RUT850 GPS Protocols

Contents

RUT850 GPS DATA PROTOCOL

AVL data array

Because the smallest information amount that can be written is one bit, there can be some bits left unused when result is byte array. Any unused bits should be left blank.

Codec ID	Number of Data	Data	Number of Data
1 Byte	*1 Byte*	*...*	*1 byte*

Number of data – number of encoded data (number of records). In RUT850 codec ID is 08

Data

AVL Data ... AVL Data

AVL data – encoded data element.

AVL Data

Timestamp	Priority	GPS Element	IO Element
8 Bytes	*1 Byte*	*15 Bytes*	*...*

Timestamp – difference, in milliseconds, between the current time and midnight, January 1, 1970 UTC.

Priority

0	low
1	High
2	Panic
3	Security

GPS Element

Longitude	Latitude	Altitude	Angle	Satellites	Speed
4 Bytes	*4 Bytes*	*2 Bytes*	*2 Bytes*	*1 Byte*	*2 Bytes*

```
X             Longitude
Y             Latitude
Altitude      In meters above sea level
Angle         In degrees, 0 is north, increasing clock-wise
Satellites    Number of visible satellites
Speed         In km/h. 0x0000 if GPS data is invalid
```

Longitude and latitude are integer values built from degrees, minutes, seconds and milliseconds by formula:

```
d             Degrees
m             Minutes
s             Seconds
ms            Milliseconds
p             Precision (10000000)
```

If longitude is in west or latitude in south, multiply result by –1. To determine if the coordinate is negative, convert it to binary format and check the very first bit. If it is 0, coordinate is positive, if it is 1, coordinate is negative. Example:

Received value: 20 9c ca 80

Converted to BIN: 00100000 10011100 11001010 10000000 first bit is 0, which means coordinate is positive

Convered to DEC: 547146368

For more information see two‘s compliment arithmetics.

I/O Element

I/O elements (sent to server only if enabled)			
Property ID	Property Name	Bytes	Description
21	GSM level	1	GSM signal level value in scale 1-5
239	Voltage state	1	Sleep mode voltage state 1 – voltage is above the limit; 0 - voltage is lower
240	Ignition state	1	Sleep mode ignition state 1 – ignition is on; 0 – ignition is off

Example

```
Received data:
000000430802000001 5FA570D4A0000E48B4FF20B8D8BF005200BD070000000303F001EF0
11503000000000001 5FA570DC70000E48B4FE20B8D8C0005200BD070000000303F001EF01
150300000000000020000 65D5
00000043 - AVL data length
08 – Codec ID
           02 - Number of Data (4 records)
                      1st record data
                          0000015FA570D4A – Timestamp in milliseconds
(1476020749000 → 1476020749,000 in Unix Timestamp = Sunday, October 9, 2016
1:45:49 PM)
                          00 – Priority

                          GPS Element
                          0E48B4FF – Longitude 23.9645951 = 23.9645951 º N
                          20B53DC3 – Latitude 54.8985023 = 54.8985023 º E
                          0052 – Altitude 82 meters
                          00BD – Angle 189º
                          07 – 7 Visible satellites
                          0000 – 0 km/h speed

                          I/O Element
                          00 – IO element ID of Event generated (in this
case when 00 – data generated not on event)
                          03 – 3 IO elements in record
                          03 – 3 IO elements, which length is 1 Byte
                          F0 – IO element ID = 240
                          01 – 240th IO element's value = 1
                          EF – IO element ID = 240
                          01 – 239th IO element's value = 1
                          15 – IO element ID = 21
                          03 – 21st IO element's value = 3
                          00 – 0 IO elements, which value length is 2 Bytes
                          00 – 0 IO elements, which value length is 4 Bytes
                          00 – 0 IO elements, which value length is 8 Bytes
                          2'nd record data
```

```
0000015FA570DC70000E48B4FE20B8D8C0005200BD070000000000000000
```

Footer

```
020000272A
02 – Number of Data (4 records); 0000272A – CRC value of data
```

Sending data over TCP/IP

AVL data packet

AVL packet is used to encapsulate AVL data and send it to server.

