 text
63-86	Units2 text
87	Equation type.
88-91	Equation parameter A float value
92-95	Equation parameter B float value
96	External (RS485) Modbus device RTU
97	External (RS485) Modbus register type
98,99	External (RS485) Modbus address
100	Alarm Configuration Version (latest version is 0x01)
101	Alarm Type
102,103	Alarm timeout
104-107	Alarm Deadband value (float)
108-111	Alarm Low Low value (float)
112-115	Alarm Low value (float)
116-119	Alarm High value (float)
120-123	Alarm High High value (float)
124	Low Low alarm enabled flag. 0 – disabled. 1 – enabled
125	Low alarm enabled flag. 0 – disabled. 1 – enabled
126	High alarm enabled flag. 0 – disabled. 1 – enabled
127	High High alarm enabled flag. 0 – disabled. 1 – enabled

128-187	Contacts ID which is bind to alarms. Please see structure of this data block below
188-199	Messages ID which is bind to alarms. Please see structure of this data block below
200	Scaling enable flag. 0 – scaling disabled. 1 – scaling enabled
201,202	Zero Level scale raw counts
203,204	Full Level scale raw counts
205-208	Zero Level output value (float)
209-212	Full Level output value (float)
213	External (RS485) Modbus register value type
214	External (RS485) Modbus register value byte order
215	Show on LCD flag. 0 – do not show on LCD, 1 – show on LCD

Contacts ID Data Block Structure (128 – 187 bytes)

This data block keeps Contact ID assigned to every possible alarm condition.
Totally firmware supports 6 alarm conditions:

HIHI
HI
NORMAL
LO
LOLO
VALUE CHANGED

Please see **8.2 Alarm Conditions** for more information.

Every Contact ID is unsigned 16 bits integer.
Every Alarm Condition can have up to 5 assigned contacts. For example when HIHI alarm condition detected the system will be able to send assigned Message to up to 5 different contacts.
If no contact assigned to Alarm Condition, then system uses special INVALID ID constant 0xFFFF
0xFFFF which means that Alarm Condition does not have assigned contacts.

Totally this data block contains 30 ID values.
If no contacts assigned all they will be 0xFFFF.

The table below show mapping of bytes and Alarm Conditions:

Byte #	Description
128,129	Contact ID #1 for HIHI
130,131	Contact ID #2 for HIHI
132,133	Contact ID #3 for HIHI
134,135	Contact ID #4 for HIHI
136,137	Contact ID #5 for HIHI
138,139	Contact ID #1 for HI
140,141	Contact ID #2 for HI
142,143	Contact ID #3 for HI
144,145	Contact ID #4 for HI
146,147	Contact ID #5 for HI

148,149	Contact ID #1 for NORMAL
150,151	Contact ID #2 for NORMAL
152,153	Contact ID #3 for NORMAL
154,155	Contact ID #4 for NORMAL
156,157	Contact ID #5 for NORMAL
158,159	Contact ID #1 for LO
160,161	Contact ID #2 for LO
162,163	Contact ID #3 for LO
164,165	Contact ID #4 for LO
166,167	Contact ID #5 for LO
168,169	Contact ID #1 for LOLO
170,171	Contact ID #2 for LOLO
172,173	Contact ID #3 for LOLO
174,175	Contact ID #4 for LOLO
176,177	Contact ID #5 for LOLO
178,179	Contact ID #1 for VALUE CHANGED
180,181	Contact ID #2 for VALUE CHANGED
182,183	Contact ID #3 for VALUE CHANGED
184,185	Contact ID #4 for VALUE CHANGED
186,187	Contact ID #5 for VALUE CHANGED

Messages ID Data Block Structure (188 – 199 bytes)

This data block keeps Message ID assigned to every Alarm Condition.
Only one message can be assigned to every Alarm Condition so this block keeps only 6 Message ID.
Every Message ID is unsigned 16 bits integer.
If no message assigned to Alarm Condition, then system uses special INVALID ID constant 0xFFFF
0xFFFF which means that Alarm Condition does not have assigned message.

The table below show mapping of bytes and Alarm Conditions:

Byte #	Description
188,189	Message ID assigned to Alarm Condition HIHI
190,191	Message ID assigned to Alarm Condition HI
192,193	Message ID assigned to Alarm Condition NORMAL
194,195	Message ID assigned to Alarm Condition LO
196,197	Message ID assigned to Alarm Condition LOLO
198,199	Message ID assigned to Alarm Condition VALUE CHANGED

5.3 Write Tag Configuration

Write single tag configuration parameters. See **Appendix G: Tag Configuration Description** for more information about Tag Configuration parameters.

Request

Byte #	Description
0	0x07 (Command ID)
1	Command Version (latest version is 0x03)
2,3	Tag ID
4	Tag type
5	Flag of logger state. 0 – this tag should not be logged. 1 – it should be logged
6	Flag of Values Map state. 0 – Values Map is not used. 1 – Values Map is used.
7-30	Tag name (always 24 bytes). Zero ended.
31,32	Internal Modbus register address to what tag is bind
33-36	Logging period in seconds
37-60	Units1 text
61-84	Units2 text
85	Equation type.
86-89	Equation parameter A float value
90-93	Equation parameter B float value
94	External (RS485) Modbus device RTU
95	External (RS485) Modbus register type
96,97	External (RS485) Modbus address
98	Alarm Configuration Version (latest version is 0x01)
99	Alarm Type
100,101	Alarm timeout
102-105	Alarm Deadband value (float)
106-109	Alarm Low Low value (float)
110-113	Alarm Low value (float)
114-117	Alarm High value (float)
119-121	Alarm High High value (float)
122	Low Low alarm enabled flag. 0 – disabled. 1 – enabled
123	Low alarm enabled flag. 0 – disabled. 1 – enabled
124	High alarm enabled flag. 0 – disabled. 1 – enabled
125	High High alarm enabled flag. 0 – disabled. 1 – enabled
126-185	Contacts ID which is bind to alarms. Please see structure of this data block below
186-197	Messages ID which is bind to alarms. Please see structure of this data block below
198	Scaling enable flag. 0 – scaling disabled. 1 – scaling enabled
199,200	Zero Level scale raw counts
201,202	Full Level scale raw counts
203-206	Zero Level output value (float)
207-210	Full Level output value (float)

211	External (RS485) Modbus register value type
212	External (RS485) Modbus register value byte order
213	Show on LCD flag. 0 – do not show on LCD, 1 – show on LCD

Response

Byte #	Description
0	0x07 (Command ID)
1,2	Error code (0 success)

NOTE: If Tag ID already exists in the internal tags list on the device, then the system will UPDATE information in tag structure.
If Tag ID is new, then new tag will be added.

5.4 Delete Tag

Delete specified tag.

Request

Byte #	Description
0	0x08 (Command ID)
1,2	Tag ID

Response

Byte #	Description
0	0x08 (Command ID)
1,2	Error code (0 success)

NOTE: All related objects like Log records, Contacts and Alarm Messages are not removed. Configuration client has to delete them itself otherwise they will be orphaned.

