NEO-M9L

Automotive dead reckoning (ADR) GNSS module

Automotive grade

Integration manual

Abstract

This document describes how to enable a successful design with the NEO-M9L module. NEO-M9L offers ultra-robust meter-level GNSS positioning performance with concurrent reception of up to four GNSS (GPS, GLONASS, BeiDou, Galileo) together with vehicle speed information and integrated 3D sensors in a 12.2 x 16.0 mm package.

www.u-blox.com

UBX-20048488 - R03
C2-Restricted

Document information

Title	**NEO-M9L**
Subtitle	Automotive dead reckoning (ADR) GNSS module
Document type	Integration manual
Document number	UBX-20048488
Revision and date	R03 12-Oct-2022
Disclosure restriction	C2-Restricted

This document applies to the following products:

Type number	Firmware version	PCN reference	RN reference
NEO-M9L-01A-00	ADR 5.10	-	UBX-20016394
NEO-M9L-20A-00	ADR 5.10	-	UBX-21028129
NEO-M9L-20B-00	ADR 5.10	-	UBX-21028142
NEO-M9L-01A-01	ADR 5.15	UBX-22037180	UBX-20016394
NEO-M9L-20A-01	ADR 5.15	UBX-22037181	UBX-21028129
NEO-M9L-20B-01	ADR 5.15	UBX-22037182	UBX-21028142

Contents

1 Integration manual overview 6
2 System description 7
2.1 Overview 7
2.1.1 Automotive dead reckoning (ADR) 7
2.1.2 Priority navigation mode 7
2.2 Architecture 8
2.2.1 Block diagram 8
3 Receiver functionality 9
3.1 Receiver configuration 9
3.1.1 Changing the receiver configuration 9
3.1.2 Default GNSS configuration 9
3.1.3 Default interface settings 9
3.1.4 Basic receiver configuration 10
3.1.5 Navigation configuration 11
3.2 Automotive dead reckoning (ADR) 13
3.2.1 Introduction 13
3.2.2 Sensor fusion dynamic platform models 13
3.2.3 Solution type 16
3.2.4 Installation configuration 16
3.2.5 Sensor configuration 22
3.2.6 ADR system configuration 25
3.2.7 Operation 26
3.2.8 Priority navigation mode 36
3.2.9 Advanced calibration handling 38
3.2.10 Wake on motion feature 39
3.2.11 Map-matching input 41
3.3 Primary and secondary output 41
3.3.1 Introduction 41
3.3.2 Configuration 42
3.3.3 Expected output behavior 43
3.3.4 Example use cases 43
3.4 SBAS 44
3.5 QZSS SLAS 45
3.5.1 Features 45
3.5.2 Configuration 46
3.6 Communication interfaces 46
3.6.1 UART 47
3.6.2 I2C interface 48
3.6.3 SPI interface 51
3.6.4 USB interface 52
3.7 Predefined PIOs 53
3.7.1 D_SEL 53
3.7.2 RESET_N 53
3.7.3 SAFEBOOT_N 53

3.7.4 TIMEPULSE 54
3.7.5 TX_READY 54
3.7.6 EXTINT 54
3.7.7 WT and DIR inputs 55
3.7.8 Wake on motion (WOM) 55
3.8 Multiple GNSS assistance (MGA) 55
3.8.1 Authorization 56
3.8.2 Preserving MGA and operational data during power-off 56
3.9 Save-on-shutdown feature 56
3.10 Clocks and time 57
3.10.1 Receiver local time 57
3.10.2 Navigation epochs 58
3.10.3 iTOW timestamps 58
3.10.4 GNSS times 59
3.10.5 Time validity 59
3.10.6 UTC representation 60
3.10.7 Leap seconds 60
3.10.8 Real-time clock 61
3.10.9 Date 61
3.10.10 Time pulse 61
3.10.11 Time mark 65
3.11 Security 66
3.11.1 Spoofing detection / monitoring 67
3.11.2 Jamming/interference detection / monitoring 67
3.11.3 GNSS receiver integrity 68
3.11.4 Receiver message authentication 68
3.12 eFuse OTP layer configuration 75
3.12.1 eFuse OTP organization in u-blox 9 75
3.12.2 eFuse OTP fixed section 75
3.12.3 eFuse OTP dynamic section 76
3.13 u-blox protocol feature descriptions 76
3.13.1 Broadcast navigation data 76
3.14 Forcing a receiver reset 80
3.15 Firmware upload 81
3.15.1 Firmware update to flash 81
3.15.2 Firmware update sample code 83
3.15.3 Firmware verification 84
3.16 Production test 84
3.16.1 Verification of PIO pins 84
3.16.2 Sensor production test 85
3.16.3 Test GNSS performance 85
4 Design 86
4.1 Pin assignment 86
4.2 Power supply 87
4.2.1 VCC: Main supply voltage 87
4.2.2 V_BCKP: Backup supply voltage 87
4.2.3 NEO-M9L power supply 88
4.3 NEO-M9L minimal design 88
4.4 WT and DIR interface example 89
4.5 Antenna 90

4.5.1 Antenna design with passive antenna....91
4.5.2 Antenna design with external LNA or active antenna....91
4.6 EOS/ESD precautions....93
4.6.1 ESD protection measures....93
4.6.2 EOS precautions....94
4.6.3 Safety precautions....94
4.7 Electromagnetic interference on I/O lines....94
4.7.1 General notes on interference issues....95
4.7.2 In-band interference mitigation....95
4.7.3 Out-of-band interference....96
4.8 Layout....96
4.8.1 Placement....96
4.8.2 Thermal management....96
4.8.3 Package footprint, copper and paste mask....97
4.8.4 Layout guidance....98
4.9 Design guidance....99
4.9.1 General considerations....99
4.9.2 Backup battery....100
4.9.3 RF front-end circuit options....100
4.9.4 Antenna/RF input....101
4.9.5 Ground pads....101
4.9.6 Schematic design....101
4.9.7 Layout design-in guideline....101
5 Product handling....102
5.1 ESD handling precautions....102
5.2 Soldering....102
5.3 Tapes....106
5.4 Reels....107
5.5 Moisture sensitivity levels....107
Appendix....108
A Glossary....108
B Providing odometer data over the software sensor interface....109
B.1 Configuration....109
B.2 Sensor data in UBX-ESF-MEAS....111
Related documents....113
Revision history....114

1 Integration manual overview

This document is an important source of information on all aspects of NEO-M9L module. The purpose of this document is to provide guidelines for a successful integration of the receiver with the customer's end product.

2 System description

2.1 Overview

The NEO-M9L GNSS receiver features the u-blox M9 standard precision GNSS platform with 3D automotive dead reckoning (ADR) . It provides exceptional sensitivity and acquisition times for all L1 GNSS systems. u-blox M9 receivers are available in different variants to serve automotive and industrial tracking applications, such as navigation, telematics and UAVs.

The u-blox M9 standard precision GNSS platform with ADR, which delivers meter-level accuracy, succeeds the well-known u-blox M8 product range.

u-blox M9 receivers support concurrent reception of four GNSS. The high number of visible satellites allows the receiver to select the best signals. This maximizes the position accuracy, in particular under challenging conditions such as deep urban canyons.

u-blox M9 receivers detect jamming and spoofing events and report them to the host, which allows the system to react to such events. Advanced filtering algorithms mitigate the impact of RF interference and jamming, thus enabling the product to operate as intended.

The receiver also provides higher navigation rate and improved security features compared to previous u-blox GNSS generations.

The intelligent combination of GNSS and sensor measurements enables accurate, real-time positioning, speed and heading information at rates up to 50 Hz. Access to native, high-rate sensor data also enables host applications to make full use of the receiver's assets.

The NEO-M9L module is available in the NEO form factor, which is a 12.2 x 16.0 mm LCC package.

2.1.1 Automotive dead reckoning (ADR)

Automotive dead reckoning (ADR) provides high-accuracy positioning in locations with poor or no GNSS coverage. ADR is based on sensor fusion dead reckoning (SFDR) technology, which combines multi-constellation GNSS measurements with the NEO-M9L's internal 6-axis IMU and wheel tick or speed.

See the ADR section in this document for more information.

2.1.2 Priority navigation mode

Priority navigation mode provides a low-latency position, velocity, and vehicle attitude solution to be output at a high rate by utilizing sensor-based propagation in between GNSS measurement updates, thus prioritizing the time-critical data.

See the Priority navigation mode section in this document for more information.

2.2 Architecture

The NEO-M9L receiver provides all the necessary RF and baseband processing to enable multi-constellation operation. The block diagram below shows the key functionality.

2.2.1 Block diagram

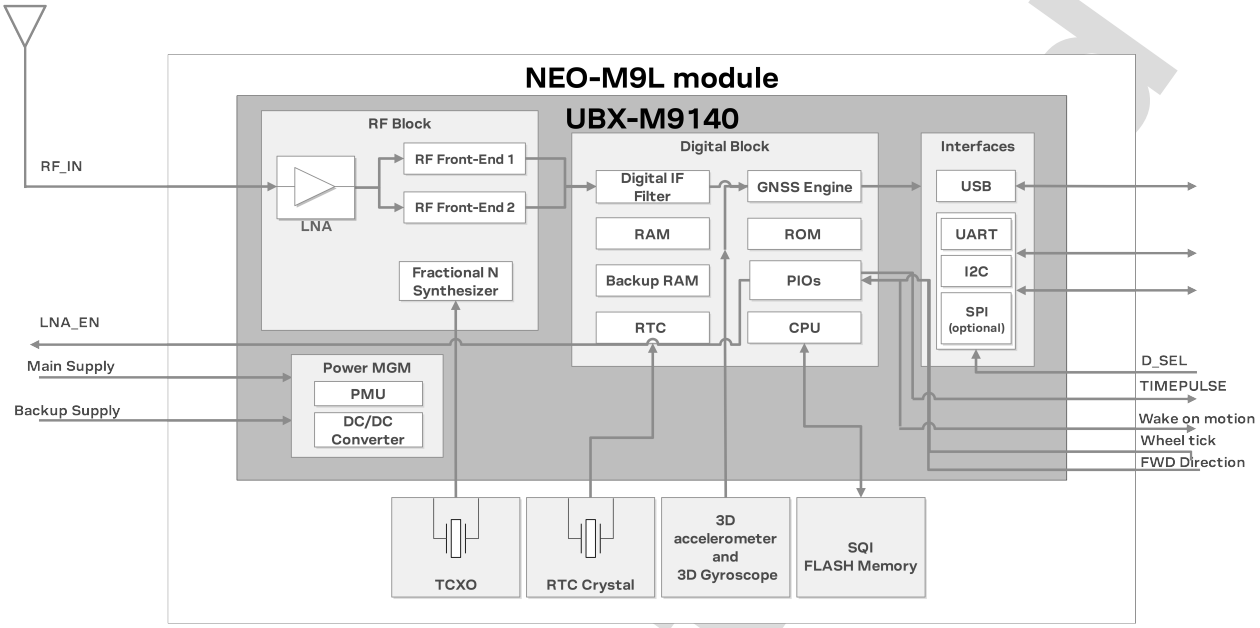


Figure 1: NEO-M9L block diagram

3 Receiver functionality

This section describes the NEO-M9L operational features and their configuration.

3.1 Receiver configuration

The NEO-M9L is fully configurable with UBX configuration interface keys. The configuration database in the receiver's RAM holds the current configuration, which is used by the receiver at run-time. It is constructed on startup of the receiver from several sources of configuration. The configuration interface and the available keys are described fully in the applicable interface description [2].

A configuration setting stored in RAM remains effective until power-down or reset. If stored in BBR (battery-backed RAM), the setting will be used as long as the backup battery supply remains. Configuration settings can be saved permanently in flash memory.

⚠ **CAUTION** The configuration interface has changed from earlier u-blox positioning receivers. Legacy messages are deprecated, and will not be supported in future firmware releases. Users are advised to adopt the configuration interface described in this document. See legacy UBX-CFG message fields reference section in the applicable interface description [2].

Configuration interface settings are held in a database consisting of separate configuration items. An item is made up of a pair consisting of a key ID and a value. Related items are grouped together and identified under a common group name: CFG-GROUP-*; a convention used in u-center and within this document. Within u-center, a configuration group is identified as "Group name" and the configuration item is identified as the "item name" under the "Generation 9 Configuration View" - "Advanced Configuration" view.

The UBX messages available to change or poll the configurations are UBX-CFG-VALSET, UBX-CFG-VALGET, and UBX-CFG-VALDEL. For more information about these messages and the configuration keys, see the configuration interface section in the applicable interface description [2].

3.1.1 Changing the receiver configuration

The configuration messages UBX-CFG-VALSET, UBX-CFG-VALGET and UBX-CFG-VALDEL will result in a UBX-ACK-ACK or a UBX-ACK-NAK response.

3.1.2 Default GNSS configuration

The NEO-M9L default GNSS configuration is set as follows:

- GPS: L1C/A
- GLONASS: L1OF
- Galileo: E1B/C
- BeiDou: B1I
- QZSS: L1C/A
- SBAS: L1C/A

For more information about the default configuration, see the applicable interface description [2].

3.1.3 Default interface settings

Interface	Settings
UART	38400 baud, 8 bits, no parity bit, 1 stop bit.
	Output messages: NMEA **GGA, GLL, GSA, GSV, RMC, VTG, TXT** (no UBX).

Interface	Settings
	Input protocols: UBX and NMEA.
USB	Output messages activated as in UART. Input protocols available as in UART.
I2C	Output messages activated as in UART. Input protocols available as in UART.
SPI	Output messages activated as in UART. Input protocols available as in UART.

Table 1: Default interface settings

☞ Refer to the applicable interface description [2] for information about further settings.

By default the NEO-M9L outputs NMEA messages that include satellite data for all GNSS bands being received. This results in a higher-than-before NMEA load output for each navigation period. Make sure the UART baud rate being used is sufficient for the selected navigation rate and the number of GNSS signals being received.

3.1.4 Basic receiver configuration

This section summarizes the basic receiver configuration most commonly used.

3.1.4.1 Communication interface configuration

Several configuration groups allow operation mode configuration of the various communication interfaces. These include parameters for the data framing, transfer rate and enabled input/output protocols. See Communication interfaces section for details. The configuration groups available for each interface are:

Interface	Configuration groups
UART1	CFG-UART1-*, CFG-UART1INPROT-*, CFG-UART1OUTPROT-*
USB	CFG-USB-*, CFG-USBINPROT-*, CFG-USBOUTPROT-*
I2C	CFG-I2C-*, CFG-I2CINPROT-*, CFG-I2COUTPROT-*
SPI	CFG-SPI-*, CFG-SPIINPROT-*, CFG-SPIOUTPROT-*

Table 2: Interface configurations

3.1.4.2 Message output configuration

This product supports two protocols for output messages. One is NMEA and the other one is a u-blox proprietary "UBX" protocol. NMEA is a well-known industry standard, used mainly for providing information about position, time and satellites. UBX messages can be used to configure the receiver and also to periodically provide information about position, time and satellites. With the UBX protocol it is easy to monitor the receiver status and get much deeper information about the receiver status. The rate of NMEA and UBX protocol output messages are configurable and it is possible to enable or disable single NMEA or UBX messages individually.

If the rate configuration value is zero, then the corresponding message will not be output. Values greater than zero indicate how often the message is output.

For periodic output messages the rate relates to the event the message is related to. For example, the UBX-NAV-PVT (navigation, position, velocity and time solution) is related to the navigation epoch. If the rate of this message is set to one (1), it will be output for every navigation epoch. If the rate is set to two (2), it will be output every other navigation epoch. The rates of the output messages are individually configurable per communication interface. See the CFG-MSGOUT-* configuration group.

Some messages, such as UBX-MON-VER, are non-periodic and will only be output as an answer to a poll request.

The UBX-INF-* information messages are non-periodic output messages that do not have a message rate configuration. Instead they can be enabled for each communication interface via the CFG-INFMSG-* configuration group.

☞ All message output is additionally subject to the protocol configuration of the communication interfaces. Messages of a given protocol will not be output until the protocol is enabled for output on the interface (see Communication interface configuration).

3.1.4.3 GNSS signal configuration

The GNSS constellations are configurable with configuration keys. Each GNSS constellation can be enabled or disabled independently.

3.1.5 Navigation configuration

This section presents various configuration options related to the navigation engine. These options can be configured through CFG-NAVSPG-* configuration keys.

3.1.5.1 Platform settings

u-blox receivers support different dynamic platform models (see the table below) to adjust the navigation engine to the expected application environment. These platform settings can be changed dynamically without performing a power cycle or reset. The settings improve the receiver's interpretation of the measurements and thus provide a more accurate position output. Setting the receiver to an unsuitable platform model for the given application environment is likely to result in a loss of receiver performance and position accuracy.

The dynamic platform model can be configured through the CFG-NAVSPG-DYNMODEL configuration item. The supported dynamic platform models and their details can be seen in Table 3 and Table 4 below.

Platform	Description
Portable	Applications with low acceleration, e.g. portable devices. Suitable for most situations.
Stationary	Used in timing applications (antenna must be stationary) or other stationary applications. Velocity restricted to 0 m/s. Zero dynamics assumed.
Pedestrian	Applications with low acceleration and speed, e.g. how a pedestrian would move. Low acceleration assumed.
Automotive (default)	Used for applications with equivalent dynamics to those of a passenger car. Low vertical acceleration assumed.
At sea	Recommended for applications at sea, with zero vertical velocity. Zero vertical velocity assumed. Sea level assumed.
Airborne <1g	Used for applications with a higher dynamic range and greater vertical acceleration than a passenger car. No 2D position fixes supported.
Airborne <2g	Recommended for typical airborne environments. No 2D position fixes supported.
Airborne <4g	Only recommended for extremely dynamic environments. No 2D position fixes supported.
Wrist	Only recommended for wrist-worn applications. Receiver will filter out arm motion.
Motorbike	Used for applications with dynamics equivalent to those of a motorbike. Low vertical acceleration assumed.
E-scooter	Used for applications with dynamics equivalent to those of an e-scooter. Low vertical acceleration assumed.

Table 3: Dynamic platform models

Platform	Max altitude [m]	Max horizontal velocity [m/s]	Max vertical velocity [m/s]	Sanity check type	Max position deviation
Portable	12000	310	50	Altitude and velocity	Medium

Platform	Max altitude [m]	Max horizontal velocity [m/s]	Max vertical velocity [m/s]	Sanity check type	Max position deviation
Stationary	9000	10	6	Altitude and velocity	Small
Pedestrian	9000	30	20	Altitude and velocity	Small
Automotive	6000	100	15	Altitude and velocity	Medium
At sea	500	25	5	Altitude and velocity	Medium
Airborne <1g	80000	100	6400	Altitude	Large
Airborne <2g	80000	250	10000	Altitude	Large
Airborne <4g	80000	500	20000	Altitude	Large
Wrist	9000	30	20	Altitude and velocity	Medium
Motorbike	6000	100	15	Altitude and velocity	Medium
E-scooter	6000	50	15	Altitude, velocity and attitude	Medium

Table 4: Dynamic platform model details

Applying dynamic platform models designed for high acceleration systems (e.g. airborne <2g) can result in a higher standard deviation in the reported position.

