Emlid Reach Binary (ERB) Protocol

1. Overview

The Emlid Reach Binary (ERB) protocol is an efficient binary protocol for communication between Reach and its consumers.

The protocol is used by Reach RTK for transmission various GPS messages such as navigation solution information, receiver status, speed values, precision and the detailed information about satellites. Some auxiliary messages like the version of protocol are also transferred.

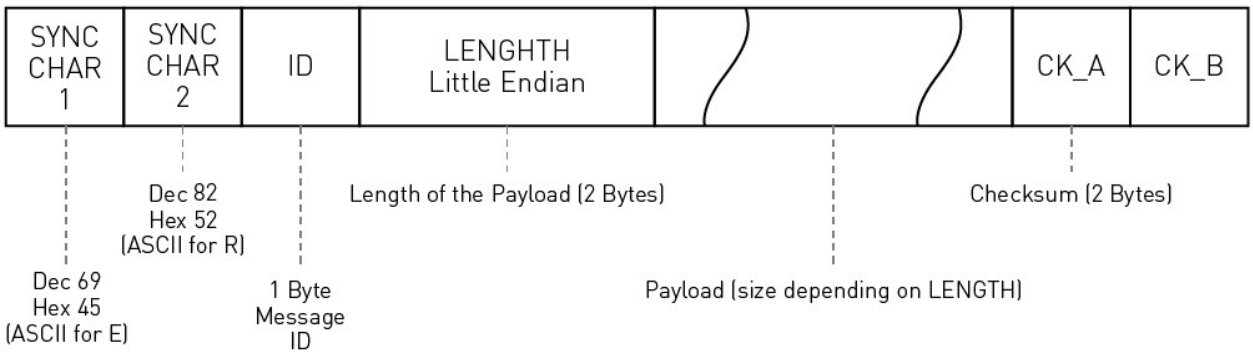
Every message has an identifier (message ID) and a checksum.

2. ERB packet structure

Features of structure:

- Messages start with 2 Sync Chars: 0x45 (‘E’) and 0x52 (‘R’).
- A 1 byte ID field defines the message that is to follow.
- After goes a 2 byte LENGTH. Value of the length is defined as the length of the payload only and does not contain Sync Chars, Length Field, ID or CRC fields. This field is in the little endian format.
- The payload is a field, which contains the data message itself. It has variable length, according to LENGTH.
- CK_A and CK_B is a 16 Bit checksum for error checking.

Packet structure of ERB:



3. ERB payload definition rules

Each message contains GPS time of week of the navigation epoch.

Place of value in packet structure depends on offset, which is defined as the sum of the size of the previous values. For example, 2 byte value starts on offset, which are a multiple of 2.

4. ERB checksum

Calculation of the checksum starts from the ID field, also including field of length of payload and payload.

The checksum algorithm used is the 8-bit Fletcher algorithm. For using this algorithm required:

- Massive buffer[N], which contains the data for checksum calculation.
- The two cka and ckb values (8-bit unsigned chars).

Cycle start from index = 2, because values of 2 Sync Chars not taken into account.

An example of 8-bit Fletcher algorithm in C code:

```
void Fletcher8(char *buffer, int N, unsigned char *cka,  unsigned char *ckb)
{
   int i;
   *cka = 0;
   *ckb = 0;
   for (i=2; i<N; i++) {
      *cka += buffer[i];
      *ckb += *cka;
   }
}
```

After the loop, the two cka and ckb values contain the checksum, transmitted at the end of the packet.

5. ERB messages overview

Name	ID	Length	Description
VER	0x01	7	Version of protocol
POS	0x02	44	Geodetic position solution
STAT	0x03	9	Receiver navigation status
DOPS	0x04	12	Dilution of precision
VEL	0x05	28	Velocity solution in NED
SVI	0x06	5+20*nSV	Space vehicle information
RTK	0x07	23	RTK information

5.1. ERB-VER (0x01)
Version of Protocol
Supported on version: 0.1.0

This message contains version of the ERB protocol. It comprises 3 numbers: high level of version, medium level of version and low level of version. Full version of protocol looks as follows:

[High level].[Medium level].[Low level]

For example 1.2.3, where:

1 – high level of version (verH)

2 – medium level of version (verM)

3 – low level of version (verL)
Message structure:

Header	ID	Length, bytes	Payload	Checksum
0x45 0x52	0x01	7	See table below	CK_A CK_B

Payload contents:

Offset, bytes	Number format	Size, bytes	Name	Scaling	Unit	Description
0	unsigned int	4	timeGPS	-	ms	GPS time of week of the navigation epoch
4	unsigned int	1	verH	-	-	High level of version
5	unsigned int	1	verM	-	-	Medium level of version
6	unsigned int	1	verL	-	-	Low level of version

5.2. ERB-POS (0x02)
Geodetic Position Solution
Supported on version: 0.1.0

This message outputs the geodetic coordinates: longitude, latitude, altitude and information about accuracy estimate.

Message structure:

Header	ID	Length, bytes	Payload	Checksum
0x45 0x52	0x02	44	See table below	CK_A CK_B

Payload contents:

Offset, bytes	Number format	Size, bytes	Name	Scaling	Unit	Description
0	unsigned int	4	timeGPS	-	ms	GPS time of week of the navigation epoch
4	double	8	lng	-	deg	Longitude
12	double	8	lat	-	deg	Latitude
20	double	8	altEl	-	m	Height above ellipsoid
28	double	8	altMsl	-	m	Height above mean sea level
36	unsigned int	4	accHor	-	mm	Horizontal accuracy estimate
40	unsigned int	4	accVer	-	mm	Vertical accuracy estimate

5.3. ERB-STAT (0x03)
Receiver Navigation Status
Supported on version: 0.1.0

This message contains status of Fix, its type and also the number of used satellites.