5.5 Delete All Tags

Delete all tags on device.

Request

Byte #	Description
0	0x16 (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x16 (Command ID)
1,2	Error code (0 success)

NOTE: All related objects like Log records, Contacts and Alarm Messages are not removed. Configuration client has to delete them itself otherwise they will be orphaned.

5.6 Read Values Map Item List

Read ID of all Values Map items. This command allow get list of ID of all Values Map items and then requests the item configuration, edit item or delete it using this ID value.

Request

Byte #	Description
0	0x2D (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x2D (Command ID)
1,2	Error code (0 success)
3	Number of ID in the list
4-N	Values Map Item ID List

NOTE: Values Map Item ID List is byte sequence. Every Values Map Item ID is unsigned 2 bytes integer. So number of bytes N depended on number of Values Map Items. For example:
- if number of Values Map Items is 1, then Values Map Item ID List will contain only 2 bytes.
- if number of Values Map Items is 10, then Values Map Item ID List will contain only 20 bytes.

5.7 Read Values Map Item Configuration

Read Values Map item configuration.

Request

Byte #	Description
0	0x2E (Command ID)
1,2	Values Map Item ID

Response

Byte #	Description
0	0x2E (Command ID)
1,2	Error code (0 success)
3	Command Version (latest version is 0x02)
4,5	Values Map Item ID
6-9	Tag ID (used only lowest 2 bytes)
10,11	Tag Raw Value
12	Replacement text length
13-N	Replacement text (without end zero byte)

This command does not have fixed size response. The length of response depends on the length of text.

Replacement text length – the number of characters in the replacement text.

5.8 Write Values Map Item Configuration

Write Values Map item configuration.

Request

Byte #	Description
0	0x2F (Command ID)
1	Command Version (latest version is 0x02)
2,3	Values Map Item ID
4-7	Tag ID
8,9	Tag Raw Value
10	Replacement text length
11-N	Replacement text (without end zero byte)

Response

Byte #	Description
0	0x2F (Command ID)
1,2	Error code (0 success)

NOTE: if Values Map Item ID found, then appropriate record will be updated. If Values Map Item ID is not found, then new item will be created and stored.

5.9 Delete Values Map Item

Delete one specified Values Map Item.

Request

Byte #	Description
0	0x30 (Command ID)
1,2	Values Map Item ID

Response

Byte #	Description
0	0x30 (Command ID)
1,2	Error code (0 success)

5.10 Delete All Values Map Items

Delete all Values Map Items.

Request

Byte #	Description
0	0x31 (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x31 (Command ID)
1,2	Error code (0 success)

5.11 Read Contact List

Read ID of all Contact records. This command allows getting a list of ID of all Contact records and then requests the Contact record configuration, edit or delete it using this ID value.

Request

Byte #	Description
0	0x1E (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x1E (Command ID)
1,2	Error code (0 success)
3	Number of ID in the list
4-N	Contact record ID List

NOTE: Contact record ID List is byte sequence. Every Contact record ID is unsigned 2 bytes integer. So number of bytes N depended on number of Contact records. For example:
- if number of Contact records is 1, then Contact record ID List will contain only 2 bytes.
- if number of Contact records is 10, then Contact record ID List will contain only 20 bytes.

5.12 Read Contact Configuration

Read Contact record configuration.

Request

Byte #	Description
0	0x1F (Command ID)
1,2	Contact record ID

Response

Byte #	Description
0	0x1F (Command ID)
1,2	Error code (0 success)
3	Command Version (latest version is 0x01)
4,5	Contact record ID
6-35	First Name (zero ended, always 30 bytes)
36-65	Last Name (zero ended, always 30 bytes)
66-85	Phone #1 (zero ended, always 20 bytes)
86-105	Phone #2 (zero ended, always 20 bytes)
106-145	Email #1 (zero ended, always 40 bytes)
146-185	Email #2 (zero ended, always 40 bytes)
185-205	Title (zero ended, always 20 bytes)

5.13 Write Contact Configuration

Write Contact record configuration.

Request

Byte #	Description
0	0x20 (Command ID)
1	Command Version (latest version is 0x02)
2,3	Contact record ID
4-33	First Name (zero ended, always 30 bytes)
34-63	Last Name (zero ended, always 30 bytes)
64-83	Phone #1 (zero ended, always 20 bytes)
84-103	Phone #2 (zero ended, always 20 bytes)
104-143	Email #1 (zero ended, always 40 bytes)
144-183	Email #2 (zero ended, always 40 bytes)
184-203	Title (zero ended, always 20 bytes)

Response

Byte #	Description
0	0x20 (Command ID)
1,2	Error code (0 success)

NOTE: if Contact record ID found, then appropriate record will be updated. If Contact record ID is not found, then new contact record will be created and stored.

5.14 Delete Contact

Delete one specified Contact record.

Request

Byte #	Description
0	0x21 (Command ID)
1,2	Contact record ID

Response

Byte #	Description
0	0x21 (Command ID)
1,2	Error code (0 success)

5.15 Delete All Contacts

Delete all Contact record.

Request

Byte #	Description
0	0x22 (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x22 (Command ID)
1,2	Error code (0 success)

5.16 Read Message List

Read ID of all Alarm Message records. This command allows getting a list of ID of all Alarm Message records and then requests the Alarm Message record configuration, edit or delete it using this ID value.

Request

Byte #	Description
0	0x23 (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x23 (Command ID)
1,2	Error code (0 success)
3	Number of ID in the list
4-N	Alarm Message record ID List

NOTE: Alarm Message record ID List is byte sequence. Every Alarm Message record ID is unsigned 2 bytes integer. So number of bytes N depended on number of Alarm Message records. For example:
- if number of Alarm Message records is 1, then Alarm Message record ID List will contain only 2 bytes.
- if number of Alarm Message records is 10, then Alarm Message record ID List will contain only 20 bytes.

5.17 Read Message Configuration

Read Alarm Message record configuration.

Request

Byte #	Description
0	0x24 (Command ID)
1,2	Alarm Message record ID

Response

Byte #	Description
0	0x24 (Command ID)
1,2	Error code (0 success)
3	Command Version (latest version is 0x01)
4,5	Alarm Message record ID
6-165	Alarm Message Text (zero ended, always 160 bytes)

5.18 Write Message Configuration

Write Alarm Message record configuration.

Request

Byte #	Description
0	0x25 (Command ID)
1	Command Version (latest version is 0x02)
2,3	Alarm Message record ID
4-163	Alarm Message Text (zero ended, always 160 bytes)

Response

Byte #	Description
0	0x25 (Command ID)
1,2	Error code (0 success)

NOTE: if Alarm Message record ID is found, then appropriate record will be updated. If Alarm Message record ID is not found, then new Alarm Message record will be created and stored.

5.19 Delete Message

Delete one specified Alarm Message record.