If a sanity check against a limit of the dynamic platform model fails, then the position solution is invalidated. Table 4 above shows the types of sanity checks which are applied for a particular dynamic platform model.

3.1.5.2 Navigation input filters

The navigation input filters in CFG-NAVSPG-* configuration group provide the input data of the navigation engine.

Configuration item	Description
CFG-NAVSPG-FIXMODE	By default, the receiver calculates a 3D position fix if possible but reverts to 2D position if necessary (**auto 2D/3D**). The receiver can be forced to only calculate 2D (**2D only**) or 3D (**3D only**) positions.
CFG-NAVSPG-CONSTR_ALT, CFG-NAVSPG-CONSTR_ALTVAR	The fixed altitude is used if fixMode is set to 2D only. A variance greater than zero must also be supplied.
CFG-NAVSPG-INFIL_MINELEV	Minimum elevation of a satellite above the horizon in order to be used in the navigation solution. Low elevation satellites may provide degraded accuracy, due to the long signal path through the atmosphere.
CFG-NAVSPG-INFIL_NCNOTHRS, CFG-NAVSPG-INFIL_CNOTHRS	A navigation solution will only be attempted if there are at least the given number of SVs with signals at least as strong as the given threshold.

Table 5: Navigation input filter parameters

If the receiver only has three satellites for calculating a position, the navigation algorithm uses a constant altitude to compensate for the missing fourth satellite. When a satellite is lost after a successful 3D fix (min four satellites available), the altitude is kept constant at the last known value. This is called a 2D fix.

☞ u-blox receivers do not calculate any navigation solution with less than three satellites.

3.1.5.3 Navigation output filters

The result of a navigation solution is initially classified by the fix type (as detailed in the `fixType` field of UBX-NAV-PVT message). This distinguishes between failures to obtain a fix at all ("No Fix") and cases where a fix has been achieved, which are further subdivided into specific types of fixes (e.g. 2D, 3D, dead reckoning).

Where a fix has been achieved, a check is made to determine whether the fix should be classified as valid or not. A fix is only valid if it passes the navigation output filters as defined in CFG-NAVSPG-OUTFIL. In particular, both PDOP and accuracy values must be below the respective limits.

☞ Important: Users are recommended to check the `gnssFixOK` flag in the UBX-NAV-PVT or the NMEA valid flag. Fixes not marked valid should not be used.

UBX-NAV-STATUS message also reports whether a fix is valid in the `gpsFixOK` flag. This message has only been retained for backwards compatibility and users are recommended to use the UBX-NAV-PVT message.

3.1.5.4 Weak signal compensation

In normal operating conditions, low signal strength (i.e. signal attenuation) indicates likely contamination by multi-path. The receiver trusts such signals less in order to preserve the quality of the position solution in poor signal environments. This feature can result in degraded performance in situations where the signals are attenuated for another reason, for example due to antenna placement. In this case, the weak signal compensation feature can be used to restore normal performance. There are three possible modes:

- Disabled: no weak signal compensation is performed
- Automatic: the receiver automatically estimates and compensates for the weak signal
- Configured: the receiver compensates for the weak signal based on a configured value

These modes can be selected using CFG-NAVSPG-SIGATTCOMP. In the case of the "configured" mode, the user should input the maximum C/N0 observed in a clear-sky environment, excluding any outliers or unusually high values. The configured value can have a large impact on the receiver performance, so it should be chosen carefully.

3.2 Automotive dead reckoning (ADR)

3.2.1 Introduction

u-blox solutions for automotive dead reckoning (ADR) allow positioning in places with poor or no GNSS coverage. ADR is based on sensor fusion dead reckoning (SFDR) technology, which combines GNSS measurements with those from external sensors. The NEO-M9L computes a solution type called GAWT (gyroscope, accelerometers, wheel tick) by combining GNSS measurements with the outputs of a 3-axis accelerometer, a 3-axis gyroscope and wheel tick (sometimes called a speed tick) or speed measurements. The utilization of these sensors ensures a quick recovery of the navigation solution after short GNSS signal outage (going under a bridge, signaling panels, and so on).

The firmware automatically detects and continuously calibrates the sensors.

3.2.2 Sensor fusion dynamic platform models

NEO-M9L supports different dynamic platform models to adjust the sensor fusion navigation engine to the expected application environment. These platform settings can be changed dynamically without performing a power cycle or reset. However, it requires a recalibration of the sensors (IMU + WT/speed measurements). The settings improve the receiver's interpretation of the measurements and thus provide a more accurate position output. Setting the receiver to an unsuitable platform model for the given application environment is likely to result in a degradation of receiver performance and loss of position accuracy.

The dynamic platform model can be configured through the `CFG-NAVSPG-DYNMODEL` configuration item.

NEO-M9L's sensor fusion algorithm is optimized for the following platforms.

- Automotive
- Motorbike
- E-scooter

3.2.2.1 Automotive dynamic model

The automotive dynamic model is used for applications with dynamics equivalent to those of a passenger car. The automotive dynamic model shall be selected by the configuration item `CFG-NAVSPG-DYNMODEL = 04`.

It combines the GNSS signal with the data from gyroscope, accelerometer and wheel ticks (or speed measurements). More information can be found in the Solution type section. The automotive dynamic model supports UDR fallback mode, with missing wheel tick data. More information can be found in the UDR fallback mode section.

The automotive dynamic model supports both the Automatic IMU-mount alignment and the User-defined IMU-mount alignment.

To obtain optimal position accuracy the automotive dynamic model supports lever arms. More information can be found in the Installation configuration section.

☞ For the automotive dynamic model, the position, velocity, heading, pitch and roll are frozen when the vehicle is static. The static state is determined once the sensors are calibrated and the wheel ticks are stationary.

3.2.2.2 Motorbike dynamic model

The motorbike dynamic model is used for applications with dynamics equivalent to those of a motorbike. A motorbike is a low-speed vehicle with one pulling non-steering rear wheel and one steering front wheel. The motorbike dynamic model shall be selected by the configuration item `CFG-NAVSPG-DYNMODEL = 10`.

The motorbike dynamic model combines the GNSS signal with the data from gyroscope, accelerometer and wheel ticks (or speed measurements). More information can be found in the Solution type section. The motorbike dynamic model supports UDR fallback mode with missing wheel tick data. More information can be found in the UDR fallback mode section.

The motorbike dynamic model only supports the User-defined IMU-mount alignment.

To obtain optimum position accuracy the motorbike dynamic model supports lever arms. More information can be found in the Installation configuration section.

The motorbike feature benefits from odometer measurements (wheel ticks or wheel speed) from either wheel. Ideally, the odometer data should come from the rear wheel if the IMU is in the deck and from the front wheel when the IMU is in the handlebar.

The receiver starts calibrating the sensors once the following conditions are met.

- Speed is over 2.5 m/s.
- Heading accuracy is under 3 deg.
- Roll rate is less than 90 deg./s.
- Pitch rate is less than 90 deg./s.
- Heading rate is less than 90 deg./s.

☞ The positioning performance may be degraded when the motorbike is driving on uneven surfaces like cobblestone roads due to the increased noise of the IMU measurements.

☞ The positioning performance is not guaranteed in special maneuvers where the motorbike is lifted and carried, flipped 180 degrees, or left lying on the ground and lifted up again.

☞ With motorbike dynamic model, the position, velocity, and attitude are frozen when the vehicle is static.

☞ Disabling the automatic wheel tick polarity detection with `CFG-SFODO-DIS_AUTODIRPINPOL = 1` may speed up the wheel tick initialization, resulting in a quicker start to the calibration.

⚠ When the IMU is installed at the handlebar, the navigation performance is degraded for two reasons: Firstly, the sensor fusion navigation filter assumes that the IMU shall be mounted to a fixed orientation with respect to the vehicle compartment where the wheel speed is measured. This assumption is broken when the IMU is installed to the handlebar while getting the wheel speed data from the rear wheel. Secondly, the IMU measurements tend to be noisier in the handlebar.

3.2.2.3 E-scooter dynamic model

The e-scooter dynamic model is used for applications with dynamics equivalent to those of an e-scooter. An e-scooter is a low-speed vehicle with one pulling non-steering rear wheel and one steering front wheel. The e-scooter dynamic model shall be selected by the configuration item `CFG-NAVSPG-DYNMODEL = 12`.

It combines the GNSS signal with the data from gyroscope, accelerometer and wheel ticks (or speed measurements), more information can be found in the Solution type section. The e-scooter dynamic model supports UDR fallback mode with missing wheel tick data. More information can be found in the UDR fallback mode section.

The e-scooter dynamic model only supports the User-defined IMU-mount alignment.

To obtain optimum position accuracy the e-scooter dynamic model supports lever arms. More information can be found in the Installation configuration section.

The e-scooter feature benefits from odometer measurements (wheel ticks or wheel speed) from either wheel. Ideally, the odometer data should come from the rear wheel if the IMU is in the deck and from the front wheel when the IMU is in the handlebar.

The receiver starts calibrating the sensors once the following conditions are met.

- Speed is over 2.5 m/s.
- Heading accuracy is under 6 deg.
- Roll rate is less than 20 deg./s.
- Pitch rate is less than 15 deg./s.
- Heading rate is less than 15 deg./s.

☞ The positioning performance may be degraded when the e-scooter is driving on uneven surfaces like cobblestone roads due to the increased noise of the IMU measurements.

☞ The positioning performance is not guaranteed in special maneuvers where the e-scooter is lifted and carried, flipped 180 degrees, or left lying on the ground and lifted up again.

☞ Unlike automotive dynamic model, position, velocity, and attitude are not frozen when the e-scooter is static.

☞ The e-scooter dynamic model is not optimized for parking garages or longer GNSS outages.

☞ Automatic wheel tick polarity detection should not be used. This is achieved by the configuration item `CFG-SFODO-DIS_AUTODIRPINPOL = 1`

⚠ When the IMU is installed at the handlebar, the navigation performance is degraded for two reasons: Firstly, the sensor fusion navigation filter assumes that the IMU shall be mounted to a fixed orientation with respect to the vehicle compartment where the wheel speed is measured. This assumption is broken when the IMU is installed to the handlebar while getting the wheel speed data from the rear wheel. Secondly, the IMU measurements tend to be noisier in the handlebar.

3.2.3 Solution type

NEO-M9L produces a solution that combines GNSS signals with data from gyroscopes, accelerometers and wheel tick sensors to compute a fused navigation solution. The solution type is called GAWT, and it is described in the following sections.

To operate the NEO-M9L in GAWT mode with optimal performance, the following tasks need to be completed:

- It is essential that the IMU-to-antenna lever arm (CFG-SFIMU-IMU2ANT keys) and the IMU-to-VRP lever arm (CFG-SFODO-IMU2VRP keys) are accurately configured (centimeter level ideally).
- The IMU misalignment angles are also essential. It is recommended to enable automatic alignment with the CFG-SFIMU-AUTO_MNTALG_ENA key, if misalignment angles are unknown. The misalignment angles can also be configured manually with CFG-SFIMU-IMU_MNTALG keys. Configuring misalignment angles manually speeds up the calibration procedure.
- It is mandatory to perform an initial calibration drive after flashing software, a cold start, or changing sensor configuration keys (CFG-SFCORE-*, CFG-SFIMU-* or CFG-SFODO-*). The calibration drive allows the software to detect and calibrate the sensors. See section Accelerated Initialization and Calibration Procedure for additional information. Performance is likely to be sub-optimal if the calibration drive is not performed correctly.
- If the maximum counter value of a wheel tick sensor cannot be represented as a power of 2 value, it must be configured manually. See section Odometer Types for additional information.

3.2.4 Installation configuration

This section describes the configurations for positioning and orientation of the NEO-M9L inside the vehicle.

3.2.4.1 Reference frames

This section describes the two reference frames used in NEO-M9L ADR implementation.

3.2.4.1.1 IMU frame

The IMU frame corresponds to the IMU instrumental frame within which the accelerometers and gyroscopes are assumed to be orthogonally mounted. The IMU frame is defined as follows:

- The origin of the IMU-frame is defined as the origin of the accelerometer sensor.
- By convention, an IMU contains three orthogonally-mounted accelerometers and three orthogonally-mounted gyroscopes which have the same orientation.
- The x-axis corresponds to the x-axis accelerometer/gyroscope sensing axis;
- The y-axis corresponds to the y-axis accelerometer/gyroscope sensing axis;
- The z-axis corresponds to the z-axis accelerometer/gyroscope sensing axis.

3.2.4.1.2 Installation frame

The installation frame is a right-handed 3D Cartesian frame rigidly connected with the vehicle, used to define the installation setup required for ADR navigation. The installation frame is defined as follows:

- The origin (O in Figure 2) is the IMU reference point (IRP);

- The x-axis points towards the front of the vehicle;
- The y-axis points towards the left of the vehicle;
- The z-axis completes the right-handed reference system by pointing up.

3.2.4.2 Reference points

This section describes the four reference points used in NEO-M9L ADR implementation.

3.2.4.2.1 Vehicle reference point (VRP)

The vehicle reference point (VRP) is defined as the center of the vehicle's rear axle. The VRP is the reference point at which the single tick or speed must be measured.

Use the CFG-SFODO-IMU2VRP configuration keys to configure the VRP. See the interface description [2] for more information.

3.2.4.2.2 Antenna reference point (ARP)

The antenna reference point (ARP) refers to the GNSS measurements i.e. antenna phase center.

ARP is the reference point of primary navigation NAV output until sensors are not initialized. ARP is always the reference point of the secondary navigation NAV2 output.

3.2.4.2.3 IMU reference point (IRP)

The IMU reference point (IRP) is defined as the center of the IMU chip within the module. The IRP is the reference point where the 3-dimensional movement is measured.

By default, the ADR navigation engine will calculate the navigation solution at the IRP.

3.2.4.2.4 Configurable reference point (CRP)

You can configure the configurable reference point (CRP) so that the navigation engine will calculate the navigation solution at the configured CRP.

Use the CFG-SFCORE-IMU2CRP configuration keys to configure the CRP. See the interface description [2] for more information.

3.2.4.3 IMU and GNSS antenna placement

It is important to correctly position the NEO-M9L and the GNSS antenna to achieve a consistent navigation solution. In particular:

- The NEO-M9L module and the GNSS antenna can be placed in any position on the vehicle. However, to achieve optimal performance, the ADR solution must be correctly configured. See the Solution type section for more information.
- In articulated vehicles (for example, truck and trailer), place the NEO-M9L on the front section of the vehicle.
- To achieve good positioning performance, position the GNSS antenna so that the mean C/N0 under good signal conditions is above 35 dBHz. If this is not possible, degraded performance may be observed.

3.2.4.3.1 IMU-to-antenna lever arm

The IMU-to-antenna lever arm is defined as the offset from the NEO-M9L IRP to the GNSS antenna center[1].

If the GNSS antenna is placed at a significant distance from the NEO-M9L module, errors can be introduced in the navigation solution (particularly when the vehicle experiences high heading rate). These errors can be compensated for by configuring the IMU-to-antenna lever arm offset in the installation frame.

[1] For best performance defining the antenna phase center is relevant, not the physical center of the antenna.

Use the CFG-SFIMU-IMU2ANT configuration keys to configure the IMU-to-antenna lever arm components. See the interface description [2] for more information.

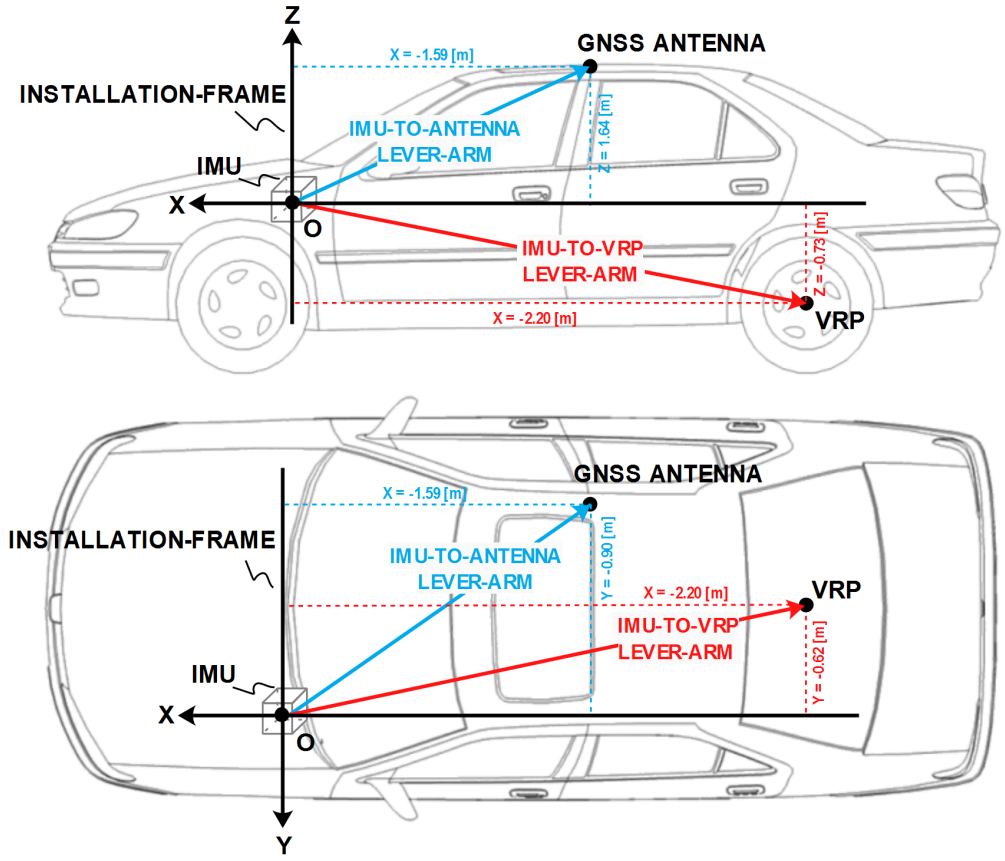


Figure 2: Lever arm configuration

3.2.4.3.2 IMU-to-VRP lever arm

The IMU-to-VRP lever arm is defined as the offset from the NEO-M9L IRP to the VRP in the installation frame.

