

## NMEA Library

Generated by Doxygen 1.8.8

Thu Apr 16 2015 15:34:34



# Contents

<b>1</b>	<b>LICENSE</b>	<b>1</b>
<b>2</b>	<b>NMEA</b>	<b>3</b>
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>Class Documentation</b>	<b>7</b>
4.1	NMEA Class Reference . . . . .	7
4.1.1	Constructor & Destructor Documentation . . . . .	7
4.1.1.1	NMEA . . . . .	7
4.1.1.2	NMEA . . . . .	8
4.1.2	Member Function Documentation . . . . .	8
4.1.2.1	begin . . . . .	8
4.1.2.2	enableEco . . . . .	8
4.1.2.3	enableFullPower . . . . .	8
4.1.2.4	enablePowerSave . . . . .	8
4.1.2.5	getAltitude . . . . .	8
4.1.2.6	getAltitudeUnits . . . . .	8
4.1.2.7	getBearing . . . . .	8
4.1.2.8	getDay . . . . .	8
4.1.2.9	getDow . . . . .	8
4.1.2.10	getEllipsoidHeight . . . . .	9
4.1.2.11	getEllipsoidHeightUnits . . . . .	9
4.1.2.12	getHour . . . . .	9
4.1.2.13	getLatitude . . . . .	9
4.1.2.14	getLongitude . . . . .	9
4.1.2.15	getMinute . . . . .	9
4.1.2.16	getMonth . . . . .	9
4.1.2.17	getSatellites . . . . .	9
4.1.2.18	getSecond . . . . .	9
4.1.2.19	getSpeed . . . . .	9

4.1.2.20	<a href="#">getYear</a>	9
4.1.2.21	<a href="#">isLocked</a>	10
4.1.2.22	<a href="#">isUpdated</a>	10
4.1.2.23	<a href="#">process</a>	10

# Chapter 1

## LICENSE

Copyright (c) 2014, Majenko Technologies All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Majenko Technologies nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.



## Chapter 2

# NMEA

Most GPS modules operate through TTL level RS-232 formatted serial. The messages they send usually conform to a standard called [NMEA](#), or *National Marine Electronics Association*. These messages are in a relatively easy to parse format for microcontrollers, but there are some often overlooked caveats, and the actual reception and identifying of the messages in a raw serial data stream can be tricky unless you know what you are doing.

To this end the [NMEA](#) library has been written. It takes raw serial data, parses it, identifies the incoming messages, and parses them into meaningful values.

It is capable of receiving data through any serial connection based on the Stream class, so any HardwareSerial object (Serial, Serial1, Serial2, etc), and any Stream based SoftwareSerial objects.

Values are parsed all the time the `process()` function is called and stored internally until required, and a full set of *getter* functions is provided to access all the data.





## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">NMEA</a> . . . . .	<a href="#">7</a>
--------------------------------	-------------------



## Chapter 4

# Class Documentation

### 4.1 NMEA Class Reference

#### Public Member Functions

- [NMEA](#) ()
- [NMEA](#) (Stream &dev)
- void [begin](#) ()
- void [process](#) ()
- double [getLatitude](#) ()
- double [getLongitude](#) ()
- double [getBearing](#) (bool mag=false)
- double [getSpeed](#) (bool knots=false)
- double [getAltitude](#) ()
- char [getAltitudeUnits](#) ()
- double [getEllipsoidHeight](#) ()
- char [getEllipsoidHeightUnits](#) ()
- uint8\_t [getSatellites](#) ()
- uint8\_t [getDay](#) ()
- uint8\_t [getMonth](#) ()
- uint16\_t [getYear](#) ()
- uint8\_t [getHour](#) ()
- uint8\_t [getMinute](#) ()
- uint8\_t [getSecond](#) ()
- uint8\_t [getDow](#) ()
- bool [isLocked](#) ()
- bool [isUpdated](#) ()
- void **onUpdate** (void(\*func)())

#### uBLOX NEO-6

*These functions are specifically for working with the uBLOX NEO-6 series of GPS modules, such as the one on the Sparkfun GPS shield.*

- void [enableEco](#) ()
- void [enablePowerSave](#) ()
- void [enableFullPower](#) ()

#### 4.1.1 Constructor & Destructor Documentation

##### 4.1.1.1 NMEA::NMEA ( )

This constructor defaults the [NMEA](#) parser to using the Serial object.