Request

Byte #	Description
0	0x26 (Command ID)
1,2	Alarm Message record ID

Response

Byte #	Description
0	0x26 (Command ID)
1,2	Error code (0 success)

5.20 Delete All Messages

Delete all Alarm Message records.

Request

Byte #	Description
0	0x27 (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x27 (Command ID)
1,2	Error code (0 success)

5.21 Read Live Value of Tag

Read live value of specified tag.

Request

Byte #	Description
0	0x39 (Command ID)
1-4	Tag ID (valid only lowest 2 bytes)

Response

Byte #	Description
0	0x39 (Command ID)
1,2	Error code (0 success)
3-6	Raw tag value as unsigned long. If real tag value type is different, then it should be cast to valid type
7-10	Calculated tag value as float value

NOTE: If Error code is not zero, then this command will not have bytes 3-10. Bytes 3-10 presented only when Error Code is 0.

5.22 Write Live Value of All Tags

Read live value of all tags which are added to the system.

Request

Byte #	Description
0	0x45 (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x45 (Command ID)
1,2	Error code (0 success)
3	Value record count
4-N	Tag value records

Tag value Record Format

Byte #	Description
0	Value validity flag
1-4	Tag ID (used only 2 lowest bytes)
5-9	Raw tag value as unsigned long. If real tag value type is different, then it should be cast to valid type
10-13	Calculated tag value as float value

Value validity flag

Byte #	Description
0	Tag value was not read yet. Usually happens when tag's value is requested when logger is stopped and firmware does not update tags
1	Tag value is invalid. This means that when logger tried to update tag value error occurred and value was not read.
2	Tag value is valid.

6. Logger Commands

6.1 Set Logger State

Set logger state. Logger can be started or stopped.
When logger stopped the system does not update tags value, does not check alarm conditions and does not log tag values to DATAFLASH / SD card.

Request

Byte #	Description
0	0x17 (Command ID)
1	Logger state (please see **8.12 Logger States** for more information)

Response

Byte #	Description
0	0x17 (Command ID)
1,2	Error code (0 success)

6.2 Get Logger State

Request

Byte #	Description
0	0x17 (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x17 (Command ID)
1,2	Error code (0 success)
3	Logger state (please see **8.12 Logger States** for more information)

6.3 Initialize Log Reading

This command informs the system about starting log reading.

Request

Byte #	Description
0	0x09 (Command ID)
1-4	First Record ID to read

Response

Byte #	Description
0	0x09 (Command ID)
1,2	Error code (0 success)

As First Record ID parameter client should send ID of first record to read. It is unsigned long value.
Each log record has unique ID. All records stored on DATAFLASH / SD card in raw data format. The filesystem is not used in order to avoid unwanted write/read operations and RAM usage efficiency.
Log has 3 types of records: DATA, ALARM, and EVENT.

DATA – this record keep tag values.
ALARM – this record keeps alarm information when it is generated.
EVEN – this record keep informational and diagnostic information.

All records placed in the same area so can be read one by one.
For now system does not support filtering of read records. Client can only request all records if specify 0 as First Record ID parameter or new records from last read, if specify last record ID + 1 as First Record ID parameter.

So logic should be following:

1st read – set First Record ID to 0. This will initial log reading from very first record.
When records read remember ID of last record.
To read new records set First Record ID to remembered ID + 1.

When client send this command the system does:
- Flash all in RAM buffers to DATAFLASH / SD card.
- Search specified Record ID parameter in the log.

6.4 Read Log Records

Read next batch of log records from device. Log reading should be initialized with previous command otherwise error or incorrect data will be returned.

Request

Byte #	Description
0	0x0A (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x0A (Command ID)
1,2	Error code (0 success)
3	Number of log records in response
4	Log record version (latest supported version is 1)
5-N	Log record list

This command can return 0 or more records. If records read without errors the Error Code field will be 0.
If during log reading end of log detected (no more records), then Error Code will be set to 1013 (reached end of log).

Log Record Format

Byte #	Description
0-3	Record ID (unsigned long value)
4,5	Year (as full year like 2014)
6	Month (1..12)
7	Day (1..31)
8	Hour (0..23)
9	Minutes (0..59)
10	Seconds (0..59)
11	Type of record (see **8.15 Data Record Types** for more information)
12,13	Tag ID (as unsigned integer)
14-23	Please see notes below
24	CRC8. Please see **Appendix H: Calculating CRC8**

NOTES: Depending on log record type in byte #11 bytes 14-23 can be parsed in different way.

DATA Log Record

Byte #	Description
14-17	Raw Tag Value. For all tag type except MODBUS this is unsigned long / signed long value. Client has to decide how to interpret this value himself. For tag with type MODBUS and Modbus Value Type set to FLOAT this will be float value.
18-21	Calculated Tag Value (as float)
22,23	Not used

ALARM Log Record

Byte #	Description
14-17	Raw Tag Value. For all tag type except MODBUS this is unsigned long / signed long value. Client has to decide how to interpret this value himself. For tag with type MODBUS and Modbus Value Type set to FLOAT this will be float value.
18-21	Calculated Tag Value (as float)
22,23	Alarm Condition (see **8.2 Alarm Conditions** for more information)

EVENT Log Record

Byte #	Description
14-17	Event ID (see **8.18 Event ID** for more information)
18-21	Error Code
22,23	Event Type (see **8.10 Event Types** for more information)

6.5 Clear Log

Clear all logged records. Actually it just reset internal pointers and record counters to zero position.

Request

Byte #	Description
0	0x0B (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x0B (Command ID)
1,2	Error code (0 success)

7. Communication Commands

7.1 Read APN Configuration

Request

Byte #	Description
0	0x2A (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x2A (Command ID)
1,2	Error code (0 success)
3	APN Configuration Format (latest supported version 0x01)
4-67	APN server name (zero ended, always 64 bytes)
68-99	APN user name (zero ended, always 32 bytes)
100-131	APN password (zero ended, always 32 bytes)

7.2 Write APN Configuration

Request

Byte #	Description
0	0x2B (Command ID)
1	APN Configuration Format (latest supported version 0x01)
2-65	APN server name (zero ended, always 64 bytes)
66-97	APN user name (zero ended, always 32 bytes)
99-129	APN password (zero ended, always 32 bytes)

Response

Byte #	Description
0	0x2B (Command ID)
1,2	Error code (0 success)

7.3 Read Data Pushing Configuration

Request

Byte #	Description
0	0x32 (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x32 (Command ID)
1,2	Error code (0 success)
3	Data Push Configuration Format (latest supported version 0x01)
4	Data Pushing Enabled flag. 0 – disabled, 1 – enabled
5,6	TCP Port
7,8	Pushing Period in minutes
9,10	Pushing Period Offset in minutes
11	Protocol (see **8.7 Data Pushing Protocols** for more information)
12	Format (see **8.6 Data Pushing Output Format** for more information)
13-140	Remote server URL (zero ended, always 128 bytes)
141-268	Remote server login (zero ended, always 128 bytes)
269-396	Remote server password (zero ended, always 128 bytes)