If the NEO-M9L module is placed at a significant distance from the VRP, errors can be introduced in the navigation solution (particularly when the vehicle experiences high heading rate). These errors can be compensated for by configuring the IMU-to-VRP lever arm offset in the installation frame.

Use the CFG-SFODO-IMU2VRP configuration keys to configure the IMU-to-VRP lever arm components. See the interface description [2] for more information.

3.2.4.3.3 User-defined lever arm

The lever arm keys CFG-SFCORE-IMU2CRP, CFG-SFODO-IMU2VRP and CFG-SFIMU-IMU2ANT must be interpreted in the following way:

- **LA_X**: Corresponds to the x-axis component of the lever arm.
- **LA_Y**: Corresponds to the y-axis component of the lever arm.
- **LA_Z**: Corresponds to the z-axis component of the lever arm.

☞ To prevent degradation of the positioning solution the lever arms must be configured with a resolution of at least 1 cm.

3.2.4.4 IMU-mount alignment

This section describes how IMU-mount misalignment angles, that is, the angles which rotate the installation frame to the IMU-frame, can be configured.

The IMU-mount misalignment angles are defined as follows:

- The transformation from the installation frame to the IMU frame is described by three Euler angles about the installation frame axes denoted as *IMU-mount roll*, *IMU-mount pitch* and *IMU-mount yaw* angles. All three angles are referred to as the *IMU-mount misalignment angles*.

The default assumption is that the IMU-frame and the installation frame have the same orientation (that is, all axes are parallel). If the IMU-mount misalignment angles are slightly incorrect (typically a few degrees), the navigation solution can be degraded. If there are large (tens of degrees) IMU-mount misalignments, the position calculation may fail. Therefore, it is essential to correctly configure the IMU-mount misalignment settings.

☞ It is strongly recommended to use the automatic IMU-mount alignment as described in the following section.

3.2.4.4.1 Automatic IMU-mount alignment

The automatic IMU-mount alignment engine automatically estimates the IMU-mount roll, pitch and yaw angles. It requires an initialization phase during which no INS/GNSS fusion can be achieved (see the Fusion filter modes section for further details). The progress of the automatic alignment initialization can be monitored with the UBX-ESF-STATUS message, and/or with the UBX-ESF-ALG message providing more details. When the vehicle is subject to sufficient dynamics (i.e. left and right turns during a normal drive), the automatic IMU-mount alignment engine will estimate the IMU-mount misalignment angles. Once the automatic IMU-mount alignment engine has sufficient confidence in the estimated angles, the IMU-mount misalignment angles initialization phase is completed. The raw accelerometer and gyroscope data (that is, the IMU observations) are then compensated for IMU-mount misalignment and sensor fusion can begin. The resulting IMU-mount misalignment angles are output in the UBX-ESF-ALG message.

Enabling/disabling automatic IMU-mount alignment

The user can activate/deactivate the automatic IMU-mount alignment with the CFG-SFIMU-AUTO_MNTALG_ENA configuration key.

☞ If automatic IMU-mount alignment is deactivated while aligning, the estimated misalignment angles that were available at deactivation time are used (only if they were initialized, see the next section). If automatic IMU-mount alignment is reactivated, alignment is pursued by starting from the state where deactivation happened.

3.2.4.4.2 User-defined IMU-mount alignment

It is possible to configure the IMU-mount misalignment angles using the CFG-SFIMU-IMU_MNTALG configuration keys. The values that should be set in the configuration message are the Euler angles required to rotate the installation frame to the IMU-frame. The IMU-mount yaw rotation should be performed first, then the IMU-mount pitch and finally the IMU-mount roll. At each stage, the rotation is around the appropriate axis of the transformed installation frame, meaning that the order of the rotation sequence is important (see the figure below).

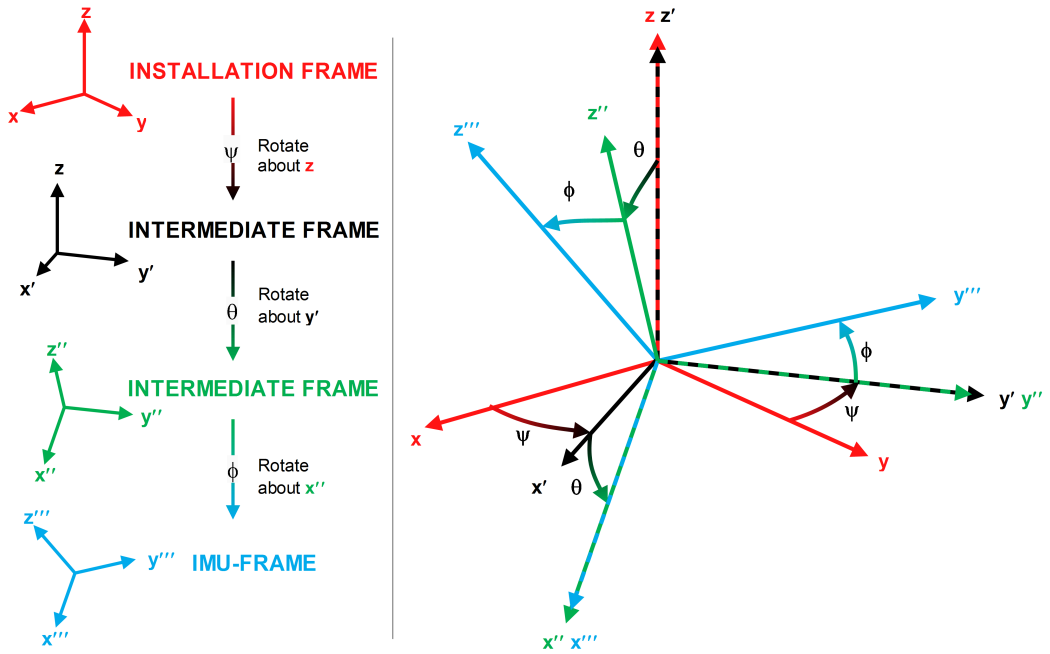


Figure 3: Euler angles

If there is only a single IMU-mount misalignment angle, it may be measured as shown in the three examples below.

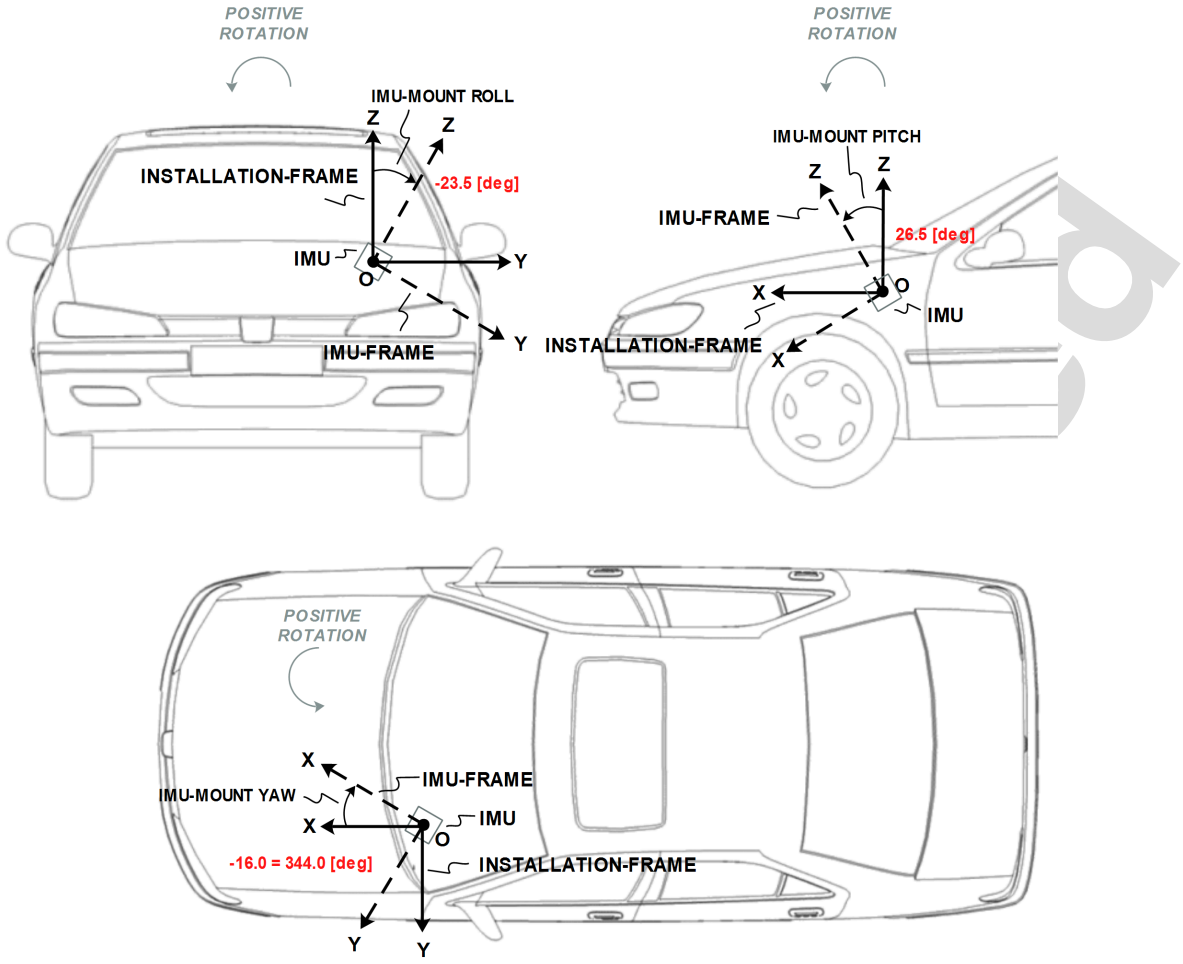


Figure 4: Installation frame

☞ In order to prevent significant degradation of the positioning solution the IMU-mount misalignment angles should be configured with an accuracy of less than 5 degrees.

The following list describes in detail how the CFG-SFIMU-IMU_MNTALG keys are to be interpreted with respect to the example illustrated in the figure above:

- **CFG-SFIMU-IMU_MNTALG_YAW**: The IMU-mount yaw angle (`yaw`) corresponds to the rotation around the installation frame z-axis (vertical) required for aligning the installation frame to the IMU frame (`yaw` = 344.0 degrees if the IMU-mount misalignment is composed of a single rotation around the installation frame z-axis, that is, with no IMU-mount roll and IMU-mount pitch rotation).
- **CFG-SFIMU-IMU_MNTALG_PITCH**: The IMU-mount pitch angle (`pitch`) corresponds to the rotation around the installation frame y-axis required for aligning the installation frame to the IMU-frame (`pitch` = 26.5 degrees if the IMU-mount alignment is composed of a single rotation around the installation frame y-axis, that is, with no IMU-mount roll and IMU-mount yaw rotation).
- **CFG-SFIMU-IMU_MNTALG_ROLL**: The IMU-mount roll angle (`roll`) corresponds to the rotation around the installation frame x-axis required for aligning the installation frame to the IMU frame (`roll` = -23.5 degrees if the IMU-mount misalignment is composed of a single

rotation around installation frame x-axis, that is, with no IMU-mount pitch and IMU-mount yaw rotation).

3.2.5 Sensor configuration

This section describes the external sensor configuration parameters.

3.2.5.1 Odometer configuration

Odometer is a generic term for wheel tick or speed sensor.

You can configure the odometer with the CFG-SFODO-* configuration keys.

☞ The NEO-M9L was designed to work with odometer input. Although the NEO-M9L calculates a position without odometer input (UDR mode), the accuracy of that position is compromised.

3.2.5.1.1 Odometer interfaces

Odometer data can be delivered to NEO-M9L via the following interfaces:

Hardware interface: NEO-M9L has a dedicated pin (WT) for analog wheel tick signal input, and another pin (DIR) dedicated to wheel tick direction signal.

- The WT pin is enabled with the CFG-SFODO-USE_WT_PIN key.
- The DIR pin polarity is automatically detected by the receiver by default. To manually configure the polarity, you must turn off automatic detection by setting the CFG-SFODO-DIS_AUTODIRPINPOL key and you must define the polarity in the CFG-SFODO-DIR_PINPOL key.
- Double-edge counting can be enabled via the CFG-SFODO-CNT_BOTH_EDGES key. It can increase performance with low-resolution wheel ticks, but is not suitable for all types of wheel tick signals. It **must not** be used with signals that are not generated with approximately 50% duty signal as it would impair performance.

Software interface: Odometer data can be delivered to the receiver over one of the communication interfaces. The data shall be contained in UBX-ESF-MEAS messages. UBX-ESF-MEAS (data type 10) shall be used for single-tick odometer data and UBX-ESF-MEAS (data type 11) shall be used for speed odometer data. See the interface description [2] for more information.

- By default, the receiver automatically ignores the WT pin if wheel tick/speed data are detected on the software interface. Therefore data coming from the software interface will be prioritized over data coming from the hardware interface. To disable the automatic use of data detected on the software interface, set the CFG-SFODO-DIS_AUTOSW key.
- While providing the speed data over software interface, the CFG-SFODO-COMBINE_TICKS should remain disabled.

3.2.5.1.2 Odometer types

NEO-M9L supports sensors delivering the following types of data:

- **Relative wheel tick data**: If the wheel tick sensor delivers relative wheel tick counts (that is, wheel tick count since the previous measurement), the CFG-SFODO-COUNT_MAX value must be set to 0.
- **Absolute wheel tick data**: If the wheel tick sensor delivers absolute wheel tick counts (that is, wheel tick count since startup at time tag 0) that always increase, regardless of driving forward or backward (with driving direction indicated separately). If the counter is configured to 1, the maximum absolute wheel tick counter value is automatically estimated by the receiver for a maximum counter value that can be represented as a 2^N value. Other maximum counter values must be manually configured. For example, a CFG-SFODO-COUNT_MAX=1024 roll-over value would be automatically estimated, but a CFG-SFODO-COUNT_MAX=1000

must be configured. The maximum counter value is configured by setting the CFG-SFODO-DIS_AUTOCOUNTMAX key and setting the CFG-SFODO-COUNT_MAX value to the upper threshold of the absolute wheel tick sensor count before starting again from zero (roll-over).

- **Speed data**: Data coming from this sensor type can only be delivered to the receiver via one of the communication ports within a UBX-ESF-MEAS (data type 11). The speed data shall be delivered in meters per second.

If speed data but no absolute or relative wheel tick data are detected, the receiver automatically uses the speed data without the need for reconfiguring the CFG-SFODO-USE_SPEED key. This behavior can be deactivated by setting the CFG-SFODO-DIS_AUTOSPEED key and by manually setting or clearing the CFG-SFODO-USE_SPEED key. If wheel tick data (or both wheel tick and speed data) are detected on the software interface, the receiver uses the data type (by default wheel tick data) corresponding to the configured CFG-SFODO-USE_SPEED key.

To make the receiver interpret incoming speed data (data type 11 in ESF-MEAS) instead of the single wheel tick data (data type 10 in ESF-MEAS) on the software interface, the CFG-SFODO-USE_SPEED key must be set.

☞ It is strongly recommended to use the absolute wheel tick sensors to ensure robust measurement processing even after sensor failures or outages.

3.2.5.1.3 Odometer settings

You can configure the following odometer settings:

- **Sampling frequency**: The wheel tick/speed data sampling frequency (CFG-SFODO-FREQUENCY) should be provided with an accuracy of approximately 10 %. If not provided, it is automatically determined during the initialization phase: this requires a consistent data rate and can take several minutes. Once initialized, the sampling frequency will be stored in a non-volatile storage. For optimal navigation performance, the standard wheel tick/speed input at 10 Hz is recommended.
- **Latency**: For best positioning performance, the latency of the wheel tick/speed data (CFG-SFODO-LATENCY) should be given as accurately as possible (to within at least 10 ms). If not provided, the wheel tick/speed data latency is assumed zero. More details about latency can be found in the Sensor Time Tagging section.
- **Quantization error**: If absolute/relative wheel tick data are used (for example, if the tick data is a distance), the quantization error can be defined in the CFG-SFODO-QUANT_ERROR key. The quantization error can be calculated as `2*Pi*R / T` with `R` the wheel radius, `T` the number of ticks per wheel rotation. If the quantization error is not provided, it is automatically initialized by the receiver.
- **Speed data accuracy (software interface only)**: If speed data are used, the speed data accuracy can be set in the CFG-SFODO-QUANT_ERROR key. If not provided, the speed data accuracy is automatically initialized by the receiver.
- **Scale factor**: If the coarse WT scale factor is not configured in the CFG-SFODO-FACTOR key, it is estimated automatically during the initialization (see section Initialization mode for more details).
- **Combination of multiple rear wheel ticks (software interface only)**: If wheel ticks are being received from both rear wheels, the receiver can be configured with the CFG-SFODO-COMBINE_TICKS key to use the combined rear wheel ticks rather than a single tick. It is recommended to use combined rear wheel ticks if available, as they are often of higher quality than the single ticks.

3.2.5.2 UBX-ESF-MEAS time tagging (software interface only)

To achieve optimal performance with the fusion solution it is essential to determine the epoch in the receiver time frame when the UBX-ESF-MEAS sensor data were taken. This may be done in either of the following ways:

- First byte reception: reception time of the first byte of the UBX-ESF-MEAS message
- Time mark on external input: reception time of time mark signal sent to external input

The latency of sensor data is the absolute difference between the time at which the sensor measurement is taken and the time at which either the first byte of the UBX-ESF-MEAS message or the pre-processor's time mark are detected at the receiver.

The latency for the wheel tick can be set with the `CFG-SFODO-LATENCY` key.

☞ It is essential that the time tags used for all UBX-ESF-MEAS data have the same resolution.

NEO-M9L automatically generates UBX-ESF-MEAS messages containing measurements from the internal IMU using the default time tag resolution of 1 millisecond. If the customer wants to input odometer data with UBX-ESF-MEAS messages, it is essential that the odometer time tag has a resolution of 1 millisecond.

3.2.5.2.1 First byte reception

The easiest way to determine the sensor measurement generation time is to have the GNSS receiver assume the time of reception of the first byte of the UBX-ESF-MEAS message (minus the constant configured latency in CFG-SFODO-LATENCY) to be the time of sensor measurement. This approach is the simplest to implement, but time mark on external input can yield better latency control and compensation.

