# NovatleOEM4 GPS Library Documentation

***Release 0.4***

**Bruno Tibério**

**May 07, 2019**

# Contents

**Date** 07 May 2019

**Version** 0.4.2

**Author** Bruno Tibério

**Contact** bruno.tiberio@tecnico.ulisboa.pt

CHAPTER 1

# Changelog

**version 0.4.2** Changed all Indice variables to Index to be in english

**version 0.4.1** Corrected bugs in reset, saveconfig and sbascontrol. Corrected messageID access using () instead [] as it is a dictionary. Added static method decoration in CRC32Value, getDebugMessage and createHeader

**version 0.4** Moved from optoparse to argparse module. Changed Queue to make it compatible with python3 queue. Backwards compatibility is maintained. Restructured default location. Moved from Lib folder to base path. Moved examples to proper folder. This cause backwards compatibility problems. On import, replace `import Lib.NovatelOEM4` with simply `import NovatelOEM4`

**version 0.3** logging configuration as moved outside module to enable user to use already configured logging handler. Check multimodule logging docs

**version 0.2** data from bestxyz message is now placed into a Queue.Queue() FIFO

**version 0.1** initial release

This module contains a few functions to interact with Novatel OEM4 GPS devices. Currently only the most important functions and definitions are configured, but the intention is to make it as much complete as possible.

A simple example can be run by executing the main function wich creates a Gps class object and execute the following commands on gps receiver:

- **begin:** on default port or given port by argv[1].
- **sendUnlogall**
- **setCom(baud=115200):** changes baudrate to 115200bps
- **askLog(trigger=2,period=0.1):** ask for log *bestxyz* with trigger *ONTIME* and period *0.1*
- wait for 10 seconds
- **shutdown:** safely disconnects from gps receiver

**Example:**

```
$python NovatelOEM4.py
```

Contents:

## 1.1 Gps Class

**class** NovatelOEM4.**Gps**(*sensorName='GPS'*)
Novatel OEM4 GPS library class

This class contents is an approach to create a library for Novatel OEM 4 GPS

**Parameters** **sensorName** (*optional*) – A sensor name if used with multiple devices.

**header_keys**
all field keys for the headers of messages.

**MessageID**
A dictionary for the types of messages sent. Not all are implemented yet!

### 1.1.1 Methods

#### 1.1.1.1 askLog

Gps.**askLog**(*logID='BESTXYZ'*, *port=192*, *trigger=4*, *period=0*, *offset=0*, *hold=0*)
Request a log from receiver.

**Parameters**

- **logID** – log type to request.
- **port** – port to report log.
- **trigger** – trigger identifier.
- **period** – the period of log.
- **offset** – offset in seconds after period.
- **hold** – mark log with hold flag or not.

**Returns** True or false if command was sucessfull or not.

The log request command is defined as:

| Field | ID | N Bytes | Description |
|---|---|---|---|
| 1 | Com header | H = 28 | Header of message |
| 2 | port | ENUM = 4 | identification of port |
| 3 | message | Ushort = 2 | Message ID of log to output |
| 4 | messageType | char = 1 | Message type (Binary) |
| 5 | RESERVED | char = 1 | |
| 6 | trigger | ENUM = 4 | message trigger |
| 7 | period | double = 8 | Log period (for ONTIME in secs) |
| 8 | offset | double = 8 | Offset for period (ONTIME in secs |
| 9 | hold | ENUM = 4 | Hold log |
| 10 | crc32 | Ulong = 4 | crc32 value |

**Note:** Total byte size = header + 32 = 60 bytes

Log trigger Identifiers (field 6):

| Binary | ASCII | Description |
|---|---|---|
| 0 | ONNEW | when the message is updated (not necessarily changed) |
| 1 | ONCHANGED | Current message and then continue to output when the message is changed |
| 2 | ONTIME | Output on a time interval |
| 3 | ONNEXT | Output only the next message |
| 4 | ONCE | Output only the current message |
| 5 | ONMARK | Output when a pulse is detected on the mark 1 input |

#### 1.1.1.2 begin

Gps.**begin**(*dataQueue*, *comPort='/dev/ttyUSB0'*, *baudRate=9600*)

Initializes the gps receiver.