7.4 Write Data Pushing Configuration

Request

Byte #	Description
0	0x33 (Command ID)
1	Data Push Configuration Format (latest supported version 0x01)
2	Data Pushing Enabled flag. 0 – disabled, 1 – enabled
3,4	TCP Port
5,6	Pushing Period in minutes
7,8	Pushing Period Offset in minutes
9	Protocol (see **8.7 Data Pushing Protocols** for more information)
10	Format (see **8.6 Data Pushing Output Format** for more information)
11-138	Remote server URL (zero ended, always 128 bytes)
139-266	Remote server login (zero ended, always 128 bytes)
267-394	Remote server password (zero ended, always 128 bytes)

Response

Byte #	Description
0	0x33 (Command ID)
1,2	Error code (0 success)

7.5 Read SMTP Configuration

Request

Byte #	Description
0	0x37 (Command ID)
1	Blank byte 0x00

Response

Byte #	Description
0	0x37 (Command ID)
1,2	Error code (0 success)
3	SMTP Configuration Format (latest supported version 0x01)
4,5	TCP Port
6-133	SMTP server URL (zero ended, always 128 bytes)
134-261	SMTP server login (zero ended, always 128 bytes)
262-389	SMTP server password (zero ended, always 128 bytes)

7.6 Write SMTP Configuration

Request

Byte #	Description
0	0x38 (Command ID)
1	SMTP Configuration Format (latest supported version 0x01)
2,3	TCP Port
4-131	SMTP server URL (zero ended, always 128 bytes)
132-259	SMTP server login (zero ended, always 128 bytes)
260-387	SMTP server password (zero ended, always 128 bytes)

Response

Byte #	Description
0	0x38 (Command ID)
1,2	Error code (0 success)

8. Constants

8.1 ADC Channels

Value	Description
0	AIN1 analog input
1	AIN2 analog input

8.2 Alarm Conditions

Value	Description
0	Unknown. Means no alarm
1	HIHI alarm condition
2	HI alarm condition
3	NORMAL alarm condition
4	LO alarm condition
5	LOLO alarm condition
6	VALUE CHANGED alarm condition

8.3 Alarm Types

Value	Description
0	Unknown
1	LIMIT
2	VALUE CHANGED

8.4 Byte Orders

Value	Description
0	No swap
1	Byte and word swap
2	Byte swap
3	Word swap

Byte order explanation

When read some float or long value in Modbus request they presented as sequence of 4 bytes. To convert them to valid number client has to know byte order to know on what place put each byte.

Example:

Bytes in response (from first to last).

0x01 0x02 0x03 0x04

Where 0x01 – first received byte and 0x04 last received byte.

Now look what UNSIGNED LONG value we can get for different byte orders:
NOTE: We write lowest byte at first (at less address).

No swap

HEX: 0x01020304
DEC: 16909060

Byte and word swap

HEX: 0x04030201
DEC: 67305985

Byte swap

HEX: 0x02010403
DEC: 33620995

Word swap

HEX: 0x03040102
DEC: 50594050

8.5 Modem Types

Value	Description
0	Unknown modem model
1	WPM50 2G Modem
2	HE-910D 3G Modem

8.6 Data Pushing Output Format

Value	Description
0	Custom Format
1	BiPOM
2	New Boundary

Custom format used for producing special data format.
This format is closed and used only for some customers.

BiPOM format can be used for uploading data over HTTP to BiPOM Web portal.
To do this should be provided account information of user registered on BiPOM Web portal www.nanowipom.com

Note, this format described in **WRTU Push Data Protocol** document at address:

www.bipom.com/documents/WiPOM/WRTU%20Push%20Data%20Protocol.pdf

Since it is simple JSON protocol any user can implement its own web server and upload data there using this protocol.

New Boundary format can be used for uploading data to New Boundary web portal over HTTP.
Read more here:

www.newboundary.com

BiPOM works on adding support for more backend servers which will be available soon.

8.7 Data Pushing Protocols

Value	Description
0	Undefined
1	HTTP
2	HTTPS
3	FTP

NOTE: This defines how formatted data should be sent from device to remote server
Not all combinations of Format / Protocol allowed. Please see table below:

Protocol / Format	Custom	BiPOM	New Boundary
HTTP	Not public		
HTTPS	Not public		
FTP	Not public		

Green cell – allowed combination.
Red cell – not allowed combination.
Not public – closed protocol used. Not for public use.

IMPORTANT: At current version of the system only CG-nanoWiPOM system supports Data Pushing.

8.8 Device Reset Codes

Value	Description
0	Unknown source of system restart
1	Power on reset
2	**A low level on the Reset pin**
3	**Low-power management reset**
4	**Independent watchdog end of count condition**
5	**Window watchdog end of count condition**
6	**Software reset**

8.9 Equation Types

Value	Description
0	Do not use calculation
1	Use linear equation (A *X + B)
2	**Use Quadratic equation (A * X ^ B)**

A, B – parameters set by client.
X – raw tag value.

8.10 Event Types

Value	Description
0	Error
1	Warning
2	Information
3	**Alarm**

8.11 Hardware Types

Value	Description
-1	Unknown
0	nanoWiPOM
1	WiPOM
2	**CG-nanoWiPOM**

8.12 Logger States

Value	Description
0	Stopped
1	Started

8.13 Modbus Register Types

Value	Description
0	Digital Output
1	Digital Input
2	Analog Input
3	Holding Register

8.14 Modbus Register Value Types

Value	Description
0	16 bits integer
1	16 bits unsigned integer
2	32 bits integer
3	32 bits unsigned integer
4	32 bits float

8.15 Data Record Types

Value	Description
0	Data record
1	Event record
2	Alarm record

8.16 Tag Types

Value	Description
255	Unknown
0	Analog Input
1	Analog Input RMS
2	Digital Input
3	Digital Input Frequency
4	Digital Output
5	Holding Register
6	External Modbus
7	TLC2543 Analog Input
8	Digital Input Counter

8.17 Tag Value States

Value	Description
0	Not read
1	Not valid
2	Valid

8.18 Event ID

Code	Description	Data Field
0	None event	Not used
1	System started	Restart Code
2	Storage devices initialization failed	
3	Cell driver initialization	
4	RS485 driver initialization	
5	UART0 driver initialization	
6	DIO subsystem initialization	
7	ADC initialization	
8	Display initialization	
9	Initializing GPRS layer on cell modem	
10	Closing GPRS layer on cell modem	
12	GPRS TCP server starting	
13	GPRS TCP server stopping	
14	GPRS modem hardware reset	
15	Save device configuration to disk failed (inside protocol command)	
16	USB driver initialization	