Message structure:

Header	ID	Length, bytes	Payload	Checksum
0x45 0x52	0x03	9	See table below	CK_A CK_B

Payload contents:

Offset, bytes	Number format	Size, bytes	Name	Scaling	Unit	Description
0	unsigned int	4	timeGPS	-	ms	GPS time of week of the navigation epoch
4	unsigned int	2	weekGPS	-	weeks	GPS week number of the navigation epoch
6	unsigned int	1	fixType	-	-	GPSfix type: 0x00 – no Fix 0x01 – Single 0x02 – Float 0x03 – RTK Fix
7	unsigned int	1	fixStatus	-	-	Navigation Fix Status. If position and velocity are valid, it takes value 0x01, else it takes 0x00
8	unsigned int	1	numSV	-	-	Number of used SVs

5.4. ERB-DOPS (0x04)
Dilution of Precision
Supported on version: 0.1.0

This message outputs dimensionless values of DOP. These values are scaled by factor 100.

For example, if received value is 123, then real is 1.23.

Message structure:

Header	ID	Length, bytes	Payload	Checksum
0x45 0x52	0x04	12	See table below	CK_A CK_B

Payload contents:

Offset, bytes	Number format	Size, bytes	Name	Scaling	Unit	Description
0	unsigned int	4	timeGPS	-	ms	GPS time of week of the navigation epoch
4	unsigned int	2	dopGeo	0.01	-	Geometric DOP
6	unsigned int	2	dopPos	0.01	-	Position DOP
8	unsigned int	2	dopVer	0.01	-	Vertical DOP
10	unsigned int	2	dopHor	0.01	-	Horizontal DOP

5.5. ERB-VEL (0x05)
Velocity Solution in NED
Supported on version: 0.1.0

This message contains the velocity in NED (North East Down) coordinates.

Message structure:

Header	ID	Length, bytes	Payload	Checksum
0x45 0x52	0x05	28	See table below	CK_A CK_B

Payload contents:

Offset, bytes	Number format	Size, bytes	Name	Scaling	Unit	Description
0	unsigned int	4	timeGPS	-	ms	GPS time of week of the navigation epoch
4	signed int	4	velN	-	cm/s	North velocity component
8	signed int	4	velE	-	cm/s	East velocity component
12	signed int	4	velD	-	cm/s	Down velocity component
16	unsigned int	4	speed	-	cm/s	Ground speed (2-D)
20	signed int	4	heading	1e-5	deg	Heading of motion 2-D
24	unsigned int	4	accS	-	cm/s	Speed accuracy Estimate

5.6. ERB-SVI (0x06)
Space Vehicle Information
Supported on version: 0.1.0

This message output information about observation satellites.

Message structure:

Header	ID	Length, bytes	Payload	Checksum
0x45 0x52	0x06	5+20*nSV	See table below	CK_A CK_B

Payload contents:

Offset, bytes	Number format	Size, bytes	Name	Scaling	Unit	Description
0	unsigned int	4	timeGPS	-	ms	GPS time of week of the navigation epoch
4	unsigned int	1	nSV	-	-	Number of visible SVs
5 + 20*N	unsigned int	1	idSV	-	-	ID of SV
6 + 20*N	unsigned int	1	typeSV	-	-	GNSS identifier: 0 — GPS 1 — GLONASS 2 — Galileo 3 — QZSS 4 — BeiDou 5 — LEO 6 — SBAS
7 + 20*N	signed int	4	carPh	1e-2	cycle	Carrier-phase

11 + 20*N	signed int	4	psRan	-	m	Pseudo range residual
15 + 20*N	signed int	4	freqD	1e-3	Hz	Doppler frequency
19 + 20*N	unsigned int	2	snr	0.25	dBHz	Signal strength
21 + 20*N	unsigned int	2	azim	1e-1	deg	Azimuth in degrees
23 + 20*N	unsigned int	2	elev	1e-1	deg	Elevation in degrees

5.7. ERB-RTK (0x07)
RTK Information
Supported on version: 0.1.0

This message output information about RTK.

Message structure:

Header	ID	Length, bytes	Payload	Checksum
0x45 0x52	0x07	23	See table below	CK_A CK_B

Payload contents:

Offset, bytes	Number format	Size, bytes	Name	Scaling	Unit	Description
0	unsigned int	1	numSV	-	-	Number of satellites used for RTK calculation
1	unsigned int	2	age	1e-2	s	Age of differential (0 when no corrections, 0xFFFF indicates overflow)
3	signed int	4	baselineN	-	mm	Distance between base and rover along the north axis
7	signed int	4	baselineE	-	mm	Distance between base and rover along the east axis
11	signed int	4	baselineD	-	mm	Distance between base and rover along the down axis
15	unsigned int	2	arRatio	1e-1	-	AR ratio
17	unsigned int	2	weekGPS	-	weeks	GPS Week Number of last baseline
19	unsigned int	4	timeGPS	-	ms	GPS Time of Week of last baseline