This function resets the current port to factory default and setup the gps receiver to be able to acept new commands. If connection to gps is made, it launchs a thread used to parse messages comming from gps.

**Parameters**

- **comPort** – system port where receiver is connected.
- **dataQueue** – a Queue object to store incoming bestxyz messages.
- **baudRate** – baudrate to configure port. (should always be equal to factory default of receiver).

**Returns** True or False if the setup has gone as expected or not.

**Example**

```
Gps.begin(comPort="<port>",
    dataQueue=<your Queue obj>,
    baudRate=9600)
```

**Default values**

**ComPort** "/dev/ttyUSB0"

**BaudRate** 9600

> **Warning:** This class uses module `logging` wich must be configured in your main program using the `basicConfig` method. Check documentation of module logging for more info.

**HW info:**

**Receptor** Novatel Flexpak G2L-3151W.

**Antenna** Novatel Pinwheel.

#### 1.1.1.3 create_header

`Gps.`**`create_header`**(*messageID*, *messageLength*, *portAddress=192*)

Creates a header object to be passed to receiver.

**Parameters**

- **`messageID`** – the corresponding value of identifying the message body.
- **`messageLength`** – size of message in bytes excluding CRC-32bit code.
- **`portAddress`** – port from where message request is sent.

**Returns** The header of message.

The header is defined as:

| Field | Value | N Bytes | Description |
|---|---|---|---|
| 1 | sync[0] | UChar = 1 | Hexadecimal 0xAA. |
| 2 | sync[1] | UChar = 1 | Hexadecimal 0x44. |
| 3 | sync[2] | UChar = 1 | Hexadecimal 0x12. |
| 4 | header-Length | UChar = 1 | Length of the header (should always be 28 unless some firmware update) |
| 5 | messageID | UShort = 2 | This is the Message ID code |
| 6 | mes-sageType | UChar = 1 | message type mask (binary and original message) |
| 7 | portAd-dress | Uchar = 1 | Corresponding value of port |
| 8 | message-Length | UShort = 2 | Length of message body |
| 9 | sequence | UShort = 2 | This is used for multiple related logs. |
| 10 | idleTime | UChar = 1 | The time that the processor is idle in the last second between successive logs with the same Message ID |
| 11 | timeStatus | Enum = 1 | Indicates the quality of the GPS time |
| 12 | week | UShort = 2 | GPS week number. |
| 13 | ms | int = 4 | Milliseconds from the beginning of the GPS week. |
| 14 | receiver-Status | Ulong = 4 | 32 bits representing the status of various hardware and software components of the receiver. |
| 15 | reserved | UShort = 2 | |
| 16 | swVersion | UShort = 2 | receiver software build number. |

**Note:** portAddress=192 (equal to thisport)

#### 1.1.1.4 getDebugMessage

**static** Gps.**getDebugMessage**(*message*)

Create a string which contains all bytes represented as hex values

Auxiliary function for helping with debug. Receives a binary message as input and convert it as a string with the hexdecimal representation.

**Parameters** **message** – message to be represented.

**Returns** A string of corresponding hex representation of message.

#### 1.1.1.5 parseResponces

Gps.**parseResponces**()
A thread to parse responses from device

#### 1.1.1.6 reset

Gps.**reset**(*delay=0*)
Performs a hardware reset

**Parameters** **delay** – seconds to wait before resetting. Default to zero.

**Returns** A boolean if request was sucessful or not

The reset message is defined as:

| Field | value | N Bytes | Description |
|---|---|---|---|
| 1 | header | H = 28 | Header of message |
| 2 | delay | UL = 4 | Seconds to wait before reset |
| CRC32 | | UL = 4 | |

Following a RESET command, the receiver initiates a coldstart boot up. Therefore, the receiver configuration reverts either to the factory default, if no user configuration was saved, or the last SAVECONFIG settings. The optional delay field is used to set the number of seconds the receiver is to wait before resetting.