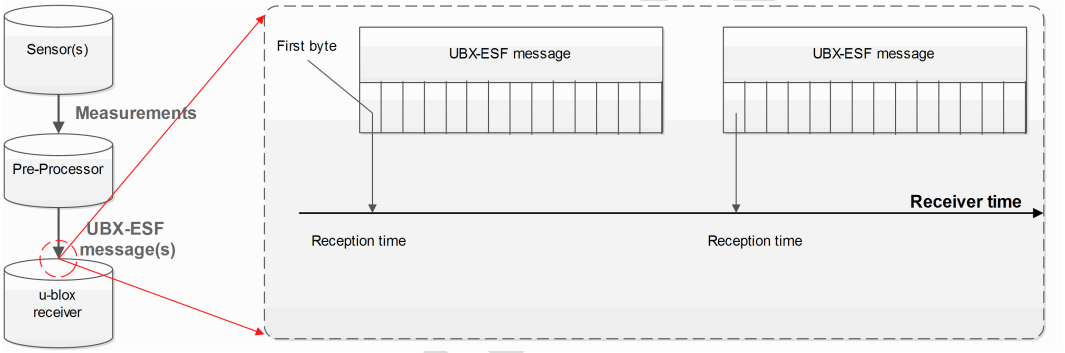


Figure 5: First byte reception

3.2.5.2.2 Time mark on external input

In this case, the preprocessor unit generating the measurements sends a signal to the EXTINT input of the GNSS receiver, marking the moment of the measurement generation. The subsequent UBX-ESF-MEAS message is then flagged accordingly, and the measurements in the message are assumed to have been generated at the time of external signal reception (minus the constant configured latency in CFG-SFODO-LATENCY). This approach is the preferred solution, but it can be difficult to realize an exact analog time signal for the preprocessor unit.

☞ The WT pin on NEO-M9L can be used as an EXTINT input. For this feature the WT pin should be disabled, i.e. CFG-SFODO-USE_WT_PIN = 0. Additionally, the CFG-TP-TP1_ENA needs to be enabled.

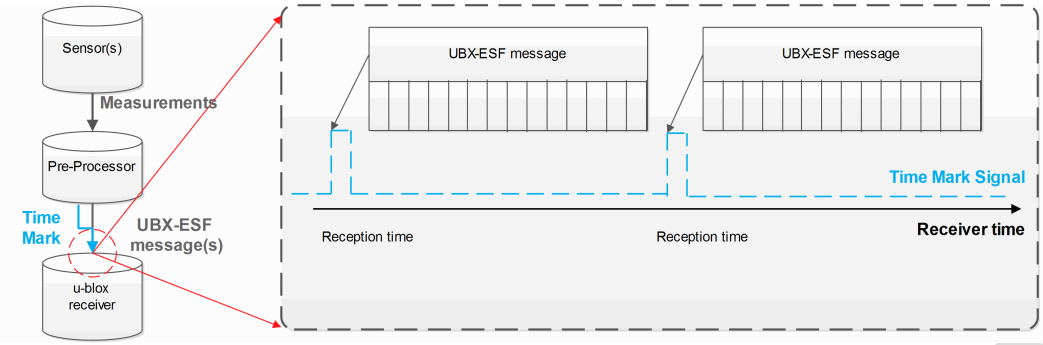


Figure 6: Time mark on external input

3.2.5.2.3 UBX-ESF-MEAS time tagging configuration

The time tag factor can be used to convert the sensor time tags into the required seconds.

☞ It is essential that the time tags used for all UBX-ESF-MEAS data have the same resolution.

NEO-M9L automatically generates UBX-ESF-MEAS messages containing measurements from the internal IMU using the default time tag resolution of 1 millisecond. If the customer wants to input odometer data with UBX-ESF-MEAS messages, it is essential that the odometer time tag has a resolution of 1 millisecond.

NEO-M9L automatically generates UBX-ESF-MEAS messages containing measurements from the internal IMU. If the customer inputs UBX-ESF-MEAS with odometer data with a different sensor time tag factor than the one used by NEO-M9L for IMU data, it will fail!

☞ The same sensor time tag (ttag) factor must be used for all UBX-ESF-MEAS data.

The following sensor time tagging settings need to be specified:

- **CFG-SFCORE-SEN_TTAG_FACT:** This parameter can be used to convert the sensor time tags from their original time unit into the required seconds. It has a resolution of 1 microsecond. The default value is 1000 (1 millisecond).
- **CFG-SFCORE-SEN_TTAG_MAX:** External sensor time tags can be encoded in different data types (signed/unsigned, varying number of bytes) which might vary across sensor types. For example, if the IMU raw packet's time tag field is encoded into an unsigned long integer (4 bytes), the maximum possible time tag value is `4294967295` (`0xFFFFFFFF`in hexadecimal).

3.2.6 ADR system configuration

3.2.6.1 Enabling/disabling fusion filter

The ADR feature can be enabled and disabled with the `CFG-SFCORE-USE_SF` key. If ADR is disabled, the receiver outputs a GNSS-only solution. IMU sensor measurements are still available in UBX-ESF-MEAS and UBX-ESF-RAW messages.

3.2.6.2 Recommended configuration

In general, it is recommended to use the default configuration values in order to achieve optimal ADR navigation performance.

By default, the navigation solution update rate is 1 Hz. It can be configured with the `CFG-RATE-NAV` key.

☞ At higher navigation rates, it is strongly recommended to check (and maybe reduce) the number of enabled output messages. CPU load, memory and interface bandwidth constraints may be a limiting factor.

3.2.7 Operation

This section describes how the ADR receiver operates.

3.2.7.1 Fusion filter modes

The fusion filter operates in different modes which are output in the UBX-ESF-STATUS message.

The table below summarizes the different fusion filter modes with the receiver's associated tasks.

Mode	Performed tasks / Possible causes	Published fix type
Initialization	Initialization of IMU Initialization of IMU-mount alignment Initialization of INS (position, velocity, attitude) IMU sensor error (e.g. missing data) detected	3D-fix (GNSS)
Fusion	Fine-calibration of IMU-mount misalignment angles Fine-calibration of IMU sensors Fine-calibration of wheel tick factors UDR fallback mode when missing WT data detected	GNSS/DR fix
Suspended fusion	Sensor error (e.g. missing data) detected Ferry detected	3D-fix (GNSS)
Disabled fusion	Fatal fusion filter error occurred Fusion filter turned off by user	3D-fix (GNSS)

Table 6: Fusion modes

More details about each fusion mode are given in the following sections.

3.2.7.1.1 Initialization mode

The purpose of the initialization phase is to estimate all unknown parameters which are required for achieving fusion. The initialization phase is triggered after a receiver cold start or a filter reset in case of fusion failure. The receiver is in initialization mode if the `fusionMode` field in the UBX-ESF-STATUS message is `0:INITIALIZING`. In this case the required sensor calibration status (`calibStatus`) is flagged as `0: NOT CALIBRATED` and the navigation solution output during initialization is based on GNSS solely.

The initialization phase comprises the following internal steps whose status is published in the `initStatus` field of the UBX-ESF-STATUS message:

- **IMU initialization:** Unknown crucial IMU parameters such as sensor sampling frequency are estimated during initialization. As long as all required IMU parameters are not initialized, the status of the IMU initialization (`imuInitStatus`) is flagged as `1:INITIALIZING` in the UBX-ESF-STATUS message. Moreover, the required sensor calibration statuses (`calibStatus`) are flagged as `0:NOT CALIBRATED` in the UBX-ESF-STATUS message. Note that if the user configured all required sensor settings, this step is skipped and IMU initialization is flagged as `2:INITIALIZED`.
- **IMU-mount alignment initialization:** If automatic IMU-mount alignment is enabled (see the Automatic IMU-mount alignment section), initial IMU-mount roll, pitch and yaw angles need to be estimated. For that, good GNSS signal reception as well as sufficient vehicle dynamics (i.e. a series of left and right turns during a normal drive) need to be at hand. As long as the IMU-mount alignment is not initialized, the status of the IMU-mount alignment (`mntAlgStatus`)

is flagged as 1:INITIALIZING in the UBX-ESF-STATUS message. Once initialized, the IMU-mount alignment status is flagged as 2:INITIALIZED. If no IMU-mount alignment is required, the IMU-mount alignment is flagged as 0:OFF. A detailed description of the automatic IMU-mount alignment operation can be found in the Automatic IMU-mount alignment section.

- **INS initialization:** Before entering fusion mode, the initial vehicle position, velocity and especially attitude (vehicle roll, pitch and heading angles) need to be known with sufficient accuracy. This is achieved during INS initialization phase using GNSS which comprises an INS coarse alignment step. As long as the fusion filter is not initialized, the status of the INS initialization (insInitStatus) is flagged as 1:INITIALIZING in the UBX-ESF-STATUS message. Once initialized, the INS initialization is flagged as 2:INITIALIZED.
- **Wheel tick sensor initialization:** Solution enters fusion mode (fusionMode field in the UBX-ESF-STATUS message is on 1:FUSION), even when wheel tick is not yet initialized, following a UDR mode approach described in UDR fallback mode section. WT sensor parameters, such as initial wheel tick factor, are estimated in parallel and are used once estimated with sufficient accuracy. As long as the wheel tick parameters are not initialized, the status of the wheel tick initialization (wtInitStatus) is flagged as 1:INITIALIZING in the UBX-ESF-STATUS message. Once initialized, the wheel tick sensor initialization is flagged as 2:INITIALIZED, WT data are used by the filter and the parameters are stored in non-volatile storage. If no wheel tick data are required, the wheel tick initialization is flagged as 0:OFF.
- **Sensor error (e.g. missing data) detected:** Sensor timeout of more than 500 ms will trigger an INS re-initialization.

Note that initialization phase requires good GNSS signal conditions as well as periods during which vehicle is stationary and moving (including left and right turns). Once all required initialization steps are achieved, fusion mode is triggered and continuous calibration begins.

3.2.7.1.2 Fusion mode

Once initialization phase is achieved, the receiver enters fusion mode. The receiver is in fusion mode if the UBX-ESF-STATUS.fusionMode field is set to 1:FUSION. The fusion filter then starts to compute combined GNSS/dead reckoning fixes (fused solutions) and to calibrate the sensors required for computing the fused navigation solution. This is the case when the sensor calibration status (UBX-ESF-STATUS.calibStatus) is set to 1:CALIBRATING. As soon as the calibration reaches a status where optimal fusion performance can be expected, (UBX-ESF-STATUS.calibStatus is set to 2/3:CALIBRATED.

3.2.7.1.3 UDR fallback mode

In case of WT sensor error / timeout (500 ms), or normal WT sensor initialization, or the WT sensor is not available, the solution falls back to full UDR (untethered dead reckoning) mode. The UBX-ESF-STATUS.fusionMode field still shows 1:FUSION when the receiver enters UDR fallback mode. However, the following flags can be used to determine when the receiver has entered UDR mode:

- The UBX-ESF-STATUS.wtInitStatus shows 1:INITIALIZING.
- The WT fault flag, UBX-ESF-STATUS.missingMeas field, is set to 1, indicating a WT timeout has been detected.

In the UDR fallback mode, the receiver does not lock the position while the vehicle is stationary, so the position could drift slightly, based on the GNSS conditions. The receiver needs WT for stationary phase detection.

3.2.7.1.4 Directionless odometer mode

This feature allows the use of odometer data for which the sign bit or the wheel tick (WT) pin polarity input is not trusted or not available. The directionless odometer support can be enabled by the configuration item CFG-SFODO-DIS_DIR_INFO=1.

After setting this configuration item, the directional information, that is, the polarity bit of the WT measurement and the sign of the wheel speed measurement in the UBX-ESF-MEAS message as well as the WT polarity pin input will be ignored. In particular, the following configuration items shall have no effect when used in combination with CFG-SFODO-DIS_DIR_INFO=1.

- CFG-SFODO-DIR_PINPOL
- CFG-SFODO-DIS_AUTODIRPINPOL

The performance with directionless odometer feature is expected to be worse than that of ADR with usual directional odometer data, but better than that of the UDR fallback mode.

☞ The odometer measurements shall be used as a velocity measurement only if the direction of motion (forwards/backwards) can be reliably determined based on other measurements (GNSS, IMU). Typically, this requires high enough speed.

3.2.7.1.5 Suspended fusion mode

Sensor fusion is temporarily suspended in cases where no fused solution should/can be computed. In this case, the receiver produces a GNSS-only solution. The receiver is considered to be in temporarily disabled fusion mode when the UBX-ESF-STATUS.fusionMode field is set to 2:SUSPENDED.

Fusion is suspended in the following scenarios:

- If one or several sensors deliver erroneous data or no data at all. Fusion is suspended during the sensor failure period. The receiver automatically recovers once the affected sensor or sensors are back to normal operation.
- If the vehicle is detected to be on a ferry or other moving platform, where odometer data does not indicate any displacement.

3.2.7.1.6 Disabled fusion mode

Sensor fusion can be permanently disabled if recurrent fusion failures occur. The receiver is considered to be in permanently disabled fusion mode if the UBX-ESF-STATUS.fusionMode field is set to 3:DISABLED. In this case, the receiver produces a GNSS-only solution if possible.

Fusion is permanently disabled in the following cases:

- If the fusion filter was manually turned off by the user with the CFG-SFCORE-USE_SF key.
- If the filter diverges due to significantly wrong installation or filter parameters.

3.2.7.2 Accelerated initialization and calibration procedure

This section describes how to perform fast initialization and calibration of the ADR receiver for evaluation purposes.

The duration of the initialization phases depends on the quality of the GNSS signals and the dynamics encountered by the vehicle.

You can shorten the initialization time required to reach sensor fusion mode by following the procedure in the order described in the table below.

Phase	Procedure	Indicator of success
IMU initialization	After receiver cold start or first receiver use, turn on car engine and stay stationary under good GNSS signal reception conditions for at least 3 minutes.	IMU initialization status UBX-ESF-STATUS.imuInitStatus shows 2:INITIALIZED.
INS initialization (position and velocity)	Once IMU is initialized, stay stationary under good GNSS signal reception conditions until a reliable GNSS fix can be achieved.	GNSS 3D fix achieved, good 3D position accuracy (at least 5 m), high number of used satellites (check UBX-NAV-PVT message).

Phase	Procedure	Indicator of success
IMU-mount alignment initialization	Start driving at a minimum speed of 30 km/h and do a series of approximately 10 left and right turns (at least 90 degrees). *You can skip this step if automatic IMU-mount alignment is turned off.*	IMU-mount alignment status (`UBX-ESF-STATUS.mntAlgStatus`) shows `2:INITIALIZED`, the IMU-mount alignment status `UBX-ESF-ALG.status` shows `3:COARSE ALIGNED`.
Wheel tick sensor initialization	Drive for at least 500 meters at a minimum speed of 30 km/h. To shorten this calibration step, drive the car at a higher speed (around 50 km/h) for at least 10 seconds under good GNSS visibility.	Wheel tick sensor initialization status `UBX-ESF-STATUS.wtInitStatus` shows `2:INITIALIZED`.
INS initialization (attitude)	Drive straight for at least 100 meters at a minimum speed of 40 km/h.	INS initialization status `UBX-ESF-STATUS.insInitStatus` shows `2:INITIALIZED`.

Table 7: Accelerated initialization procedure for automotive vehicles

Once initialization is completed, the `UBX-ESF-STATUS.fusionMode` field shows `1:FUSION`, combined GNSS/dead reckoning fixes (fused solutions) are output and the sensors used in the navigation filter start calibrating. Calibration is a continuous process running in the background, and it directly impacts the navigation solution quality.

You can shorten the calibration time required for reaching optimal ADR navigation performance by following the procedure described in the table below.

Phase	Procedure	Indicator of success
IMU-mount alignment calibration	Keep driving at a minimum speed of 30 km/h and do a series of left and right turns (at least 90 degrees). At each turn the estimated IMU-mount misalignment angles are refined and their accuracy increased. *You can skip this step if automatic IMU-mount alignment is turned off.*	Once the IMU-mount alignment engine has high confidence in its misalignment angle estimates, the IMU-mount alignment status `UBX-ESF-ALG.status` is set to `4:FINE ALIGNED`.
IMU calibration (gyroscope and accelerometer)	Drive curves and straight segments for a few minutes by including a few stops lasting at least 30 seconds each. This drive should also include some periods with higher speed (at least 50 km/h) and can typically be carried out on normal open-sky roads with good GNSS signal reception conditions.	The calibration status of the used sensors `UBX-ESF-STATUS.calibStatus` shows `2/3:CALIBRATED`.

Table 8: Accelerated calibration procedure for automotive vehicles

Table 9 describes how to perform fast initialization and calibration of the receiver, while using the e-scooter dynamic model for evaluation purposes.

Phase	Procedure	Indicator of success
IMU initialization	After receiver cold start or first receiver use, stay stationary under good GNSS signal reception conditions for at least 3 minutes.	IMU initialization status `UBX-ESF-STATUS.imuInitStatus` shows `2:INITIALIZED`.
INS initialization (position and velocity)	Once IMU is initialized, stay stationary under good GNSS signal reception conditions until a reliable GNSS fix can be achieved.	GNSS 3D fix achieved, good 3D position accuracy (at least 5 m), high number of used SVs (check UBX-NAV-PVT message).
Wheel tick sensor initialization	Drive for at least 100 meters at a minimum speed of 10 km/h.	Wheel tick sensor initialization status `UBX-ESF-STATUS.wtInitStatus` shows `2:INITIALIZED`.
INS initialization (attitude)	Drive straight for at least 100 meters at a minimum speed of 10 km/h.	INS initialization status `UBX-ESF-STATUS.insInitStatus` shows `2:INITIALIZED`.

Phase	Procedure	Indicator of success
IMU calibration (gyroscope and accelerometer)	Drive curves and straight segments for a few minutes by including a few stops lasting at least 30 seconds each. This drive should be carried out on normal open-sky roads with good GNSS signal reception conditions.	The calibration status of the used sensors `UBX-ESF-STATUS.calibStatus` shows `2/3:CALIBRATED`.

Table 9: Accelerated initialization and calibration procedure for e-scooters

Note that the calibration status (`UBX-ESF-STATUS.calibStatus`) of some used sensors might fall back to `1:CALIBRATING` if the receiver is operated in challenging conditions. In such a case, fused navigation solution uncertainty increases until optimal conditions are observed again for re-calibrating the sensors.

☞ In the presence of significant temperature gradient affecting the gyroscopes, the fused navigation performance might also depend on how well the temperature compensation table is populated. The table is gradually filled in while the vehicle is stationary and by observing gyroscope biases at different temperatures. Therefore the quality of the gyroscope temperature compensation depends on how many temperature bins could be observed while the vehicle was stationary and on the duration of the observation for each bin.

3.2.7.3 Automatic IMU-mount alignment

3.2.7.3.1 Alignment solution output

The IMU-mount misalignment angles are shown in UBX-ESF-ALG messages.