Code	Description	Data Field
17	Add tag	Tag ID
18	Update tag	Tag ID
19	Load tag	Tag ID
20	UART2 initialization	
21	UART3 initialization	
22	TLC2543 initialization	
23	Add contact	Contact ID
24	Load contacts failed	
25	Load device configuration failed	
26	Load tags failed	
27	Save tags failed	
28	Add Alarm Message Record	Message ID
29	Load Alarm Message Record failed	
30	Save Alarm Message Record failed	
31	Load tag failed	
32	Cell Communication was enabled	
33	Cell Communication was disabled	
34	Configuration Storage Device Id	Id of device where stored configuration
35	Log Data Storage Device Id	Id of device where stored log records
36	Alarm SMS sending failed (after 5 attempts)	
37	Alarm SMS sent successfully	Not used
38	Initialize DI in COUNTER mode failed	
39	Start DI COUNTER failed	
40	Stop DI COUNTER failed	
41	Detecting cell modem	
42	Alarm email sending failed	
43	Alarm email sent successfully	Not used
44	Load device configuration failed	
45	Load APN configuration failed	
46	Load peripheral devices configuration failed	
47	Load pushing data configuration failed	
48	Load SMTP configuration failed	
49	Load internal counters failed	
50	Cell modem re-initialized	
51	Modem model	Modem Model ID
52	Modem read data timed out	
53	TCP client stopped on cell modem	
54	TCP client started on cell modem	
55	TCP connection closed by peer	
56	Test incoming TCP connection failed	

Code	Description	Data Field
57	TCP server closed	
58	TCP server opened	
59	Cell network status	Cell Network Status
60	Cell network status read	
61	Cell network registration state	Registration Code
62	Cell network registration state read	
63	SIM card error	SIM Error Code
64	SIM card status code	SIM Status Code
65	SIM card status read	
66	CSQ level in dBm	Signal Strength
67	CSQ status as raw value from modem	Signal Strength
68	CSQ level read	
69	GPRS layer closed	
70	GPRS layer opened	
71	GPRS configured	
72	Modem status updated	
73	Error code of last AT modem command	AT Command Error
74	Software reset of cell modem	0 – success 1 – failed
75	Hardware reset of cell modem	0 – success 1 – failed
76	Cell modem write data timed out	
77	SMTP error code	Error from SMTP server
78	Software restarted by user request	
79	Firmware revision	Firmware Revision

NOTE: Many events have additional value in Data field of Event Log Record.
If event has type **Error** that data field keeps **Error Code**.
All informational events can have extra data in Data field.
If 3rd column does not have description this means that it will have Error Code which was returned as result of event operation.

9. Samples

9.1 Read device configuration

To read device configuration client has to do:

- Connect to device using any suitable communication port (USB serial port, UART0 download serial port or Cell Modem).
- For USB/UART0 should be used Modbus RTU, for Cell Modem – Modbus TCP.
- Send following sequence of commands:
    - Read Device information.
    - Get Device Features.
    - Read ADC Calibration Information.
    - Read RS485 Port Configuration.
    - Read RS232 Port Configuration.
    - Read LCD Configuration.
    - Read Tag List.
        - Read Tag Configuration for every Tag ID in Tag ID List.
    - Read Values Map Item List.
        - Read Values Map Item Configuration for every Values Map Item ID in ID List.
    - Read Contact List.
        - Read Contact Configuration for every Contact ID in ID List.
    - Read Message List.
        - Read Message Configuration for every Message ID in ID List.
    - Get Logger State.
    - Read APN Configuration.
    - Read Data Pushing Configuration.
    - Read SMTP Configuration.
- Disconnect from device.

After this client will have full device configuration.
Note, it is not required to read whole configuration to edit it. Client can read only required parts of configuration to edit them and then write back.

9.2 Write new device configuration

To write device configuration client has to do:

- Connect to device using any suitable communication port (USB serial port, UART0 download serial port or Cell Modem).
- For USB/UART0 should be used Modbus RTU, for Cell Modem – Modbus TCP.
- Send following sequence of commands:
    - Write Device information.
    - Set Device Features.
    - Write RS485 Port Configuration.
    - Write RS232 Port Configuration.
    - Write LCD Configuration.
    - Write Tag Configuration for every tag (new or edited).
    - Write Values Map Item Configuration for every Values Map Item (new or edited).
    - Write Contact Configuration for every Contact (new or edited).
    - Write Message Configuration for every Message (new or edited).
    - Write APN Configuration.
    - Write Data Pushing Configuration.
    - Write SMTP Configuration.
    - Set Logger State.
- Disconnect from device.

After this device will have new configuration.
Note, it is not required to write whole configuration at once. Client can write only required parts of configuration.

9.3 Read logged data

The most important part is reading logged data from device.

To initialize log reading client has to issue **Initialize Log Reading** command.
When client send this command the system does:

- Pass as parameter First Record ID from which start to read log. It can be 0 to start from begin.
- Flash all in RAM buffers to DATAFLASH / SD card.
- Search specified Record ID parameter in the log.

So after **Initialize Log Reading** command the system store pointer to first record from which it will start to read log.
To read log records client has to send one or several **Read Log Records** command(s).
This command requests the device to return as much as possible records from log to the client.
Since internal communication buffer size is limited this command cannot return all records if log has a lot of records. In response of this command the system returned number of actually read records and also Error Code. Client can use Error Code to detect when all log records read and internal pointer reaches the last record. In this case Error Code will be set to value 1013. Client can stop reading at any moment and does not wait on this stop marker.

IMPORTANT: The system remembers last state of log reading so to start read log correctly always issue **Initialize Log Reading** command in first turn.

Appendix A: Internal Modbus Registers

Digital Inputs

Code	Description
10001	Digital Input #1
10002	Digital Input #2
10003	Digital Input #3
10004	Digital Input #4

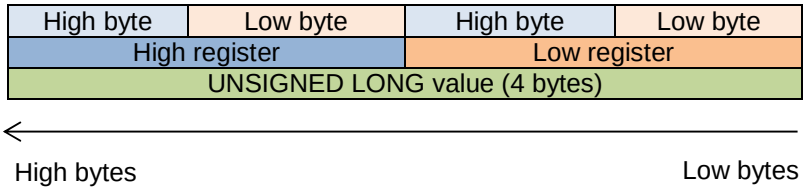
Digital Inputs (Frequency Counter)

Code	Description
40113	Digital Input #1
40114	Digital Input #2

Digital Inputs (Pulses Counter)

Code	Description
40122	Digital Input #1 (High 2 bytes of LONG value)
40123	Digital Input #1 (Low 2 bytes of LONG value)
40124	Digital Input #2 (High 2 bytes of LONG value)
40125	Digital Input #2 (Low 2 bytes of LONG value)
40126	Digital Input #3 (High 2 bytes of LONG value)
40127	Digital Input #3 (Low 2 bytes of LONG value)
40128	Digital Input #4 (High 2 bytes of LONG value)
40129	Digital Input #4 (Low 2 bytes of LONG value)

NOTE: Bytes placed in following order:



For example, read values for DI #1 in Pulses Counter mode.
Register #40122 returns 0xA102.
Register #40123 returns 0x0C15.
The result unsigned long value will be 0xA1020C15.

