Irish Robotics
VT25g Manual

Version 1.02

Contents

General Device Operation

The device has two modes of operation: ON and STANDBY.
When the device senses the IGNITION input is active, it turns on, otherwise it stays in standby.

Under normal circumstances, when the device is on it will upload position information to the server approximately every sixty (60) seconds, with each upload consisting of 4 position reports.
If the device is unable to connect to the server it will buffer a limited number of messages and upload all available data when it is again possible to connect to the server.

Indicator LEDs

The device has three LEDs (Red, Yellow and Green) used to indicate its current status.

Red LED

This is used to indicate both power and ignition status.

LED State	*Meaning*
Off	No power and No ignition.
Flashing	Power but no ignition.
ON	Power and ignition.

Currently there is a twenty (20) second delay in detecting a change of ignition status.

Yellow LED

This is used to indicate navigation status when the unit is in the ON state.

LED State	*Meaning*
Off	No data being received from GPS module.
Flashing	Searching for satellites.
ON	Position data valid.

Green LED

This is used to indicate the status of the modem

LED State	*Meaning*
Off	The modem is off.
600 ms on / 600ms off	No SIM card inserted – Error condition. No PIN entered – Error condition. Network search or login in progress – Normal on start up.
75 ms on / 3 s off	IDLE mode.
75 ms on / 75 ms off / 75 ms on / 3 s off	One or more GPRS PDP contexts activated.
500 ms on / 25 ms off	Packet switched data transfer is in progress – Error condition.
Permanently on	Voice or data call – Error condition.

Data format

Data is posted to the server has the following mime-type: application/octet-stream.
Posted data has the following format:

```
<SYNC_1><SYNC_2><VERSION><LEN_HI><LEN_LO><SERIAL_NUMBER><One or more position records><CHECKSUM>
```

Field name	*Size*	*Value*
SYNC_1	1	0x02
SYNC_2	1	0xFD
VERSION	1	0x02*
LEN_HI	1	See below
LEN_LO	1	See below
SERIAL_NUMBER	4	See below
CHECKSUM	1	See below

Version

The format, and hence the length, of position messages is likely to change over time, e.g. extra fields, such as altitude and temperature, might be included. The version number indicates the format of position messages and therefore their length. Code parsing position messages should NOT make assumptions about either the version or the length of position messages. At time of writing the position message version is: 0x02.

Length

Taken together the <LEN_HI> and <LEN_LO> fields indicate the amount of data uploaded from <SYNC_1> to <CHECKSUM> INCLUSIVE.

Examples
LEN_HI = 0x12
LEN_LO = 0x03
Total data uploaded = 0x1203 bytes = 4611 in decimal.
Serial number
This is a four byte field that should be treated as an unsigned integer. It is transferred MOST SIGNIFICANT BYTE FIRST.
Example.
If the serial number bytes where to contain: 0x12, 0x34, 0x56, 0x78, then the serial number would be 0x12345678 or 305419896 decimal.

Checksum

XOR of all bytes preceding the checksum field.

Position messages Version 2

A version 2 position report consists of 21 octets ordered as follows:

```
<latitude><longitude><altitude><velocity><year><rest_of_timestamp><flags><temperature>
```

Fields are formatted as shown below.

Field name	*Size*	*Description*
latitude	4	32-bit IEEE floating point, big endian.
longitude	4	32-bit IEEE floating point, big endian.
altitude	4	32-bit IEEE floating point, big endian.
velocity	2	Speed and direction encoded in to 16-bits.
year	1	Fix year – 2000.
rest of time stamp	4	Remainder of fix time stamp, encoded into 32 bits.
flags	1	Device status flags.
temperature	1	Signed char.

Velocity encoding:

Field name	*Bits*	*Description*
speed	15 : 9	7-bit integer giving speed in meters per second in the range 0 to 128.
direction	8 : 0	9-bit integer giving the bearing in the range 0 to 359 degrees.

"**rest of time stamp**" encoding:

Field name	*Bits*	*Description*
month	31 : 28	Fix month as a 4-bit integer .
day	27 : 23	Fix day as a 5-bit integer.
hour	22 : 18	Fix hour as a 5-bit integer.
minute	17 : 12	Fix minute as a 6-bit integer.
second	11 : 6	Fix second as a 6-bit integer.
pad	5 : 0	Unused, set to 0.