- **IMU-mount angle initialization**: During IMU-mount angle initialization the (`UBX-ESF-ALG.status` field is equal to `2`), and the published angles (`yaw, pitch and roll`) correspond to the current estimated values but are not yet applied for rotating the IMU observations.
- **IMU-mount angle initialization complete**: After initialization the (`UBX-ESF-ALG.status` field is equal to or higher than `3`), the published angles correspond to the estimated value and are applied for rotating the IMU observations.
- **Automatic IMU-mount alignment disabled**: If automatic IMU-mount alignment is disabled, the published angles correspond to the IMU-mount angles configured by the the user (see User-defined IMU-mount Alignment section) and are applied for rotating the IMU observations.

⚠ **CAUTION** If user-defined IMU-mount misalignment angles were configured by the user using CFG-SFIMU-IMU_MNTALG keys (see User-defined IMU-mount alignment section) and automatic IMU-mount alignment is active, the angles output in the UBX-ESF-ALG message still correspond to the definition given above: they represent the full rotation required for transforming IMU data from installation frame to IMU frame. This means that the output misalignment angles are computed from the composed rotation of the user-defined rotation and the internally-estimated rotation.

3.2.7.3.2 Alignment progress

The progress of the automatic IMU-mount alignment can be monitored by checking the `UBX-ESF-ALG.status` field.

- **IMU-mount roll/pitch angle initialization ongoing**: The alignment engine is initializing the IMU-mount roll and pitch angles (`UBX-ESF-ALG.status` is 1). Both angles can only be initialized if vehicle encounters left and right turns (as during a normal drive).
- **IMU-mount yaw angle initialization ongoing**: The alignment engine is initializing the IMU-mount yaw angle (`UBX-ESF-ALG.status` is 2). IMU-mount yaw angle can only be initialized once IMU-mount roll and pitch angles are initialized and if vehicle encounters left and right turns (as during a normal drive).

- **IMU-mount alignment coarse calibration ongoing**: Once initialized (`UBX-ESF-ALG.status` is 3), the automatic IMU-mount alignment engine has sufficient confidence in all IMU-mount misalignment angles and validates their use for compensating the accelerometer and gyroscope data (fused navigation solutions can be computed). The IMU-mount misalignment angles are filtered each time the observed vehicle dynamics are measurable.
- **IMU-mount alignment fine calibration ongoing**: Once the IMU-mount misalignment angles are estimated with a good accuracy, the automatic IMU-mount alignment engine becomes more conservative in updating the IMU-mount misalignment angles (`UBX-ESF-ALG.status` is 4).

3.2.7.3.3 Alignment reset

If for some reasons the IMU-mount misalignment angles estimated by the automatic IMU-mount alignment engine are no longer valid (for example due to a change in the physical mounting of the device), the IMU-mount misalignment angles can be reset by sending a UBX-ESF-RESETALG message. In addition, the misalignment angles are also reset in the following cases:

- If a cold start command is sent.
- If the user-defined IMU-mount misalignment angles are changed by sending CFG-SFIMU-IMU_MNTALG keys (see User-defined IMU-mount alignment section for more details.

The IMU-mount alignment engine then falls back into initialization mode.

3.2.7.3.4 Alignment errors

The following errors might be output in the `UBX-ESF-ALG.error` bit field:

- **IMU-mount roll/pitch angle error**: If the automatic IMU-mount alignment engine suspects wrong IMU-mount roll and/or IMU-mount pitch misalignment angles (either due to a wrong initialization or a change in the physical mounting of the device), the `UBX-ESF-ALG.error` bit 0 is set to 1.
- **IMU-mount yaw angle error**: If the automatic IMU-mount alignment engine suspects wrong IMU-mount yaw misalignment angle (either due to a wrong initialization or a change in the physical mounting of the device), the `UBX-ESF-ALG.error` bit 1 is set to 1.
- **Euler angle singularity ('gimbal-lock') error**: The Euler angle singularity `UBX-ESF-ALG.error` bit 2 is set when the automatic IMU-mount alignment engine detects an installation where the IMU frame is misaligned in such a way that a degree of freedom is lost when two IMU-mount misalignment (Euler) angles begin to describe the same rotations (or axes). This happens, for example, with an IMU-mount misalignment of +/- 90 degrees around the IMU-mount pitch axis, where IMU-mount roll and IMU-mount yaw cannot be distinguished from each other. In such a case, these IMU-mount misalignment angles start to heavily fluctuate with time due to the mathematical singularity occurring at these points, meaning that the IMU-mount misalignment angles output in the UBX-ESF-ALG are not stable in time. Note however that each individual set of IMU-mount misalignment angles output in such a case still describes the correct rotation. Moreover, the internal rotation applied for aligning the IMU readings does not suffer from this singularity issue and optimal fusion can still be achieved.

3.2.7.4 Navigation output

3.2.7.4.1 Local-level north-east-down (NED) frame

The local-level frame is a geodetic frame with the following features:

- The origin (O) is a point on the Earth's surface;
- The x-axis points to north;
- The y-axis points to east;
- The z-axis completes the right-handed reference system by pointing down.

The frame is referred to as North-East-Down (NED) since its axes are aligned with the North, East and down directions.

3.2.7.4.2 Vehicle frame

The vehicle frame is a right-handed 3D Cartesian frame rigidly connected with the vehicle and is used to determine the attitude of the vehicle with respect to the local-level frame. It has the following features:

- The origin (O) is the VRP;
- The x-axis points towards the front of the vehicle;
- The y-axis points towards the right of the vehicle;
- The z-axis completes the right-handed reference system by pointing down.

3.2.7.4.3 Vehicle position and velocity output

The position and velocity information is output in several messages, for example, UBX-NAV-PVT. Position and velocity computed by the ADR navigation filter are referenced to the IRP.

☞ The output is referenced to IRP when the sensors are calibrated and the lever arms are configured correctly. If the sensors are not calibrated the navigation output is referenced to the GNSS antenna phase center.

3.2.7.4.4 Vehicle position and velocity output in CRP

When setting the IMU-to-CRP lever arm, position and velocity computed by the ADR navigation filter are referenced to the CRP.

☞ After configuring the IMU-to-CRP lever arm, the output is referenced to CRP only when the sensors are calibrated. If the sensors are not calibrated the navigation output is referenced to the GNSS antenna phase center.

3.2.7.4.5 Vehicle attitude output

The transformation between the vehicle frame and the local-level frame is described by three attitude angles about the local-level axes denoted as *vehicle roll*, *vehicle pitch* and *vehicle heading*. All three angles are referred to as *vehicle attitude* and are illustrated in the figure below:

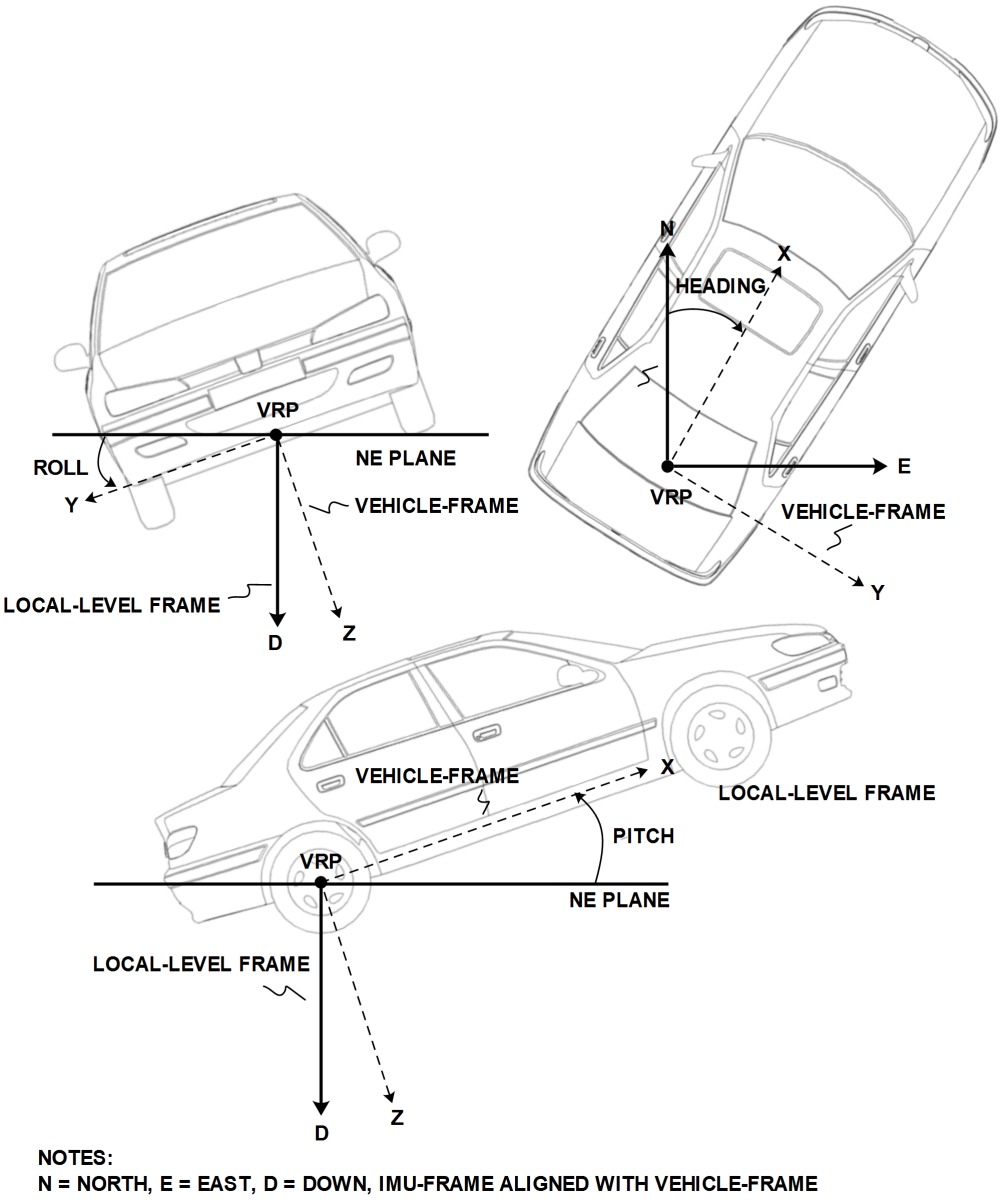


Figure 7: Vehicle attitude output

The order of the sequence of rotations around the navigation axes defining the vehicle attitude matrix in terms of vehicle attitude angles is illustrated below:

VEHICLE ATTITUDE DEFINITION

ϕ : Vehicle roll angle

θ : Vehicle pitch angle

ψ : Vehicle heading angle

$\mathbf{C}_b^n$: Rotation between body-frame (b) and local-level NED navigation-frame (n)

$$\mathbf{C}_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \quad \mathbf{C}_Y = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \quad \mathbf{C}_Z = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{C}_b^n = \mathbf{C}_Z^T \cdot \mathbf{C}_Y^T \cdot \mathbf{C}_X^T$$

$$= \begin{bmatrix} \cos(\theta)\cos(\psi) & \sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) \\ \cos(\theta)\sin(\psi) & \sin(\phi)\sin(\theta)\sin(\psi) + \cos(\phi)\cos(\psi) & \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) \\ -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix}$$

Figure 8: Vehicle attitude definition

Note that in this figure the body frame corresponds to the vehicle frame.

The vehicle attitude is output in the UBX-NAV-ATT message. The message provides all three angles together with their accuracy estimates.

3.2.7.4.6 Vehicle dynamics output

The UBX-ESF-INS message outputs information about vehicle dynamics provided by the INS, compensated vehicle angular rates, and compensated vehicle acceleration. The acceleration data is free of any gravitational acceleration. Its accuracy is directly dependent on the filter attitude estimation accuracy.

Compensated vehicle dynamics information is output with respect to the vehicle frame.

☞ The message outputs only dynamics information that is directly compensated by the fusion filter. This implies that depending on the solution type and the sensor availability, dynamics along some axes of the vehicle frame might not be available.

3.2.7.5 Sensor data types

The UBX-ESF-MEAS messages can be used as input and/or output messages. They can provide the following functionality:

- Output the NEO-M9L internal IMU measurements;
- Input external wheel tick or speed measurements from a host to NEO-M9L.

A different number of data fields may be used, and these can contain different types of measurements. The type of each measurement is specified in the `UBX-ESF-MEAS.dataType` field.

The supported sensor data types are:

Type	Description	Unit	Format of the 24 data bits
0	none, data field contains no data		
1...4	reserved		
5	z-axis gyroscope angular rate	deg/s *2^-12	signed
6...7	reserved		

Type	Description	Unit	Format of the 24 data bits
8	rear-left wheel ticks		Bits 0-22: unsigned tick value. Bit 23: direction indicator (0=forward, 1=backward)
9	rear-right wheel ticks		Bits 0-22: unsigned tick value. Bit 23: direction indicator (0=forward, 1=backward)
10	single tick (speed tick)		Bits 0-22: unsigned tick value. Bit 23: direction indicator (0=forward, 1=backward)
11	speed	m/s * 1e-3	signed
12	gyroscope temperature	deg Celsius * 1e-2	signed
13	y-axis gyroscope angular rate	deg/s *2^-12	signed
14	x-axis gyroscope angular rate	deg/s *2^-12	signed
15	reserved		
16	x-axis accelerometer-specific force	m/s^2 *2^-10	signed
17	y-axis accelerometer-specific force	m/s^2 *2^-10	signed
18	z-axis accelerometer-specific force	m/s^2 *2^-10	signed
19...63	reserved		

Table 10: Definition of data types

3.2.7.6 Raw sensor data output

The UBX-ESF-RAW message can be used to access raw sensor measurements. A variable number of data fields may be used in a single message and these can contain different types of measurements. The type of each measurement is specified in the `UBX-ESF-RAW.dataType` field.

The possible sensors for the `data` field are accelerometer, gyroscope and temperature readings.The data types are the same as described in the Sensor Data Types section.

The measurements are available at a fixed rate. The sampling rate or other sensor configuration options cannot be changed, they depend on the inertial sensors connected. The raw measurements are output as one sample of every data type per message.

To enable this feature, the `UBX-ESF-RAW` message must be enabled using `CFG-MSGOUT-UBX_ESF_RAW-*` keys. If any non-zero rate is selected the message will be output at the rate at which the sensors are sampled.

☞ The feature can only be enabled if the active inertial sensors are connected to the GNSS receiver directly via hardware interface.

☞ Turning on this feature does not disable sensor fusion in the receiver.

3.2.7.6.1 Interrupt-based IMU sensor output

Applications which require lower latency and jitter can utilize the interrupt-based IMU sensor output feature. This is done by setting the configuration key `CFG-HW-SENS_IRQ_EN = 1`.

- After setting the configuration key `CFG-HW-SENS_IRQ_EN = 1`, the configuration is saved to flash and BBR.
- Next, the receiver needs to have this value set at start-up, so a reboot is needed after setting it for the first time for the feature to be enabled.
- Once enabled, the low latency IMU sensor output is available via `UBX-ESF-RAW` message.
- To disable the interrupt-based IMU sensor output, the configuration key `CFG-HW-SENS_IRQ_EN` is set to `0`, the configuration is saved in the flash and BBR, afterward the receiver needs to rebooted.

3.2.7.7 Calibrated sensor data output

The UBX-ESF-CAL message can be used to access calibrated sensor measurements. A variable number of data fields may be used in a single message in order to contain different types of measurements. The possible sensors for the data field are accelerometer, gyroscope and temperature readings. The type of the measurement is specified in the `UBX-ESF-CAL.dataType` field and is defined according to the Sensor Data Types section.

This message can be useful whenever there is a need of high rate IMU data (e.g. 100 Hz) that is calibrated in real-time by the sensor fusion algorithm using the most recent estimates. The message format is similar to UBX-ESF-RAW but there are some differences. More information can be found in the interface description [2].

Accelerometer and gyroscope data is output at a fixed rate. Temperature readings, differently from UBX-ESF-RAW, may be output at a lower rate (e.g. 1 Hz) to save bandwidth. For example if IMU rate is 100 Hz, roughly only one out of 100 consecutive messages includes a temperature reading. This is because raw temperatures are not calibrated but the user can still access them through UBX-ESF-CAL if needed, without having to enable UBX-ESF-RAW.

Calibrated data is produced at any time, regardless of the position fix type. Meaning that the data from UBX-ESF-RAW and UBX-ESF-CAL is the same during the initial startup phase when the sensors are not calibrated. The user can regularly check the calibration status via `UBX-ESF-STATUS.fusionMode flag`.

Sensor data is reported in the IMU frame and at the IMU reference point (IRP), i.e. same representation as in the UBX-ESF-RAW. The user can use information from UBX-ESF-ALG to rotate the data into the installation frame, if needed.

The UBX-ESF-CAL message is disabled in the default configuration and must be enabled using `CFG-MSGOUT-UBX_ESF_CAL*` keys. If a non-zero rate is selected, the message will be output at the rate at which the sensors are sampled.

UBX-ESF-RAW and UBX-ESF-CAL are independent messages, and both of them can be separately enabled, depending on the application. If UBX-ESF-RAW is enabled with UBX-ESF-CAL, the two messages have the same update rate and are aligned with their timestamps.

⚠ Enabling both UBX-ESF-RAW and UBX-ESF-CAL at the same time can easily overload the communication port if port speed is insufficient or many other messages are enabled.

⚠ The feature can only be enabled if the active inertial sensors are connected to the GNSS receiver directly via hardware interface.

3.2.7.8 Receiver startup and shutdown

Continuous dead reckoning is possible over receiver restarts if all following conditions are true:

- Non-volatile storage is available, the save-on-shutdown feature (SOS), or the advanced calibration handling feature is used
- A real-time clock is available or time assistance is provided on startup
- The sensor data stream is only stopped when the vehicle is stationary
- The vehicle is not moved while the receiver is off

3.2.8 Priority navigation mode

3.2.8.1 Introduction

NEO-M9L provides a low-latency position, velocity, and vehicle attitude solution to be output at a high rate by utilizing sensor-based propagation in between GNSS-measurement updates, thus prioritizing the time-critical data.

The receiver issues priority navigation messages first, and non-priority navigation messages when time allows it. The non-priority messages might, therefore, be output with some delay.

The following tables list priority navigation messages.