Digital Outputs

Code	Description
1	Digital Output #1
2	Digital Output #2

Analog Outputs

Code	Description
30001	Analog Input #1 (AN2)
30002	Analog Input #2 (AN3)
30003	Analog Input #3 (PC3)
30004	Analog Input #4 (Battery voltage)
30005	Analog Input #5 (5V)

Analog Outputs (RMS mode)

Code	Description
30006	Analog Input #1 (AN2)
30007	Analog Input #2 (AN3)

DAQ-2543 Inputs

Code	Description
30008	Channel #0
30009	Channel #1
30010	Channel #2
30011	Channel #3
30012	Channel #4
30013	Channel #5
30014	Channel #6
30015	Channel #7
30016	Channel #8
30017	Channel #9
30018	Channel #10

Holding Registers

Code	Description
40001	Hardware Id. 0 – nanoWiPOM, 1 – WiPOM, -1 – unknown device
40100	RTU number
40101	RTC - Year
40102	RTC - Month
40103	RTC - Day
40104	RTC - Hour
40105	RTC - Minutes
40106	RTC - Seconds
40107	Reboot Source Code. 1 - POR (Power-On-Reset) 2 - External Pin Reset 3 - Low Power Reset 4 - Watchdog Reset 5 - Window Watchdog Reset 6 - Software Reset
40108	Last System Error Code (See Appendix B for list of possible errors)
40109	Uptime in seconds (low word)
40110	Uptime in seconds (high word)
40111	Hardware Status. Please see Note A.1 below.
40112	Firmware Version. Please see Note A.2 below.
40115	Total Modbus Errors (counter reset on system start)
40116	Total Modbus Exceptions (counter reset on system start)
40117	Total DATAFLASH Write Errors (counter reset on system start)
40118	Total DATAFLASH Read Errors (counter reset on system start)
40119	Total SD card Write Errors (counter reset on system start)
40120	Total SD card Read Errors (counter reset on system start)
40121	Cell Modem Type 0 – Unknown 1 – WMP50 (2G) 2 – HE910D (3G) 3 – CloudGate (nanoWiPOM used as peripheral board on Cloudgate device)

Read-only Registers

Note A.1: Hardware Status

Bit #	Description
0	Core system initialized
1	UART0 driver initialized
2	UART1 driver initialized
3	UART2 driver initialized
4	UART3 driver initialized
5	USB driver initialized
6	IO driver initialized
7	ADC driver initialized
8	UART0 service initialized
9	RS485 service initialized
10	Cell modem service initialized
11	USB service initialized
12	Display driver initialized
13	DAQ2543 driver initialized
14	Cell modem detected

If bit set to 0 – it means that device/driver/subsystem was not initialized and some issues were detected.
If bit set to 1 – it means that device/driver/subsystem was initialized.

Note A.2: Firmware Version

Firmware version coded as integer value.
It contains 2 parts – major version number and minor version number.
To get correct version it should be processed as:

Major = VALUE / 100
Minor = VALUE % 100

/ - integer dividing
% - dividing by module

For example, register returned value 211 (decimal).
This means:

Major = 211 / 100 = 2
Minor = 211 % 100 = 11

Version = 2.11

Appendix B: Error codes

General

Internal errors (not shown on LCD).

Code	Description
1	Invalid Argument. Some function received wrong input data.
2	Queue is empty when trying to read next alarm/event from queue
3	Alarm/event queue is locked
4	Timeout on wait operation

Application Subsystem

Code	Description
101	Starting main task failed
102	Starting cell modem task failed
103	Starting RS485 server task failed
104	Starting USB server task failed
105	Starting UART0 server task failed
106	Starting data logger task failed

Configuration Subsystem

Code	Description
201	No space for new tags
202	Load device configuration failed
203	Load tag's configuration failed
204	No space for new contact record
205	Invalid contact record ID
206	Invalid contact record index
207	Unsupported version of contact record
208	Load contact records failed
209	Load message records failed
210	No space for new message record
211	Invalid message record ID
212	Invalid message record index
213	Unsupported version of message record
214	Unsupported version of tag configuration

Core Modbus Subsystem

Code	Description
401	Unsupported register address
402	Unknown function ID
403	RTU is incorrect in reply
404	Function ID is incorrect in reply
405	CRC16 is incorrect in reply

Modbus RTU Subsystem

Code	Description
501	Request data is not complete (broken)
502	Too big request data (no space in internal buffer)
503	Unknown function ID
504	Bad CRC16 of MODBUS request/reply
505	Processing MODBUS request failed

Modbus TCP Subsystem

Code	Description
601	Request data is not complete (broken)
602	Too big request data (no space in internal buffer)
603	Unknown function ID
604	Processing MODBUS request failed
605	Unknown Protocol ID
606	Length in Modbus TCP header is incorrect

RS485 Driver

Code	Description
701	RS485 driver is not initialized
702	Failed to clear RX buffer
703	Failed to clear TX buffer
704	Read Timeout
705	Write Timeout
706	No data to read
707	Echo not received

Cell Modem Driver

Code	Description
801	Cell Modem Driver is not initialized
802	Failed to clear RX buffer
803	Failed to clear TX buffer
804	Read Timeout
805	Write Timeout
806	Read operation didn't return any data
807	No incoming TCP connection
808	No reply for AT command
809	Unknown reply for AT command
810	Data to read is available

Hardware Subsystem

Code	Description
901	Hardware subsystem not initialized
902	DAQ2543 / ADC self-test failed
903	DAQ2543 / ADC not initialized
904	Incorrect ADC channel number
905	Incorrect DI pin number
906	Incorrect DO pin number
907	Write DO failed
908	Read DI failed
909	LCD Contrast adjustment failed
910	LCD Backlight adjustment failed
911	LCD initialization failed
912	Incorrect LCD line number
913	Configure Cell Modem input port failed
914	DATAFLASH initialization failed
915	SD card initialization failed
916	No storage device detected (no SD card and no DATAFLASH)
917	Incorrect frequency channel number
918	Incorrect DAQ2543 channel number

Data Storage Subsystem

Code	Description
1001	Unknown storage device Id
1002	Storage device not initialized
1003	Version structure has incorrect signature byte
1004	Version structure has incorrect version byte
1005	Version structure has incorrect LRC byte
1006	Data block has incorrect CRC16
1007	No data available to read
1008	Deep data check failed (read data is different from wrote data)
1009	Read operation failed
1010	Write operation failed
1011	Find structure not initialized
1012	Search action already started
1013	Find end of log records
1014	Sector number is incorrect

UART1 Driver

Code	Description
1101	UART1 driver is not initialized
1102	Failed to clear RX buffer
1103	Failed to clear TX buffer
1104	Read Timeout
1105	Write Timeout
1106	No data to read

USB Driver

Code	Description
1201	UART1 driver is not initialized
1202	Failed to clear RX buffer
1203	Failed to clear TX buffer
1204	Read Timeout
1205	Write Timeout
1206	No data to read