#### 4.1.1.2 NMEA::NMEA ( Stream & dev )

This constructor allows you to pass a specific serial object to the parser. It is your responsibility to ensure that the serial object is configured for 38400 baud (or the baud of your GPS module) and has had "begin" called on it.

### 4.1.2 Member Function Documentation

#### 4.1.2.1 void NMEA::begin ( )

Pre-configures any required variables. Calling this before any processing is done is required.

#### 4.1.2.2 void NMEA::enableEco ( )

Enable ECO power mode. This is a half-way house between full power and power saving mode. It uses more power during acquisition but saves power during idle time.

#### 4.1.2.3 void NMEA::enableFullPower ( )

Full power is what it says - it uses the maximum power all the time. Everything works much faster; satellites are found more reliably, etc.

#### 4.1.2.4 void NMEA::enablePowerSave ( )

Power save mode uses the minimum power. Everything takes longer though.

#### 4.1.2.5 double NMEA::getAltitude ( )

Returns the current height above sea level. The units are not defined, but can be obtained with the `getAltitudeUnits()` function.

#### 4.1.2.6 char NMEA::getAltitudeUnits ( )

Returns the units used for the height above sea level. Usually 'M' for meters.

#### 4.1.2.7 double NMEA::getBearing ( bool mag = false )

Returns the current calculated bearing or heading. If true is passed as a parameter it returns the bearing to magnetic north. If false is passed, or no parameter is used, it returns the bearing to true north.

#### 4.1.2.8 uint8\_t NMEA::getDay ( )

Returns the current day of the month (1 ... 31).

#### 4.1.2.9 uint8\_t NMEA::getDow ( )

Calculates the day of the week (0 ... 6, 0 being Sunday) from the current date values.

#### 4.1.2.10 double NMEA::getEllipsoidHeight ( )

Returns the height above the WGS84 Ellipsoid. The units are not defined, but can be obtained with the [getEllipsoidHeightUnits\(\)](#) function.

The WGS84 Ellipsoid is a mathematical approximation of the shape of the earth as a smooth oblate spheroid.

For more information see [http://en.wikipedia.org/wiki/World\\_Geodetic\\_System](http://en.wikipedia.org/wiki/World_Geodetic_System)

#### 4.1.2.11 char NMEA::getEllipsoidHeightUnits ( )

Returns the units used for the height above the WGS84 Ellipsoid. Usually 'M' for meters.

#### 4.1.2.12 uint8\_t NMEA::getHour ( )

Returns the current hour of the day (0 ... 23).

#### 4.1.2.13 double NMEA::getLatitude ( )

Returns the current latitude in degrees

#### 4.1.2.14 double NMEA::getLongitude ( )

Returns the current longitude in degrees

#### 4.1.2.15 uint8\_t NMEA::getMinute ( )

Returns the current minutes (0 ... 59).

#### 4.1.2.16 uint8\_t NMEA::getMonth ( )

Returns the current month number (1 ... 12).

#### 4.1.2.17 uint8\_t NMEA::getSatellites ( )

Returns the number of currently locked satellites.

#### 4.1.2.18 uint8\_t NMEA::getSecond ( )

Returns the current seconds (0 ... 59).

#### 4.1.2.19 double NMEA::getSpeed ( bool *knots* = false )

Returns the current calculated speed. If true is passed as a parameter it returns the speed in Knots. If false is passed, or no parameter is used, it returns the speed in kilometers per hour.

#### 4.1.2.20 uint16\_t NMEA::getYear ( )

Returns the current year (2000 ... 2099).

#### 4.1.2.21 `bool NMEA::isLocked ( )`

Returns true if the receiver is locked on, false otherwise.

#### 4.1.2.22 `bool NMEA::isUpdated ( )`

Returns true if the processor has received and processed a new valid message. Resets the updated flag internally.

#### 4.1.2.23 `void NMEA::process ( )`

This function is the main heart of the [NMEA](#) processor. It receives characters from the serial device, identifies the frame wrapper characters, and stores the frame data into a buffer. When the frame is complete it calls the relevant decoder function to handle the data.

This function must be called frequently to process the data.

The documentation for this class was generated from the following files:

- `NMEA.h`
- `NMEA.cpp`