#### 1.1.1.7 saveconfig

Gps.**saveconfig**()
Save user current configuration

**Returns** A boolean if request was sucessful or not

Saveconfig message is defined as:

| Field | value | N Bytes | Description |
|---|---|---|---|
| 1 | header | H = 28 | Header of message |
| CRC32 | | UL = 4 | |

This command saves the user's present configuration in non-volatile memory. The configuration includes the current log settings, FIX settings, port configurations, and so on. Its output is in the RXCONFIG log.

#### 1.1.1.8 sbascontrol

Gps.**sbascontrol**(*keywordID=1*, *systemID=1*, *prn=0*, *testmode=0*)
Set SBAS test mode and PRN SBAS

**Parameters**

- **keywordID** – True or false. Control the reception of SBAS corrections Enable = 1, Disable = 0.
- **systemID** – SBAS system to be used.
- **prn** – PRN corrections to be used.
- **testmode** – Interpretation of type 0 messages.

**Returns** A boolean if request was sucessful or not

sbascontrol message is defined as:

| Field | value | N Bytes | Description |
|---|---|---|---|
| 1 | header | H = 28 | Header of message |
| 2 | keyword | Enum = 4 | Enable = 1 or Disable = 0 |
| 3 | system | Enum = 4 | Choose the SBAS the receiver will use |
| 4 | prn | UL = 4 | 0 - Receiver will use any PRN |
| | | | 120~138 - Receiver will use SBAS only from this PRN |
| 5 | testmode | Enum = 4 | Interpretation of type 0 messages |
| CRC32 | | UL = 4 | |

System (Field 2) is defined as:

| Binary | ASCII | Description |
|---|---|---|
| 0 | NONE | Don't use any SBAS satellites. |
| 1 | AUTO | Automatically determinate satellite system to use (default). |
| 2 | ANY | Use any and all SBAS satellites found |
| 3 | WAAS | Use only WAAS satellites |
| 4 | EGNOS | Use only EGNOS satellites |
| 5 | MSAS | Use only MSAS satellites |

Testmode (field 5) is defined as:

| Binary | ASCII | Description |
|---|---|---|
| 0 | NONE | Interpret Type 0 messages as they are intended (as do not use).(default) |
| 1 | ZEROTOTWO | Interpret Type 0 messages as type 2 messages |
| 2 | IG-NOREZERO | Ignore the usual interpretation of Type 0 messages (as do not use) and continue |

This command allows you to dictate how the receiver handles Satellite Based Augmentation System (SBAS) corrections and replaces the now obsolete WAASCORRECTION command. The receiver automatically switches to Pseudorange Differential (RTCM or RTCA) or RTK if the appropriate corrections are received, regardless of the current setting.

#### 1.1.1.9 sendUnlogall

Gps.**sendUnlogall**(*port=8*, *held=1*)

Send command unlogall to gps device.

On sucess clears all logs on all ports even held logs.

**Returns** True or False if the request has gone as expected or not.

unlogall message is defined as:

| Field | value | N Bytes | Description |
|---|---|---|---|
| 1 | header | H = 28 | Header of message |
| 2 | port | ENUM = 4 | identification of port |
| 3 | Held | ENUM = 4 | can only be 0 or 1. Clear logs with hold flag or not? |
| CRC32 | | UL = 4 | |

**Note:** See: OEMStar Firmware Reference Manual Rev 6 page 161

#### 1.1.1.10 setCom

Gps.**setCom**(*baud*, *port=6*, *parity=0*, *databits=8*, *stopbits=1*, *handshake=0*, *echo=0*, *breakCond=1*)
Set com configuration.

**Parameters**

- **baud** – communication baudrate.
- **port** – Novatel serial ports identifier (default 6 = "thisport").
- **parity** – byte parity check (default 0).
- **databits** – Number of data bits (default 8).
- **stopbits** – Number of stop bits (default 1).
- **handshake** – Handshaking (default No handshaking).
- **echo** – echo input back to user (default false)
- **breakCond** – Enable break detection (default true)