UBX protocol message	Content	Priority level	Output rate
UBX-NAV-PVT	Position, velocity, heading and time data	High	0-50 Hz (Configurable)
UBX-NAV-PVAT	Position, velocity, attitude and time data	High	0-50 Hz (Configurable)
UBX-NAV-HPPOSECEF	High-precision position solution in ECEF	High	0-50 Hz (Configurable)
UBX-NAV-POSECEF	Position solution in ECEF	High	0-50 Hz (Configurable)
UBX-NAV-HPPOSLLH	High-precision geodetic position solution	High	0-50 Hz (Configurable)
UBX-NAV-POSLLH	Geodetic position solution	High	0-50 Hz (Configurable)
UBX-NAV-ATT	Attitude data	High	0-50 Hz (Configurable)
UBX-ESF-INS	INS data	High	0-50 Hz (Configurable)
UBX-NAV-VELECEF	ECEF velocities	High	0-50 Hz (Configurable)
UBX-NAV-VELNED	NED velocities	High	0-50 Hz (Configurable)

Table 11: UBX priority navigation messages

NMEA protocol message	Content	Priority level	Output rate
NMEA-Standard-DTM	Datum info	High	0-50 Hz (Configurable)
NMEA-Standard-RMC	Recommended minimum data	High	0-50 Hz (Configurable)
NMEA-Standard-VTG	Course over ground and ground speed	High	0-50 Hz (Configurable)
NMEA-Standard-GNS	GNSS fix data	High	0-50 Hz (Configurable)
NMEA-Standard-GGA	Global positioning system fix data	High	0-50 Hz (Configurable)
NMEA-Standard-GLL	Latitude and longitude, with time of position fix and status	High	0-50 Hz (Configurable)
NMEA-Standard-THS	True heading and status	High	0-50 Hz (Configurable)
NMEA-PUBX-POSITION	Latitude and longitude position data	High	0-50 Hz (Configurable)

Table 12: NMEA priority navigation messages

☞ The NEO-M9L requires an initialization phase if the sensors have not been calibrated. During this initialization, the receiver delivers GNSS-only navigation solution data, still ensuring high-rate and low-latency output. The NEO-M9L works optimally in priority navigation mode when the IMU and WT sensors have been calibrated, and the alignment angles are correct.

☞ If priority navigation mode is enabled, comparing time information of a non-priority UBX message with a priority NMEA message may not be sensible (see the iTOW timestamps section). Thus, it is recommended to compare messages with the same priority level. For instance, comparing the time information of a non-priority UBX message with a non-priority NMEA message (such as NMEA-Standard-ZDA).

☞ When switching back from priority mode to non-priority mode, the TOW could jump back in time, similarly, switching from non-priority to priority mode may cause TOW to jump ahead in time.

3.2.8.2 Configuration

You can configure the priority navigation mode output using the CFG-RATE-NAV_PRIO configuration key.

☞ If a high priority navigation rate has been configured with CFG-RATE-NAV_PRIO, it is strongly recommended to check (and maybe reduce) the number of enabled output messages. Interface bandwidth constraints may be a limiting factor.

As mentioned in the table above, the allowed range for the priority navigation mode is 0-50 Hz.

When not zero, the receiver outputs navigation data as a set of messages with two priority levels:

1. Priority navigation mode: the navigation solution data is computed and output with high rate and low latency.
2. Non-priority navigation mode: the navigation data is computed and output with low rate and high latency.

When zero, the receiver outputs the navigation data as a set of messages with the same priority level i.e. non-priority navigation mode.

3.2.9 Advanced calibration handling

The advanced calibration handling feature enables a host to poll and later send sensor initialization and calibration parameters. This information can be used to aid quick recovery of the receiver into sensor fusion or dead reckoning mode following a device reset or power down, especially when the battery backup is not provided or the save-on-shutdown feature is not used.

The following points outline the procedure when using the advanced calibration handling feature:

- Host should poll `UBX-MGA-SF` regularly (optimal time: every 5 min.) after the fusion mode has been achieved. The `UBX-ESF-STATUS.fusionMode` field should be set to `1:FUSION` before polling the `UBX-MGA-SF` message.
- Receiver replies with one or more `UBX-MGA-SF` messages. These messages should be stored at the host side.
- In the event of a reset or a power down, host can send the latest stored `UBX-MGA-SF` messages to the receiver. If there is a sudden power off, the user has to provide the last known position and the attitude solution to the receiver after the power cycle.
- The receiver attempts to reach sensor fusion mode using the data available in these messages.
- Depending on the availability of the GNSS signals the receiver can switch to either fusion mode (3D + DR) or dead reckoning mode (DR only). Host can poll `UBX-ESF-STATUS` to track the status of the sensors' calibration.

☞ `UBX-MGA-SF` message is not output periodically, therefore it is required that the host polls the message regularly. This ensures that the host saves the latest parameters required for quick recovery into sensor fusion mode.

☞ Make sure to send the latest `UBX-MGA-SF` messages to the receiver. Sending outdated messages could mean the receiver will take longer to reach sensor fusion mode.

☞ Only send the saved `UBX-MGA-SF` messages to the same receiver they were obtained from.

☞ After the power cycle, the user needs to provide the last known position and the attitude via `UBX-MGA-INI-POS` and `UBX-MGA-INI-ATT` messages. The information can be taken from `UBX-NAV-PVT` and `UBX-NAV-ATT` or `UBX-NAV-PVAT`.

☞ Sending only the `UBX-MGA-SF` message will be rejected by the receiver even if the receiver is in 3D-only mode.

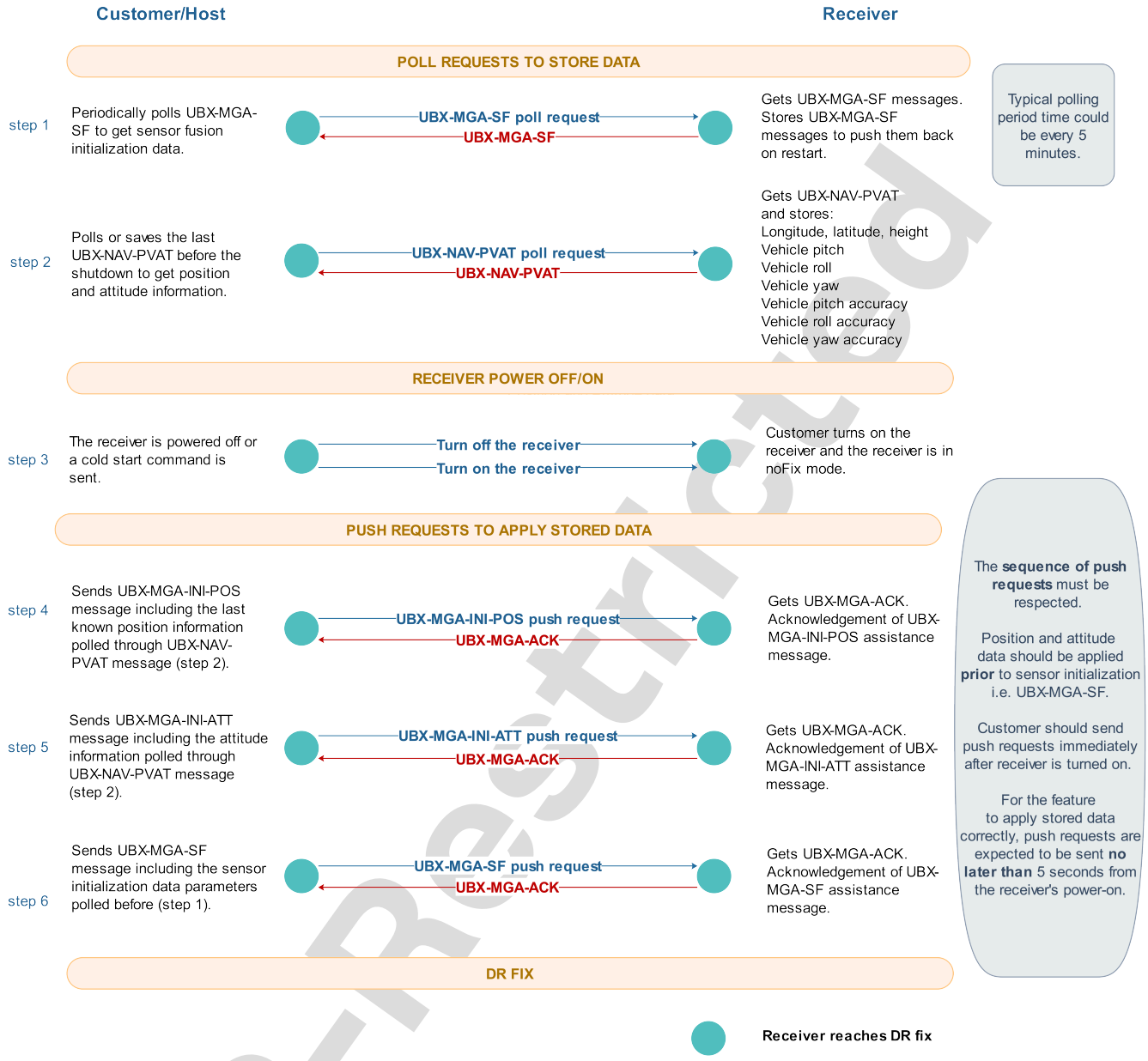


Figure 9: Advanced calibration handling operational steps

3.2.10 Wake on motion feature

3.2.10.1 Introduction

NEO-M9L supports the wake on motion feature.

This feature utilizes the on-board IMU sensor's interrupt pin to wake up the module, the host or both. The receiver should be in the software backup mode for this feature to work. The software backup mode acts as a sleep mode for the receiver. IMUs are capable of sending discrete signals on acceleration events while the sensor itself could be in low power consumption mode.

3.2.10.2 Configuration

Enabling the wake on motion feature requires two configurations:

- CFG-HW-SENS_WOM_MODE: Is used to enable and set the mode of the wake on motion.
- CFG-HW-SENS_WOM_THLD: Is used to set the required acceleration on a single accelerometer axis for triggering the wake up.

With CFG-HW-SENS_WOM_MODE, the receiver can be configured with four different wake on motion modes. By default the feature is disabled so CFG-HW-SENS_WOM_MODE is set to 0. With CFG-HW-SENS_WOM_MODE set to 1, the IMU interrupt would wake up the receiver and no discrete signal will be output by the receiver over the WOM pin. For the use cases where only the host needs to wake up, the CFG-HW-SENS_WOM_MODE should be set to 2. Now the receiver will output a discrete signal over the WOM pin whenever motion is detected, but the feature will not wake up the receiver. The last option is where the CFG-HW-SENS_WOM_MODE is set to 3, with this mode the IMU interrupt will wake up both the receiver and output a discreet signal over the WOM pin for the host. The receiver will start normal operation afterwards.

For CFG-HW-SENS_WOM_THLD the value ranges between 1-255, where 1 corresponds to 0 g and 255 to 1 g, where g = 9.81 m/s^2. e.g. for 0.25 g threshold the configured value should be 64. The default value for CFG-HW-SENS_WOM_THLD is 0, which corresponds to the threshold of 0.5 g. The threshold is configured for all axis, so if the acceleration on any axis exceeds the threshold, the IMU interrupt would be triggered and would wake up the host or the receiver depending on the configured mode.

3.2.10.3 Procedure to use the wake on motion feature

The following outlines the procedure when using the wake on motion feature:

- Set the required mode for the wake on motion feature via CFG-HW-SENS_WOM_MODE.
- Depending on the use case, set the required threshold for a single accelerometer axis via CFG-HW-SENS_WOM_THLD.
- Set the desired configuration to all layers and send it to the receiver.
- Afterwards, power cycle the receiver. This will initialize the sensors for wake on motion feature. This only needs to be done once the wake on motion configuration is changed.
- The next step is to set the receiver in software backup mode. The receiver can be set to software backup mode via UBX-RXM-PMREQ message. Duration of the requested task should be set to zero.
- Afterwards, based on the threshold and the mode the IMU interrupt would wake up the receiver for normal operations.

☞ In UBX-RXM-PMREQ message the wakeupSources.extint0 flag should not be disabled for the wake on motion feature.

☞ The test setup should be as close to the final integration as possible, as the detection threshold will vary depending on the place of installation/dampening.

3.2.10.4 Example use cases

The following CFG-HW-SENS_WOM_THLD values correspond to the bare minimum acceleration that would be produced in each scenario and should be used as a guideline, the specific thresholds will depend significantly on the final installation:

- Opening the door and entering the car: 15 (0.06 g)
- Starting the vehicle and driving out slowly: 25 (0.1 g)
- A bump on a parked car: 50 (0.2 g)

- Hard stopping the vehicle: 200 (0.78 g)

3.2.11 Map-matching input

3.2.11.1 Introduction

The map-matching input feature allows a map-matched position to be fed back to the receiver to aid the next navigation solution and improve positioning performance, as long as the map matching is accurate. The data can be supplied using the UBX-AID-MAPM input message.

3.2.11.2 Input data

The UBX-AID-MAPM message may be used to input WGS84 latitude, longitude and altitude as well as heading of motion. Alternatively, altitude can be given as height above mean sea level. The GNSS reference time defines the epoch of the map-matched navigation solution. The latency of the data must not exceed three epochs and should be minimized to ensure best positioning performance. The supplied accuracy of the horizontal position, altitude, and heading should indicate the quality of the map-matching data. The accuracy of these factors are used to determine the relative weight of the map-matched data when merging with the receiver solution. The status of the map matching feature is indicated in the fix status flags of message UBX-NAV-STATUS.

3.2.11.3 Recommendations

Map matching input implements a closed feedback loop. The map-matched solution is based on the receiver's position output and, if fed back, influences the receiver's next fix. Therefore the reliability of the map matching input data is critical for both the receiver and the map matching provider. It is recommended to only feed the map-matched solution back to the receiver if there is a high confidence that it is correct. Incorrect map matching feedback may cause the position solution to diverge from the true position in an unpredictable manner. If in doubt, it is better not to feed any information back to the receiver.

3.3 Primary and secondary output

3.3.1 Introduction

u-blox GNSS receivers output various navigation results and data calculated as part of the navigation solution. These include results such as position, altitude, velocity, status flags, accuracy estimate figures, satellite/signal information and more.

The NEO-M9L can provide this output in two streams:

- **Primary output:** Reports the results of a full navigation solution using all capabilities of the NEO-M9L, such as, for example, sensor fusion.
- **Secondary output:** Reports the results of a GNSS standalone navigation solution.

Both the primary output and secondary output provide a similar set of information but the two outputs report different results. The primary output is reported in the form of UBX-NAV-* messages, while the secondary output is reported in the form of UBX-NAV2-* messages. Therefore, the UBX message class can be used to distinguish between the primary output and the secondary output. For the specification of the UBX-NAV2-* messages and for a full list of available UBX-NAV2-* messages, see the applicable interface description [2].

☞ The secondary output may also be available as NMEA-NAV2-* messages. To confirm availability of NMEA-NAV2-* messages, see the applicable interface description [2]. NMEA-NAV2-* messages can only be output if the configured NMEA version is NMEA 4.0 or later.

For simplicity this section refers only to UBX-NAV2-* messages but the same information applies for NMEA-NAV2-* messages.

☞ The secondary output is complementary to the primary output. It does not provide the full navigation solution of the primary output. It can be used to expand the applications of the NEO-M9L to enable using a second navigation solution in parallel with the primary navigation solution.

The rest of this section describes how to configure and use the secondary output, what is the expected output behavior, and provides examples that illustrate potential uses for the secondary output, while highlighting the differences between the primary and the secondary output.

3.3.2 Configuration

Configuring the secondary output to the application's needs requires:

- Enabling the secondary output
- Configuring the desired secondary output UBX-NAV2-* messages
- Optionally, configuring the properties of the secondary output navigation solution

The configuration items relevant to the secondary output are in the CFG-NAV2-* configuration group. The configuration items for enabling and configuring the output rate of the UBX-NAV2-* messages are in the CFG-MSGOUT-* group and are of the form CFG-MSGOUT-UBX_NAV2_*. An example set of secondary output configuration items is shown in the table below. For all available configuration items, see the applicable interface description [2].

Configuration item	Description
CFG-NAV2-OUT_ENABLED	Enables secondary output
CFG-NAV2-SBAS_USE_INTEGRITY	Enables using SBAS integrity information in the secondary output
CFG-MSGOUT-UBX_NAV2_PVT_*	Enables UBX-NAV2-PVT secondary output message
CFG-MSGOUT-UBX_NAV2_TIMEGPS_*	Enables UBX-NAV2-TIMEGPS secondary output message

Table 13: Example secondary output configuration items

Enabling the secondary output: The first necessary step to enable the secondary output is to configure the CFG-NAV2-OUT_ENABLED configuration item appropriately. This will enable the secondary output navigation solution to run in parallel with the primary output navigation solution. By default, the secondary output is disabled. Note that if you do not follow the next step, there will be no secondary output visible in the NEO-M9L communication interfaces in the form of UBX-NAV2-* messages.

☞ Both primary and secondary output report a navigation solution computed at the same navigation rate. Enabling the secondary output may affect the maximum achievable navigation update rate due to the extra computational load.

Configuring the desired secondary output UBX-NAV2-* messages: The second necessary step is to configure the desired CFG-MSGGOUT-UBX_NAV2_* configuration items appropriately. These set the message output rates for the UBX-NAV2-* messages that you wish to output. By default, all UBX-NAV2-* message output rates are set to 0 and as such are not being output.

☞ Due to the increased message output, the interface load will be higher while the secondary output messages are enabled. Therefore, the interface baud rate may need to be adapted accordingly. Alternatively, it is possible to configure the UBX-NAV2-* messages with a different output rate from that of their primary output UBX-NAV-* counterparts.

Configuring the properties of the secondary output navigation solution: Optionally, it is possible to configure the properties of the secondary output navigation solution in order to adapt it to the application's needs.

A minimal subset of the primary output navigation solution configuration is available for the secondary output navigation solution configuration. All such available configuration items are in the CFG-NAV2-* configuration group (see applicable interface description [2]).

Configuring any of the CFG-NAV2-* configuration items changes the behavior of the secondary output navigation solution only and not the primary output one. All such configuration items have a primary output configuration counterpart and have the same default value as their primary output configuration counterpart.

For example, the CFG-NAV2-SBAS_USE_INTEGRITY configuration item allows configuring the SBAS integrity feature differently for the primary output and the secondary output. Its primary output counterpart is the CFG-SBAS-USE_INTEGRITY configuration item and the default value of both configuration items is the same.

3.3.3 Expected output behavior

Once the secondary output is enabled and the desired secondary output UBX-NAV2-* messages are configured, the NEO-M9L will output both primary and secondary output data in the form of the enabled UBX-NAV-* and UBX-NAV2-* messages respectively.

In every navigation epoch, a set of UBX-NAV-* messages will be output followed by another set of UBX-NAV2-* messages. Both sets will be referring to the navigation solution of the same navigation epoch.