Four zeros	Data length	Data	CRC

```
Four zeros        Four zero bytes (0x00)
Data length       Number of bytes in data field (Integer)
Data              Any AVL data array
CRC               16bit CRC value of data (Integer). Polynomial 0xA001.
```

Communication with server

First when module connects to server, module sends its IMEI. IMEI is sent the same way as encoding barcode. First comes short identifying number of bytes written and then goes IMEI as text (bytes).

For example IMEI 123456789012345 would be sent as 000F313233343536373839303132333435.

After receiving IMEI, server should determine if it would accept data from this module. If yes server will reply to module 01 if not 00. Note that confirmation should be sent as binary packet.

Then module starts to send first AVL data packet. After server receives packet and parses it, server must report to module number of data received as integer (four bytes).

If sent data number and reported by server doesn't match module resends sent data.

Example:

Module connects to server and sends IMEI:

000F313233343536373839303132333435

Server accepts the module:

01

Module sends data packet:

AVL data packet header	AVL data array	CRC

Four zero bytes, ‘AVL data array’ length – 254	CodecId – 08, NumberOfData – 2. (Encoded using continuous bit stream. Last byte padded to align to byte boundary)	CRC of ‘AVL data array’
00000000000000FE	0802...(data elements)...02	00008612

Server acknowledges data reception (2 data elements): 00000002

SENDING DATA OVER UDP/IP

UDP channel protocol

UDP channel is a transport layer protocol above UDP/IP to add reliability to plain UDP/IP using acknowledgment packets. The packet structure is as follows:

UDP datagram

UDP channel packet x N	Packet length	2 bytes	Packet length (excluding this field) in big endian byte order
	Packet Id	2 bytes	Packet id unique for this channel
	Packet type	1 byte	Type of this packet
	Packet payload	m bytes	Data payload

Packet Type

1	Data packet requiring acknowledgment

Acknowledgment packet should have the same packet id as acknowledged data packet and empty data payload. Acknowledgement should be sent in binary format.

Acknowledgement packet

Packet length	2 bytes	0x0003
Packet id	2 bytes	The same as in acknowledged packet
Packet type	1 byte	0x01

Sending AVL data using UDP channel

AVL data is sent encapsulated in UDP channel packets (Data payload field).

AVL data encapsulated in UDP channel packet

AVL packet id (1 byte)	Module IMEI	AVL data array

```
AVL packet id (1 byte)    id identifying this AVL packet
Module IMEI               IMEI of a sending module encoded the same as with
TCP
AVL data array            array of encoded AVL data
```

Server response to AVL data packet

AVL packet id (1 byte)	Number of accepted AVL elements (1 byte)

AVL packet id (1 byte) – id of received AVL data packet

Number of AVL data elements accepted (1 byte) – number of AVL data array entries from the beginning of array, which were accepted by the server.

Scenario:

Module sends UDP channel packet with encapsulated AVL data packet (Packet type=1).

Server sends UDP channel packet with encapsulated response (Packet type=1).

Module validates AVL packet id and Number of accepted AVL elements. If server response with valid AVL packet id is not received within configured timeout, module can retry sending.

Example:

Module sends the data:

UDP channel header	**AVL packet header**	**AVL data array**
Len – 253, Id – 0xCAFE, Packet type – 01	AVL packet id – 0xDD, IMEI – 1234567890123456	CodecId – 08, NumberOfData – 2. (Encoded using continuous bit stream)
00FDCAFE01	DD000F3133343536373839303132333435	0802...(data elements)...02

Server must respond with acknowledgment:

UDP channel header	**AVL packet acknowledgement**
Len – 5, Id – 0xCAFE, Packet type – 01	AVL packet id – 0xDD, NumberOfAcceptedData – 2
0005CAFE01	DD02