Communication Protocol

Code	Description
1301	Unknown command received

Appendix C: Event codes

Code	Description
0	None event
1	System started
2	Storage devices initialization failed
3	Cell driver initialization
4	RS485 driver initialization
5	UART0 driver initialization
6	DIO subsystem initialization
7	ADC initialization
8	Display initialization
9	Initializing GPRS layer on cell modem
10	Closing GPRS layer on cell modem
12	GPRS TCP server starting
13	GPRS TCP server stopping
14	GPRS modem hardware reset
15	Save device configuration to disk failed (inside protocol command)
16	USB driver initialization
17	Add tag
18	Update tag
19	Load tag
20	UART2 initialization
21	UART3 initialization
22	TLC2543 initialization
23	Add contact
24	Load contacts failed
25	Load device configuration failed
26	Load tags failed
27	Save tags failed
28	Add Alarm Message Record
29	Load Alarm Message Record failed
30	Save Alarm Message Record failed
31	Load tag failed
32	Cell Communication was enabled
33	Cell Communication was disabled
34	Configuration Storage Device ID (Id of device where stored configuration)
35	Log Data Storage Device ID (Id of device where stored log records)
36	Alarm SMS sending failed (after 5 attempts)
37	Alarm SMS sent successfully
38	Initialize DI in COUNTER mode failed
39	Start DI COUNTER failed
40	Stop DI COUNTER failed

Code	Description
41	Detecting cell modem
42	Alarm email sending failed
43	Alarm email sent successfully
44	Load device configuration failed
45	Load APN configuration failed
46	Load peripheral devices configuration failed
47	Load pushing data configuration failed
48	Load SMTP configuration failed
49	Load internal counters failed
50	Cell modem re-initialized
51	Modem model
52	Modem read data timed out
53	TCP client stopped on cell modem
54	TCP client started on cell modem
55	TCP connection closed by peer
56	Test incoming TCP connection failed
57	TCP server closed
58	TCP server opened
59	Cell network status
60	Cell network status read
61	Cell network registration state
62	Cell network registration state read
63	SIM card error
64	SIM card status code
65	SIM card status read
66	CSQ level in dBm
67	CSQ status as raw value from modem
68	CSQ level read
69	GPRS layer closed
70	GPRS layer opened
71	GPRS configured
72	Modem status updated
73	Error code of last AT modem command
74	Software reset of cell modem
75	Hardware reset of cell modem
76	Cell modem write data timed out
77	SMTP error code
78	Software restarted by user request
79	Firmware revision

Appendix D: Calculating Modbus CRC16

```
static const unsigned char aucCRCHi[] = {
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40
};


static const unsigned char aucCRCLo[] = {
    0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7,
    0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E,
    0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9,
    0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC,
    0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
    0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32,
    0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D,
    0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38,
    0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF,
    0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
    0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1,
    0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4,
    0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB,
    0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA,
    0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
    0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0,
    0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97,
    0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E,
    0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89,
    0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
    0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83,
    0x41, 0x81, 0x80, 0x40
};
```

```
unsigned int CRC16(unsigned char *pData, int length)
{
    unsigned char ucCRCHi = 0xFF;
    unsigned char ucCRCLo = 0xFF;
    int           idx;

    while( length-- )
    {
        idx = ucCRCLo ^ *( pData++ );
        ucCRCLo = ( UBYTE )( ucCRCHi ^ aucCRCHi[idx] );
        ucCRCHi = aucCRCLo[idx];
    }

    return ( unsigned int )( ucCRCHi << 8 | ucCRCLo );
}
```

Appendix F: Calculating DATAFLASH and SD card CRC16

```
unsigned short CRC16(unsigned char *pData,
      unsigned int length,
      unsigned int startCrc16Value)
{
      unsigned short crc_16_table[16] =
      {
            0x0000, 0xCC01, 0xD801, 0x1400,
            0xF001, 0x3C00, 0x2800, 0xE401,
            0xA001, 0x6C00, 0x7800, 0xB401,
            0x5000, 0x9C01, 0x8801, 0x4400
      };

      unsigned short crc = startCrc16Value;
      unsigned short r;

      while (length-- > 0)
      {
            r = crc_16_table[crc & 0xF];
            crc = (crc >> 4) & 0x0FFF;
            crc = crc ^ r ^ crc_16_table[*pData & 0xF];

            r = crc_16_table[crc & 0xF];
            crc = (crc >> 4) & 0x0FFF;
            crc = crc ^ r ^ crc_16_table[(*pData >> 4) & 0xF];

            pData++;
      }

      return crc;
}
```

Appendix G: Protocol Limits

The protocol has following limits:

Parameter	Limit
Maximum number of tag	40
DATAFLASH / SD card sector size	512 bytes
Device name length	32 bytes (31 characters + zero byte)
Tag name length	24 bytes (23 characters + zero byte)
Tag units text length	24 bytes (23 characters + zero byte)
Contact First Name length	30 bytes (29 characters + zero byte)
Contact Last Name length	30 bytes (29 characters + zero byte)
Contact Phone Number length	20 bytes (19 characters + zero byte)
Contact Email length	40 bytes (39 characters + zero byte)
Contact Title length	10 bytes (19 characters + zero byte)
Maximum Contacts number	100
Maximum Contacts per Alarm Condition	5
Alarm Message length	160 bytes (159 characters + zero byte)
Values Map Replacement Text length	64 bytes (63 characters + zero byte)
APN Server name length	64 bytes (63 characters + zero byte)
APN Username length	32 bytes (31 characters + zero byte)
APN Password length	32 bytes (31 characters + zero byte)
Data Pushing URL length	128 bytes (127 characters + zero byte)
Data Pushing Login length	128 bytes (127 characters + zero byte)
Data Pushing Password length	128 bytes (127 characters + zero byte)
SMTP Server URL length	128 bytes (127 characters + zero byte)
SMTP Login URL length	128 bytes (127 characters + zero byte)
SMTP Password URL length	128 bytes (127 characters + zero byte)

Appendix G: Tag Configuration Description

This section describes Tag Configuration Parameters in details.

Tag ID
This is unique tag identifier on the system. It is just a number from 1 to 65535.
System does not track this value itself so configuration client has to check that every new tag written to the system has unique value. So the correct sequence to add new tag is:

- Read tag id list.
- Find next not used ID (it can be as simple as plus 1 to largest ID in the list).
- When write new tag use this new unique ID.

Note, many things in configuration and also log records bind to tag ID value.
So when configuration client edit tag it has to keep Tag ID as it was read from device otherwise all tag's bind items will be lost.

Tag type

This filed uses to group tags by its type.
Please see **8.16 Tag Types** for list of supported types.
It is important to set correct Tag Type to help logger select correct way to read assigned Modbus address.
Also tag values differently shown on LCD depending on its type.

Logger State Flag
This field used by the system to decide should this tag be logged to DATAFLASH / SD card or just used for process alarm conditions.

If this flag set to 0 this means that the system should not log tag value. Only alarm condition will be checked on every new value update.

If this flag set to 1 this means that the system should log tag value using **Log Period** field value.