**Returns** True or false if command was sucessfull or not.

The com request command is defined as:

| Field | ID | N Bytes | Description |
|---|---|---|---|
| 1 | Com header | H = 28 | Header of message |
| 2 | port | ENUM = 4 | identification of port |
| 3 | baud | Ulong = 4 | Communication baud rate (bps) |
| 4 | parity | ENUM = 4 | Parity |
| 5 | databits | Ulong = 4 | Number of data bits (default = 8) |
| 6 | stopbits | Ulong = 4 | Number of stop bits (default = 1) |
| 7 | handshake | ENUM = 4 | Handshaking |
| 8 | echo | ENUM = 4 | No echo (default)(must be 0 or 1) |
| 9 | break | ENUM = 4 | Enable break detection (default 0) ,(must be 0 or 1) |

**Note:** Total byte size = header + 32 = 60 bytes

COM Serial Port Identifiers (field 2):

| Binary | ASCII | Description |
|---|---|---|
| 1 | COM1 | COM port 1 |
| 2 | COM2 | COM port 2 |
| 6 | THISPORT | The current COM port |
| 8 | ALL | All COM ports |
| 9 | XCOM1 | Virtual COM1 port |
| 10 | XCOM2 | Virtual COM2 port |
| 13 | USB1 | USB port 1 |
| 14 | USB2 | USB port 2 |
| 15 | USB3 | USB port 3 |
| 17 | XCOM3 | Virtual COM3 port |

Parity(field 4):

| Binary | ASCII | Description |
|---|---|---|
| 0 | N | No parity (default) |
| 1 | E | Even parity |
| 2 | O | Odd parity |

Handshaking (field 7):

| Binary | ASCII | Description |
|---|---|---|
| 0 | N | No handshaking (default) |
| 1 | XON | XON/XOFF software handshaking |
| 2 | CTS | CTS/RTS hardware handshaking |

**Note:** See: OEMStar Firmware Reference Manual Rev 6 page 56

#### 1.1.1.11 setDynamics

Gps.**setDynamics**(*dynamicID*)

Set Dynamics of receiver.

**Parameters** **dynamicID** – identifier of the type of dynamic.

**Returns** True or False if the request has gone as expected or not.

dynamics message is defined as:

| Field | value | N Bytes | Description |
|---|---|---|---|
| 1 | header | H = 28 | Header of message |
| 2 | dynamics | ENUM = 4 | identification of dynamics |
| CRC32 | | UL = 4 | |

The dynamics identifiers (field 2) are defined as:

| Binary | ASCII | Description |
|---|---|---|
| 0 | AIR | Receiver is in an aircraft or a land vehicle, for example a high speed train, with velocity greater than 110 km/h (30 m/s). This is also the most suitable dynamic for a jittery vehicle at any speed. |
| 1 | LAND | Receiver is in a stable land vehicle with velocity less than 110 km/h (30 m/s). |
| 2 | FOOT | Receiver is being carried by a person with velocity less than 11 km/h (3 m/s). |

This command adjusts the receiver dynamics to that of your environment. It is used to optimally tune receiver parameters. The DYNAMICS command adjusts the Tracking State transition time-out value of the receiver. When the receiver loses the position solution, it attempts to steer the tracking loops for fast reacquisition (5 s time-out by default). The DYNAMICS command allows you to adjust this time-out value, effectively increasing the steering time. The three states 0, 1, and 2 set the time-out to 5, 10, or 20 seconds respectively.

---

**Note:**

- The DYNAMICS command should only be used by advanced users of GPS. The default of AIR should not be changed except under very specific conditions.
- The DYNAMICS command affects satellite reacquisition. The constraint of the DYNAMICS filter with FOOT is very tight and is appropriate for a user on foot. A sudden tilted or up and down movement, for example while a tractor is moving slowly along a track, may trip the RTK filter to reset and cause the position to jump. AIR should be used in this case.

---

#### 1.1.1.12 shutdown

`Gps.`**`shutdown`**`()`

Prepare for exiting program

**Returns** always returns true after all tasks are done.

Prepare for turn off the program by executing the following tasks:

- unlogall

- reset port settings
- close port

#### 1.1.1.13 CRC32Value

**static** Gps.**CRC32Value**(*i*)

Calculate the 32bits CRC of message.

See OEMStar Firmware Reference Manual Rev 6 page 24 for more information.

**Parameters** **i** – message to calculate the crc-32.

**Returns** The CRC value calculated over the input message.

# Index