Flags:

Field name	*Bits*	*Description*
Not used	7 : 4	N/A
Door status	3	1 = door open, 0 = door closed
Ignition status	2	1 = ignition on, 0 = ignition off
Nav status	0:1	0 = no nav data 1 = position not valid 3 = position valid

Temperature:

Value	*Meaning*
-10 to +85	Valid temperature reading.
+127	Out of range high.
-128	Out of range low.
-127	Conversion error.

Code samples

Parsing uploaded data in Java

NOTE: code assumes version 2 formatted data.

```
public Vector parsePosMsg(byte[] b)
    {
    Vector v = new Vector();
    int c,len,sn = 0,num_positions;
    byte cs = 0;

    //Checksum the message.
    for(c = 0;c < b.length - 1;c++)
       cs ^= (int)b[c] & 0xFF;
    if(cs != b[b.length -1])
       return v;

    //Extract the msg length and set the number of position reports it contains.
    len = (int)b[MSGConstants.LEN_HI_OFFSET] & 0xFF;
    len <<= 8;
    len |= (int)b[MSGConstants.LEN_LO_OFFSET] & 0xFF;

    //Calc the number of positions the message contains.
    num_positions = (len - MSGConstants.MSG_OVERHEAD) / GPSConstants.GPS_POS_REP_LEN;
    if(((len - MSGConstants.MSG_OVERHEAD) % GPSConstants.GPS_POS_REP_LEN) != 0)
       {
       System.out.println("message length is wrong");
       return v;
       }

    //Extract the serial number.
    for(c = 0,sn = 0;c < 4;c++)
       {
       sn <<= 8;
       sn |= (int)b[MSGConstants.SER_NUM_START + c] & 0xFF;
       }

    //Populate the vector with position records.
    for(c = 0;c < num_positions;c++)
       {
       PositionRecord pr = PositionRecord.parse(sn,b,MSGConstants.DATA_START_OFFSET + (c *
GPSConstants.GPS_POS_REP_LEN));
       v.add(pr);
       }
    return v;
    }
```

Parsing a version 2 position message in Java

```
public PositionRecord parse(int ser_num,byte[] b,int idx)
    {
    Calendar cal = Calendar.getInstance();
    int c,tmp = 0;
    PositionRecord pr = new PositionRecord();

    pr.setSerNum(ser_num);

    //Extract lat
    for(c = 0,tmp = 0;c < 4;c++)
       tmp |= ((int)b[idx++] & 0xFF) << (8 * c);
    pr.setLat(Float.intBitsToFloat(tmp));

    //Extract lng
    for(c = 0,tmp = 0;c < 4;c++)
       tmp |= ((int)b[idx++] & 0xFF) << (8 * c);
    pr.setLng(Float.intBitsToFloat(tmp));

    //Extract altitude
    for(c = 0,tmp = 0;c < 4;c++)
       tmp |= ((int)b[idx++] & 0xFF) << (8 * c);
    pr.setAlt(Float.intBitsToFloat(tmp));

    //Extract velocity
    tmp = (int)b[idx++] & 0xFF;
    tmp <<= 8;
    tmp |= (int)b[idx++] & 0xFF;
    pr.setSpeed((tmp & GPSConstants.GSP_PKD_VEL_V_MSK) >> GPSConstants.GPS_PKD_VEL_SHIFT);

    //Extract the direction
    pr.setDir(tmp & GPSConstants.GPS_PKD_VEL_D_MSK);

    /*Extract the position timestamp*/
    int year = 2000 + ((int)b[idx++] & 0xFF);
    for(c = 0,tmp = 0;c < 4;c++,idx++)
       {
       tmp <<= 8;
       tmp |= (int)b[idx] & 0xFF;
       }

    tmp &= 0xFFFFFFC0;
    tmp >>= 6;

    int sec = tmp & 0x3f;
    tmp >>= 6;
    int min = tmp & 0x3F;
    tmp >>= 6;
    int hrs = tmp & 0x1F;
    tmp >>= 5;
    int day = tmp & 0x1F;
    tmp >>= 5;
    int month = tmp & 0x0F;

    cal.set(year,month - 1,day,hrs,min,sec);
    pr.setTimestamp(cal.getTimeInMillis());

    /*Get the ignition status and temperature – ignore Door and Nav bit for the moment*/
    pr.setIgn((((b[idx++] & MSGConstants.IGN_FLG_MSK) == 0) ? 0 : 1));
    pr.setTemp((int)b[idx]);

    return pr;
    }
```

Contact Details

Irish Robotics
19 Cluain Ri
Ashbourne
Co. Meath
Ireland.

T: +353 (0) 1 8499476

M: +353872144506

M: +353862344779

E: support@irishrobotics.ie

W: www.irishrobotics.ie