Each set will be delimited at its end with a UBX-NAV-EOE or a UBX-NAV2-EOE message respectively. In other words, a UBX-NAV-EOE message will be output at the end of the UBX-NAV-* class enabled messages and a UBX-NAV2-EOE message will be output at the end of the UBX-NAV2-* class enabled messages. For example, if only UBX-NAV-PVT, UBX-NAV2-PVT, UBX-NAV-TIMEGPS and UBX-NAV2-TIMEGPS are enabled on the same port with message output rate 1, then every navigation epoch output will be as follows: UBX-NAV-PVT, UBX-NAV-TIMEGPS, UBX-NAV-EOE, UBX-NAV2-PVT, UBX-NAV2-TIMEGPS, UBX-NAV2-EOE.

☞ Secondary output messages appear after the primary output messages. This results in a higher latency for the secondary output messages than the primary output messages.

☞ Contrary to UBX-NAV2-* messages, secondary output NMEA-NAV2-* messages are not delimited by an NMEA-equivalent to UBX-NAV-EOE.

The specification of the UBX-NAV2-* messages resembles that of the UBX-NAV-* messages. The payload specification of a UBX-NAV2 message is identical to the payload specification of its UBX-NAV counterpart, allowing to easily adapt any existing message parsers. The primary output will contain results and data reflecting the full navigation solution of the NEO-M9L. The secondary output will contain results and data reporting a GNSS standalone navigation solution.

3.3.4 Example use cases

As an example, an application using a NEO-M9L that is operating in sensor fusion mode can enable the secondary output and monitor a GNSS standalone solution to understand the improvements introduced by using sensor fusion.

3.4 SBAS

The NEO-M9L is capable of receiving multiple SBAS signals concurrently, even from different SBAS systems (WAAS, EGNOS, MSAS, etc.). They can be tracked and used for navigation simultaneously. Every SBAS satellite that broadcasts ephemeris or almanac information can be used for navigation, just like a normal GNSS satellite.

For receiving correction data, the NEO-M9L automatically chooses the best SBAS satellite as its primary source. It will select only one since the information received from other SBAS satellites is redundant and could be inconsistent. The selection strategy is determined by the proximity of the satellites, the services offered by the satellite, the configuration of the receiver (test mode allowed/ disallowed, integrity enabled/disabled) and the signal link quality to the satellite.

If corrections are available from the chosen SBAS satellite and used in the navigation calculation, the differential status will be indicated in several output messages such as UBX-NAV-PVT, UBX-NAV-STATUS, UBX-NAV-SAT, NMEA-GGA, NMEA-GLL, NMEA-RMC and NMEA-GNS (see the applicable interface description [2]). The message UBX-NAV-SBAS provides detailed information about which corrections are available and applied.

The most important SBAS feature for accuracy improvement is ionosphere correction. The measured data from regional Ranging and Integrity Monitoring Stations (RIMS) are combined to make a Total Electron Content (TEC) map. This map is transferred to the receiver via SBAS satellites to allow a correction of the ionosphere error on each received signal.

Message type	Message content	Source
0(0/2)	Test mode	All
1	PRN mask assignment	Primary
2, 3, 4, 5	Fast corrections	Primary
6	Integrity	Primary
7	Fast correction degradation	Primary
9	Satellite navigation (ephemeris)	All
10	Degradation	Primary
12	Time offset	Primary
17	Satellite almanac	All
18	Ionosphere grid point assignment	Primary
24	Mixed fast / long-term corrections	Primary
25	Long-term corrections	Primary
26	Ionosphere delays	Primary

Table 14: Supported SBAS messages

Each satellite services a specific region and its correction signal is only useful within that region. Planning is crucial to determine the best possible configuration, especially in areas where signals from different SBAS systems can be received:

- **Example 1 - SBAS receiver in North America:** In the eastern parts of North America, make sure that EGNOS satellites do not take preference over WAAS satellites. The satellite signals from the EGNOS system should be disallowed by using the PRN mask.
- **Example 2 - SBAS receiver in Europe:** Some WAAS satellite signals can be received in the western parts of Europe, therefore it is recommended that the satellites from all but the EGNOS system should be disallowed using the PRN mask.

☞ Although u-blox receivers try to select the best available SBAS correction data, it is recommended to configure them to exclude unwanted SBAS satellites.

To configure the SBAS functionalities use the CFG-SBAS-* configuration group.

Parameter	Description
CFG-SIGNAL-SBAS_ENA	Enabled/disabled status of the SBAS subsystem
CFG-SBAS-USE_TESTMODE	Allow/disallow SBAS usage from satellites in test mode
CFG-SBAS-USE_RANGING	Use the SBAS satellites for navigation (ranging)
CFG-SBAS-USE_DIFFCORR	Combined enable/disable switch for fast, long-term, and ionosphere corrections
CFG-SBAS-USE_INTEGRITY	Apply integrity information data
CFG-SBAS-PRNSCANMASK	Allows selectively enabling/disabling SBAS satellites

Table 15: SBAS configuration parameters

☞ When SBAS integrity data are applied, the navigation engine stops using all signals for which no integrity data are available (including all non-GPS signals). It is not recommended to enable SBAS integrity on borders of SBAS service regions in order not to inadvertently restrict the number of available signals.

☞ SBAS integrity information is required for at least 5 GPS satellites. If this condition is not met, SBAS integrity data will not be applied.

☞ SBAS is only used if no correction services are available. If the connection stream is lost during the operation, the receiver will switch to using the SBAS corrections after the time set in CFG-NAVSPG-CONSTR_DGNSSTO (60 s by default) has elapsed.

☞ When the receiver switches from a solution using correction data to a standard position solution, the reference frame of the output position will switch from the one of the correction data to that of the standard position solution. For an SBAS solution this reference frame will be aligned within a few cm of WGS84 (and modern ITRF realizations).

3.5 QZSS SLAS

QZSS SLAS (Sub-meter Level Augmentation Service) is an augmentation technology, which provides correction data for pseudoranges of GPS, QZSS, and other major GNSS satellites. The correction stream is transmitted on the L1S signal at the L1 frequency (1575.42 MHz).

For more information on QZSS SLAS, visit the website qzss.go.jp/en/.

☞ QZSS SLAS is only used if no other correction services are available (e.g. RTCM, SPARTN, CLAS). If the connection stream is lost during the operation, the receiver will switch to use the SLAS corrections after the time set in CFG-NAVSPG-CONSTR_DGNSSTO (60 s by default) has elapsed.

3.5.1 Features

Multiple QZSS SLAS signals can be received simultaneously.

When receiving QZSS SLAS correction data, the NEO-M9L module will autonomously select the best QZSS satellite. The selection strategy is determined by the quality of the QZSS L1S signals, the receiver configuration (test mode allowed or not), and the location of the receiver with respect to the QZSS SLAS coverage area. When outside of this coverage area, the receiver will likely fall back to using SBAS corrections.

If QZSS SLAS corrections are used in the navigation solution, the differential status will be indicated in several output messages such as UBX-NAV-PVT, UBX-NAV-STATUS, UBX-NAV-SAT, NMEA-GGA, NMEA-GLL, NMEA-RMC and NMEA-GNS (see the applicable interface description [2]). The message UBX-NAV-SLAS provides detailed information about which corrections are available and applied.

Message type	Message content
0	Test mode
47	Monitoring station information
48	PRN mask
49	Data issue number
50	DGPS correction
51	Satellite health

Table 16: Supported QZSS L1S SLAS messages for navigation enhancing

3.5.2 Configuration

To enable support for the necessary QZSS L1S signal, use the CFG-SIGNAL-QZSS_L1S_ENA configuration item. To configure further QZSS SLAS functionalities, use the CFG-QZSS-USE_SLAS* configuration items.

Parameter	Description
CFG-QZSS-USE_SLAS_DGNSS	Apply QZSS SLAS corrections
CFG-QZSS-USE_SLAS_TESTMODE	Allow the correction provided by QZSS satellites that are in test mode
CFG-QZSS-USE_SLAS_RAIM_UNCORR	If this configuration is set, the receiver will try to estimate the position by using only corrected measurements; if all corrected measurements are not available, it will not use any corrections. If this configuration is not set, the receiver will mix corrected and uncorrected measurements for the navigation solution.

Table 17: QZSS SLAS configuration parameters

☞ If the RAIM option is set, other GNSS time systems than the QZSS time system cannot be observed by measurements.

3.6 Communication interfaces

u-blox receivers are equipped with a communication interface which is multi-protocol capable. The interface ports can be used to transmit GNSS measurements, monitor status information and configure the receiver.

A protocol (e.g. UBX, NMEA) can be assigned to several ports simultaneously, each configured with individual settings (e.g. baud rate, message rates, etc.). More than one protocol (e.g. UBX protocol and NMEA) can be assigned to a single port (multi-protocol capability), which is particularly useful for debugging purposes.

The NEO-M9L provides UART1, SPI, I2C and USB interfaces for communication with a host CPU. The interfaces are configured via the configuration methods described in the applicable interface description [2].

The following table shows the port numbers reported in the UBX-MON-COMMS messages.

Port no.	UBX-MON-COMMS portId	Electrical interface
1	0x0100	UART1
3	0x0300	USB

Table 18: Port number assignment

☞ It is important to isolate interface pins when VCC is removed. They can be allowed to float or they can be connected to a high impedance.

Example isolation circuit is shown below.

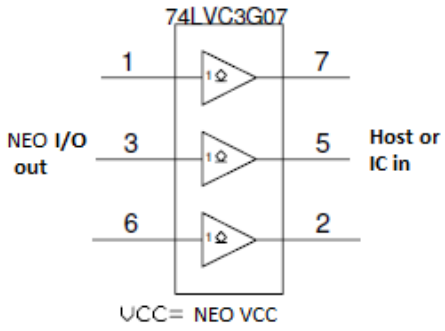


Figure 10: NEO-M9L output isolation

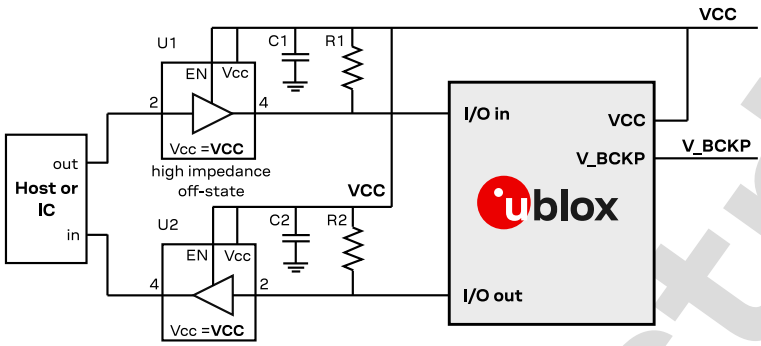


Figure 11: NEO-M9L interface isolation

3.6.1 UART

A Universal Asynchronous Receiver/Transmitter (UART) port consists of an RX and a TX line. Neither handshaking signals nor hardware flow control signals are available. The UART interface protocol and baud rate can be configured but there is no support for setting different baud rates for reception and transmission.

The NEO-M9L includes a single UART serial port. UART1 can be used as a host interface for configuration, monitoring and control.

⚠ Do not use UART2 as the only one interface to the host. Not all UBX functionality is available on UART2, such as firmware upgrade, safeboot or backup modes functionalities. No startup boot screen is sent out from UART2.

☞ The UART RX interface will be disabled when more than 100 frame errors are detected during a one-second period. This can happen if the wrong baud rate is used or the UART RX pin is grounded. An error message appears when the UART RX interface is re-enabled at the end of the one-second period.

Baud rate	Data bits	Parity	Stop bits
9600	8	none	1
19200	8	none	1
38400	8	none	1
57600	8	none	1
115200	8	none	1

Baud rate	Data bits	Parity	Stop bits
230400	8	none	1
460800	8	none	1
921600	8	none	1

Table 19: Possible UART interface configurations

The default baud rate is 38400 baud. To prevent buffering problems it is recommended not to run at a lower baud rate than the default.

Allow a short time delay of typically 100 ms between sending a baud rate change message and providing input data at the new rate. Otherwise some input characters may be ignored or the port could be disabled until the interface is able to process the new baud rate.

Note that for protocols such as NMEA or UBX, it does not make sense to change the default word length values (data bits) since these properties are defined by the protocol and not by the electrical interface.

If the amount of data configured is too much for a certain port's bandwidth (e.g. all UBX messages output on a UART port with a baud rate of 9600), the buffer will fill up. Once the buffer space is exceeded, new messages to be sent will be dropped. To prevent message loss, the baud rate and communication speed or the number of enabled messages should be carefully selected so that the expected number of bytes can be transmitted in less than one second.

3.6.2 I2C interface

An I2C interface is available for communication with an external host CPU or u-blox cellular modules. The interface can be operated in slave mode only. The I2C protocol and electrical interface are fully compatible with Fast-mode of the I2C industry standard. Since the maximum SCL clock frequency is 400 kHz, the maximum transfer rate is 400 kb/s. The SCL and SDA pins have internal pull-up resistors which should be sufficient for most applications. However, depending on the speed of the host and the load on the I2C lines additional external pull-up resistors may be necessary.

To use the I2C interface D_SEL pin must be left open.

In designs where the host uses the same I2C bus to communicate with more than one u-blox receiver, the I2C slave address for each receiver must be configured to a different value. Typically most u-blox receivers are configured to the same default I2C slave address value. To poll or set the I2C slave address, use the CFG-I2C-ADDRESS configuration item (see the applicable interface description [2]).

The CFG-I2C-ADDRESS configuration item is an 8-bit value containing the I2C slave address in 7 most significant bits, and the read/write flag in the least significant bit.

3.6.2.1 I2C register layout

The I2C interface allows 256 registers to be addressed. As shown in Figure 12, only three of these are currently implemented.

The data registers 0 to 252 at addresses 0x00 to 0xFC contain reserved information, the result from their reading is currently undefined. The data registers 0 to 252 are 1 byte wide.

At addresses 0xFD and 0xFE it is possible to read the currently available number of bytes.

The register at address 0xFF allows the data stream to be read. If there is no data awaiting transmission from the receiver, then this register delivers value 0xFF, which cannot be the first byte of a valid message. If the message data is ready for transmission, the successive reads of register 0xFF will deliver the waiting message data.

Do not use registers 0x00 to 0xFC. They are reserved for future use and they do not currently provide any meaningful data.

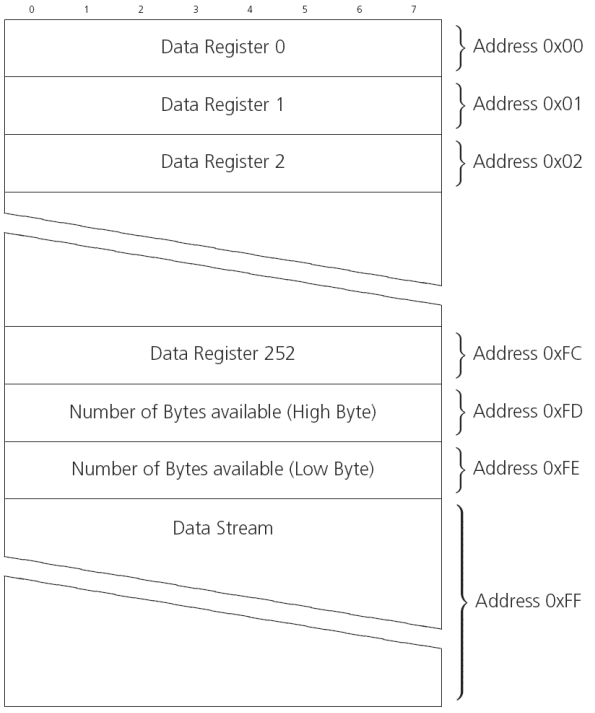


Figure 12: I2C register layout

3.6.2.2 Read access types

There are two I2C read transfer forms:

- The "random access" form: includes a slave register address and allows any register to be read.
- The "current address" form: omits the register address.

Figure 13 shows the format of the first one, the "random access" form of the request. Following the start condition from the master, the 7-bit device address and the RW bit (which is a logic low for write access) are clocked onto the bus by the master transmitter. The receiver answers with an acknowledge (logic low) to indicate that it recognizes the address.

Next, the 8-bit address of the register to be read must be written to the bus. Following the receiver's acknowledgment, the master again triggers a start condition and writes the device address, but this time the RW bit is a logic high to initiate the read access. Now, the master can read 1 to N bytes from the receiver, generating a not-acknowledge and a stop condition after the last byte being read.

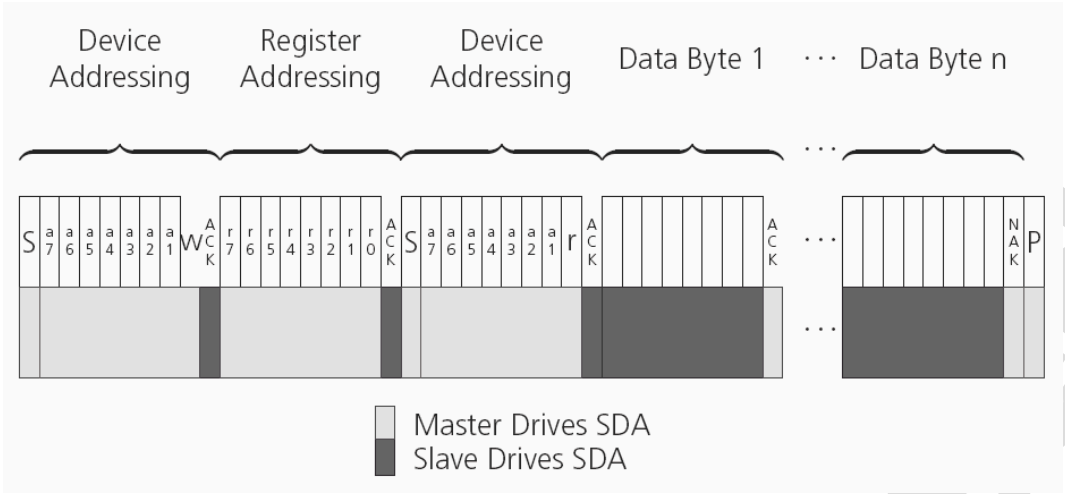


Figure 13: I2C random read access

If the second form, "current address" is used, an address pointer in the receiver is used to determine which register to read. This address pointer will increment after each read unless it is already pointing at register 0xFF, the highest addressable register, in which case it remains unaltered.

The initial value of this address pointer at startup is 0xFF, so by default all current address reads will repeatedly read register 0xFF and receive the next byte of message data (or 0xFF if no message data is waiting).