Logging Period
This field define how often tag value should be logged to DATAFLASH / SD card.
Tag value updated continuously (few times per second).
But the system may want to log value once per 15 minutes, 1 hour or any other period.
Configuration client can set this period in this field in seconds. The largest period is 86400 what is equal to 1 day. So 1 day this is largest log period.

Values Map Flag
This field used by the system to determine if tag value displayed on LCD or inserted to Alarm Message as raw Value or appropriate text replacement should be taken from Values Map.

If this flag set to 0 this means that Values Map is not used.
If this flag set to 1 this means that Values Map is used. In this case the system takes raw tag value and does lookup in Values Map.
If record exists for specified Tag ID and Tag Raw Value in Values Map, then the system takes the replacement text from Values Map record and shows it on LCD or inserts it in Alarm Message.

Tag name
Just a name of a tag. It is always zero ended and can be maximum 23 characters length (24 byte used for ending 0).

Internal Modbus Register Address
This field keeps Modbus register address from what the system should read tag value.
Note, if Tag Type set to **WRTU_TAG_TYPE_MB** (6) this field will be ignored because data should be read from external Modbus Device connected to RS485 serial port.

Units1
This field keep text of tag value units like Hz, V, mA, uA, Watt, etc.
If tag has type **WRTU_TAG_TYPE_DI,** then this text is used as replacement for CLOSED state of input.

Units2
This field keeps text replacement for OPEN state of digital input.

Equation Type
Please see **8.9 Equation Types** for list of all possible types.
This field defines if tag value should be calculated using one of selected equation.
Each tag has RAW value and CALCULATED value.
RAW value means value read from Modbus register as it is.
CALCULATED value presented result of calculation if some Equation Type is selected.
If Equation Type is not selected (set to default WRTU_EQUATION_RAW), then CALCULATION value is the same as RAW value.

Equation Parameter A and B
These fields keep parameters for equation. Both fields are float 4 bytes values.

External Modbus Device RTU
If tag type set as WRTU_TAG_TYPE_MB, then the system will read tag value from external Modbus device connected to RS485 port. In this case device should know some parameters as:

- External device RTU.
- External Device register Address.
- External Device register value type, like unsigned/signed int, unsigned/signed long, float.
- External Device register value byte order (only for long/float values).

This field keeps external Modbus device RTU as number 1..255.

External Modbus Register Type
This field keeps external Modbus device register type. Please see **8.13 Modbus Register Types** for full list of possible values.

External Modbus Address
This field keeps external Modbus device register address.

External Modbus Register Value Type
This field keeps external Modbus device register value type. Please see **8.14 Modbus Register Value Types** for full list of possible values.

External Modbus Register Value Byte Order
This field keeps external Modbus device register value byte order. Please see **8.4 Byte Orders** for full list of possible values.

Alarm Type
This field keeps Alarm type assigned to this tag. Please see **8.3 Alarm Types** for full list of possible values. If this field set to value WRTU_ALARM_TYPE_NONE this means that tag value will not be tested for alarm conditions.

Alarm Timeout
This field keeps timeout in seconds. This timeout applied when test tag value on alarm condition.
If it is greater than 0 that means that alarm condition should be detected during this timeout before system decide generate this alarm.
For example Alarm Type is WRTU_ALARM_TYPE_VALUECHANGE. This means that alarm should be generated when tag value changed to new value.
If Timeout set to 0, then alarm generated immediately as soon as value is changed.
But if Timeout set to 3 seconds for example, then system will wait 3 seconds and test tag value. If during this time tag value keep new value, then alarm generated. If during this timeout value changes back to original one, then alarm will not be generated.

Alarm Deadband
This field keeps deadband value for alarm levels. It helps to avoid multiply alarm generation if tag value jumping around alarm level (not stable signal).
This value is subtracted or added to alarm level depending on previous tag value to prevent alarm generation on small spikes in signal.
For example LOW level for alarm set to 5V and LOW and NORMAL alarms enabled.
This means that as soon as tag value will be greater than 5V NORMAL alarm will be generated.
If tag value will be less than 5V, then LOW alarm will be generated.
But if deadband will be 0 and signal will jump around 5V like 4.99 ~ 5.01, then system will generate many LOW and NORMAL alarms back to back.

To avoid this user can set Deadband to 0.3. This will means that:
- If tag value is falling and cross 5V level, then alarm will be generated only when tag value will be less than 4.97.
- If tag value is rising and cross 5V level, then alarm will be generated only when tag value will be greater than 5.03.

Alarm Level Values
If alarm type set to WRTU_ALARM_TYPE_LIMIT, then user can set 4 alarm levels: LOW LOW, LOW, HIGH, HIGH HIGH.

The main restriction is:

LOW LOW < LOW < HIGH < HIGH HIGH

All 4 levels are float values. But system will test tag value across the alarm level only if it is enabled with special **Alarm Enabled** flag.

LOW LOW – if tag value is less than this level, then system will generate LOLO alarm.
LOW – if tag value is less than this level, then system will generate LO alarm.
HIGH – if tag value is greater than this level, then system will generate HI alarm.
HIGH HIGH – if tag value is greater than this level, then system will generate HIHI alarm.

Note, if tag value is between LOW and HIGH levels, then NORMAL alarm will be generated.

Alarm Enabled Flags
These fields enable appropriate Alarm Limit value to be tested for alarm condition.
So system will check appropriate limit value with tag value only when the limit value is enabled with this flag. Otherwise system ignores the limit value.

Scaling Enable Flag
Analog Input signal can be scaled to get scaled value. To enable this configuration client should set this flag and also write correct values to scaling fields (see following 4 fields below). If this flag is enabled, then the system will scale RAW value and store result in CALCULATED value.

The formula of scaling is following:

Vcalc = (FullOut - ZeroOut) * (Value - ZeroCnt) / (FullCnt - ZeroCnt) + ZeroOut

where

Vcalc – CALCULATED value
FullOut – Full Level Output Value
ZeroOut – Zero Level Output Value
FullCnt – Full Level Scale Raw Value
ZeroCnt – Zero Level Scale Raw Value

Zero Level Scale Raw Counts
Float value used in calculation of scaled tag value. This means minimal raw output from ADC.

Full Level Scale Raw Counts
Float value used in calculation of scaled tag value. This means maximum raw output from ADC.

Zero Level Output Value
Float value used in calculation of scaled tag value. This means minimal value mapped to zero raw value on ADC.

Full Level Output Value
Float value used in calculation of scaled tag value. This means maximum value mapped to maximum raw value on ADC.

Show on LCD Flag
This flag controls appearing tag value on LCD. If it is set to 0, then this tag will not be shown on LCD. If this flag set to 1, then this tag value will be shown on LCD.

Appendix H: Calculating CRC8

```
unsigned char CRC8(unsigned char *pData, unsigned int length)
{
    unsigned char crc = 0;

    int i;
    for (i=0; i < length; i++)
        crc += pData[i];

    crc = (unsigned char)(0x5A - crc);

    return crc;
}
```