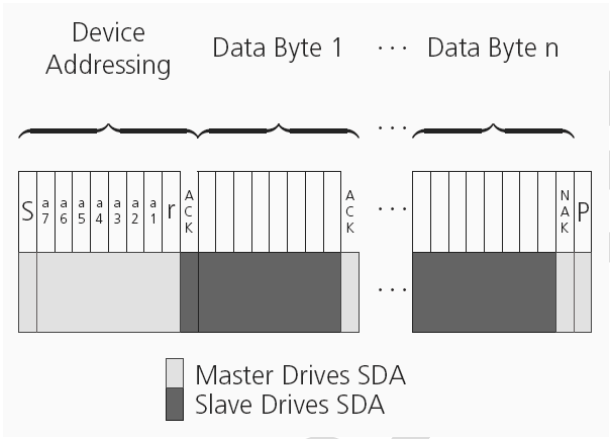


Figure 14: I2C current address read access

3.6.2.3 Write access

The receiver does not provide any write access except for writing UBX and NMEA messages to the receiver, such as configuration or aiding data. Therefore, the register set mentioned in the section Read access is not writeable.

Following the start condition from the master, the 7-bit device address and the RW bit (which is a logic low for write access) are clocked onto the bus by the master transmitter. The receiver answers with an acknowledge (logic low) to indicate that it is responsible for the given address.

The master can write 2 to N bytes to the receiver, generating a stop condition after the last byte being written. The number of data bytes must be at least 2 to properly distinguish from the write access to set the address counter in random read accesses.

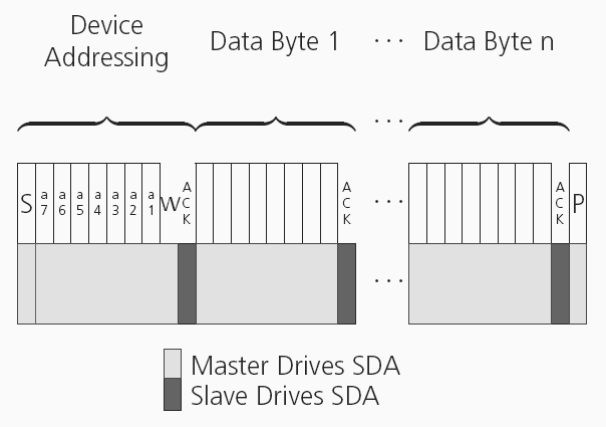


Figure 15: I2C write access

3.6.3 SPI interface

NEO-M9L has an SPI slave interface that can be selected by setting D_SEL = 0. The SPI slave interface is shared with UART1 and I2C port, the physical pins are same. The SPI pins available are:

- SPI_MISO (TXD)
- SPI_MOSI (RXD)
- SPI_CS_N
- SPI_CLK

See more information about communication interface selection from the D_SEL section.

The SPI interface is designed to allow communication to a host CPU. The interface can be operated in slave mode only.

3.6.3.1 Read access

As the register mode is not implemented for the SPI port, only the UBX/NMEA message stream is provided. This stream is accessed using the back-to-back read and write access (see section Back-to-back read and write access below). When no data is available to be written to the receiver, `MOSI` should be held logic high, i.e. all bytes written to the receiver are set to 0xFF.

To prevent the receiver from being busy parsing incoming data, the parsing process is stopped after 50 subsequent bytes containing 0xFF. The parsing process is re-enabled with the first byte not equal to 0xFF.

If the receiver has no more data to send, it sets `MISO` to logic high, i.e. all bytes transmitted decode to 0xFF. An efficient parser in the host will ignore all 0xFF bytes which are not part of a message and will resume data processing as soon as the first byte not equal to 0xFF is received.

3.6.3.2 Back-to-back read and write access

The receiver does not provide any write access except for writing UBX and NMEA messages to the receiver, such as configuration or aiding data. For every byte written to the receiver, a byte will simultaneously be read from the receiver. While the master writes to `MOSI`, at the same time it needs to read from `MISO`, as any pending data will be output by the receiver with this access. The data on `MISO` represents the results from a current address read, returning 0xFF when no more data is available.

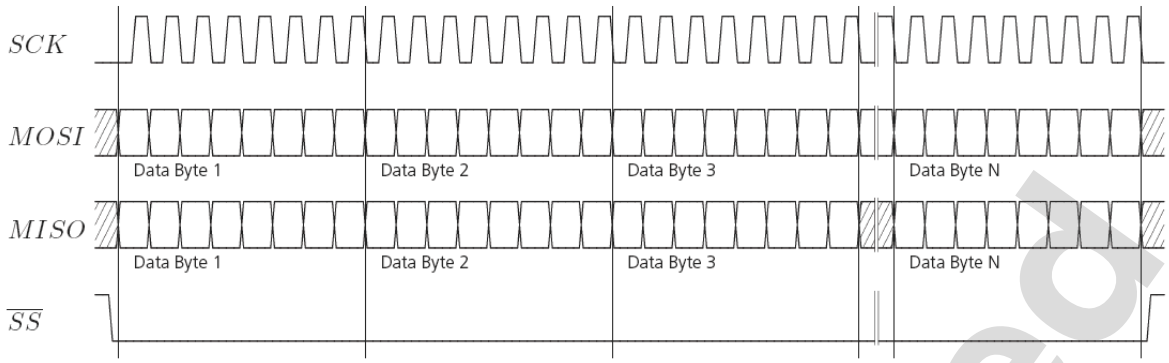


Figure 16: SPI back-to-back read/write access

3.6.4 USB interface

A single USB port is provided for host communication purposes.

The USB 2.0 FS (Full speed, 12 Mbit/s) interface can be used for host communication. Due to the hardware implementation, it may not be possible to certify the USB interface.

If the receiver executes code from internal ROM (i.e. when a valid flash firmware image is not detected), the USB behavior can differ compared to executing a firmware image from flash memory. USB host compatibility testing is thus recommended in this scenario.

The NEO-M9L receiver supports only self-powered mode operation in which the receiver is supplied from its own power supply. The V_USB pin is used to detect the availability of the USB port, i.e. whether the receiver is connected to a USB host.

☞ USB suspend mode is not supported.

☞ USB bus-powered mode is not supported.

☞ It is important to connect V_USB to ground and leave data lines open when the USB interface is not used in an application.

☞ The voltage range for V_USB is specified from 3.0 V to 3.6 V, which differs slightly from the specification for VCC.

☞ The boot screen is retransmitted on the USB port after enumeration. However, messages generated between receiver startup and USB enumeration are not visible on the USB port.

There are additional hardware requirements if USB is used:

- V_USB (pin 7) requires 1 uF capacitor mounted adjacent to the pin to ensure correct V_USB voltage detection
- The V_USB (Pin 7) voltage should be sourced from an LDO enabled by the module VCC and supplied from the USB host.
- A pull-down resistor is required on the output of this V_USB LDO
- Pin 5 is USB_DM. Pin 6 is USB_DP.
- Apply USB_DM and USB_DP series resistors; typically 27 Ω

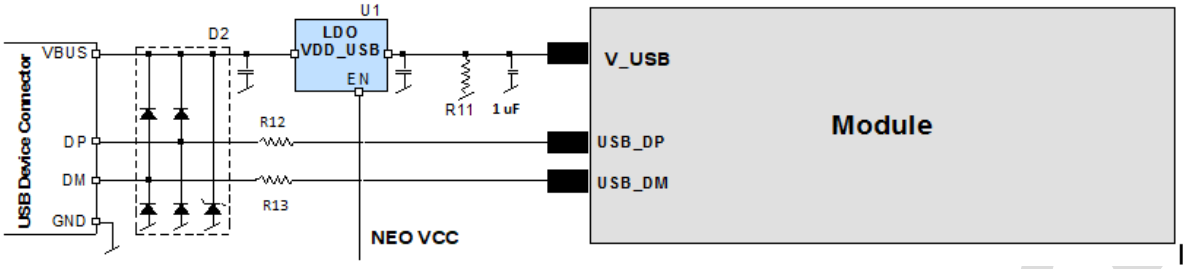


Figure 17: NEO-M9L example circuit for USB interface

R12, R13 = 27 Ω is recommended

3.7 Predefined PIOs

In addition to the communication ports, there are some predefined PIOs provided by NEO-M9L to interact with the receiver. These PIOs are described in this section.

☞ If hardware backup mode is used a proper isolation of the interfaces is needed.

3.7.1 D_SEL

The D_SEL pin can be used to configure the functionality of the combined UART1, I2C, and SPI pins. It is possible to configure the pins as UART1 + I2C, or as SPI. SPI is not available unless D_SEL pin is set to low. See Table 20 below.

Pin no.	D_SEL == 0	D_SEL == 1
20	SPI_MISO	UART1 TXD
21	SPI_MOSI	UART1 RXD
18	SPI_CS_N	I2C SDA
19	SPI_CLK	I2C SCL

Table 20: D_SEL configuration

3.7.2 RESET_N

The NEO-M9L provides the ability to reset the receiver. The RESET_N pin is an input-only pin with an internal pull-up resistor. Driving RESET_N low for at least 100 ms will trigger a cold start.

☞ The RESET_N pin will delete all information and trigger a cold start. It should only be used as a recovery option.

3.7.3 SAFEBOOT_N

The NEO-M9L provides a SAFEBOOT_N pin that is used to command the receiver safeboot mode.

If this pin is low at power up, the receiver starts in safeboot mode and GNSS operation is disabled.

The safeboot mode can be used to recover from situations where the flash content has become corrupted and needs to be restored.

In safeboot mode the receiver runs from a passive oscillator circuit with less accurate timing and hence the receiver is unable to communicate via USB.

In this mode only UART1, I2C or SPI communication is possible. For communication via UART1 in safeboot mode, the host must send a training sequence (0x55 0x55 at 9600 baud) to the receiver in order to begin communication. After this the host must wait at least 2 ms before sending any data.

Safeboot mode is used in production to program the flash and to set the low level configuration in the eFuse.

It is recommended to have the possibility to pull the SAFEBOOT_N pin low in the application. This can be provided using an externally connected test point or a host I/O port.

3.7.4 TIMEPULSE

The NEO-M9L module provides a time pulse on the TIMEPULSE pin.

More information about the time pulse feature and its configuration can be found in the Time pulse section.

3.7.5 TX_READY

This feature enables each port to define a corresponding pin, which indicates if bytes are ready to be transmitted. A listener can wait on the TX-READY signal instead of polling the I2C or SPI interfaces. The CFG-TXREADY message lets you configure the polarity and the number of bytes in the buffer before the TX-READY signal goes active. By default, this feature is disabled. For USB, this feature is configurable but might not behave as described below due to a different internal transmission mechanism. If the number of pending bytes reaches the threshold configured for this port, the corresponding pin will become active (configurable active-low or active-high), and stay active until the last bytes have been transferred from software to hardware.

☞ This is not necessarily equal to all bytes transmitted, i.e. after the pin has become inactive, up to 16 bytes might still need to be transferred to the host.

The TX_READY pin can be selected from all PIOs which are not in use (see UBX-MON-HW3 in the applicable interface description [2] for a list of the PIOs and their mapping). Each TX_READY pin is exclusively associated to one port and cannot be shared. If PIO is invalid or already in use, only the configuration for the specific TX_READY pin is ignored, the rest of the port configuration is applied if valid. The acknowledge message does not indicate if the TX-READY configuration is successfully set, it only indicates the successful configuration of the port. To validate successful configuration of the TX_READY pin, the port configuration should be read back and the settings of TX-READY feature verified (will be set to disabled/all zero if the settings are invalid).

The threshold when TX_READY is asserted should not be set above 2 kB as it is possible that the internal message buffer limit is reached before this. This results in the TX_READY pin never being set as the messages are discarded before the threshold is reached.

3.7.5.1 Extended TX timeout

If the host does not communicate over SPI or I2C for more than approximately 2 seconds, the device assumes that the host is no longer using this interface and no more packets are scheduled for this port. This mechanism can be changed by enabling "extended TX timeouts", in which case the receiver delays idling the port until the allocated and undelivered bytes for this port reach 4 kB. This feature is especially useful when using the TX-READY feature with a message output rate of less than once per second, and polling data only when data is available, determined by the TX_READY pin becoming active.

3.7.6 EXTINT

EXTINT is an external interrupt pin with fixed input voltage thresholds with respect to VCC. It can be used for functions such as accurate external frequency aiding and on/off control. Leave open if unused, this function is disabled by default.

The WT pin on NEO-M9L can be used as an EXTINT. For this feature the WT pin should be disabled, i.e. CFG-SFODO-USE_WT_PIN = 0.

3.7.7 WT and DIR inputs

NEO-M9L pin 4 (WT) is available as a wheel tick input. Pin 15 (DIR) is available as a direction input (forward/reverse indication).

By default the wheel tick count is derived from the rising edges of the WT input.

The WT input may be configured to use both the rising and falling edges of the wheel tick signal. To use both edges, the wheel tick pulses shall have approximately 1:1 mark:space ratio regardless of speed. The minimum recommended pulse width is 10 us.

For optimal performance the wheel tick resolution should be less than 5 cm. With the maximum supported wheel tick resolution is 40 cm.

The wheel tick input shall change frequency linearly with respect to the change in vehicle speed.

When the vehicle is stationary, there shall be no wheel tick pulses at the WT input. This is especially important at system shutdown and startup.

☞ Performance may be affected if there are wheel ticks on the WT pin while the vehicle is stationary.

The DIR input shall indicate whether the vehicle is moving forwards or backwards.

By default, the DIR pin polarity is automatically initialized once the vehicle has reached the required minimum speed (see Accelerated initialization and calibration procedure section). The DIR pin polarity can also be configured with the CFG-SFODO-DIR_PINPOL key.

☞ Incorrect operation may occur if WT input is used without a matching DIR input.

Alternatively, the vehicle WT (or speed) and DIR inputs can be provided via one of the communication interfaces with UBX-ESF-MEAS messages. In this use case, the WT pin can be configured as EXTINT to provide a time mark for the UBX-ESF-MEAS messages. The DIR pin can be left open.

☞ Do not exceed the maximum voltage of 3.6 V at the WT and DIR inputs.

☞ For more information, see the interface description [2].

3.7.8 Wake on motion (WOM)

NEO-M9L pin 17 (WOM) is available as the wake on motion output. By default the wake on motion feature is disabled.

3.8 Multiple GNSS assistance (MGA)

The u-blox AssistNow services provide a proprietary implementation of an A-GNSS protocol compatible with u-blox GNSS receivers.

The MGA services consist of AssistNow Online and Offline variants delivered by HTTP or HTTPS protocol. AssistNow Online optionally provides immediate satellite ephemerides, health information and time aiding data suitable for GNSS receiver systems with direct internet access.

When a client device makes an AssistNow request, the service responds with the requested data using standard UBX protocol MGA messages. These messages are ready for direct transmission from the client to the receiver port without requiring any modification.

The data downloaded from the service is organized by date and encoded into a sequence of UBX MGA messages.

Requirements	AssistNow Online	AssistNow Offline	AssistNow Autonomous
Requires external flash memory	No	Optional	Optional
Requires internet connection	Permanently	Sporadically	No
Amount of internet data	Medium	High	None
Ephemeris in data	Yes	No	No
Almanac in data	Yes	Yes	Yes

Table 21: AssistNow service overview

3.8.1 Authorization

To use the AssistNow services, customers will need to obtain an authorization token from u-blox. Go to https://www.u-blox.com/en/solution/services/assistnow or contact your local technical support to get more information and to request access to the service.

3.8.2 Preserving MGA and operational data during power-off

The time-to-fix after a receiver power interruption is dependent on the amount of operational data available at startup. Satellite broadcast information plus an estimate of accurate time can be fetched form the AssistNow service. In addition, the following techniques can restore previously stored data prior to power down.

- **Battery-backed RAM:**The receiver operational state stored in this RAM can be maintained during power outages by connecting the V_BCKP pin to an independent supply, e.g a battery. This is a recommended method as it will maintain all MGA related information plus any user configuration, calibration data and an estimate of time via the Real Time Clock. See section V_BCKP: Backup supply voltage for more information.
- **Save-on-shutdown:**The receiver can be instructed to dump its current state to flash memory as part of the shutdown procedure; this data is then automatically retrieved when the receiver is restarted. For more information, see section Save-on-shutdown feature for more info. For information on the UBX-UPD-SOS messages consult the applicable interface description [2].
- **Advanced calibration handling:**The receiver can be instructed to dump its current state (UBX-MGA-SF and UBX-NAV-PVAT) to host as part of the shutdown procedure or during the drive; this data is then sent back to the receiver when the receiver is restarted. For more information, see section Advanced calibration handling for more info.
- **Database dump:** The receiver can be made to dump the state of its navigation database in the form of a sequence of UBX messages reported to the host; these messages can be stored by the host and then sent back to the receiver when it has been restarted. See the description of the UBX-MGA-DBD messages in the applicable interface description [2] for more information.

3.9 Save-on-shutdown feature

The save-on-shutdown feature (SOS) enables the u-blox receiver to store the contents of the battery-backed RAM to an external flash memory and restore it upon startup. This allows the u-blox receiver to preserve some of the features available only with a battery backup (preserving configuration, IMU calibration, and satellite orbit knowledge) without having a battery backup supply present. It does not, however, preserve any kind of time knowledge. Save-on-shutdown must be commanded by the host. The restoring of data on startup is automatically done if the corresponding data is present in the flash. Data expiration is not checked.

The following outlines the suggested shutdown procedure when using the save-on-shutdown feature:

- With the UBX-CFG-RST message, the host commands the u-blox receiver to stop, specifying reset mode 0x08 ("Controlled GNSS stop") and a BBR mask of 0 ("Hotstart").
- The host commands the saving of the contents of BBR to the flash memory using the UBX-UPD-SOS-BACKUP message.
- For a valid request the u-blox receiver reports on the success of the backup operation with a UBX-UPD-SOS-ACK message.
- The host powers off the u-blox receiver.

The startup procedure is as follows:

- The host powers on the u-blox receiver.
- The u-blox receiver detects the previously stored data in the flash. It restores the corresponding memory and reports the success of the operation with a UBX-UPD-SOS-RESTORED message on the port on which it had received the save command message (if the output protocol filter on that port allows it). It does not report anything if no stored data has been detected.
- Additionally the u-blox receiver outputs a UBX-INF-NOTICE and/or a NMEA-TXT message with the contents `RESTORED` in the boot screen (depends on the configuration of the port and information messages) upon success.
- Optionally the host can deliver coarse time assistance using UBX-MGA-INI-TIME_UTC for better startup performance.

Once the u-blox receiver has started up it is recommended to delete the stored data using a UBX-UPD-SOS-CLEAR message. The u-blox receiver responds with a UBX-ACK-ACK / UBX-ACK-NAK message.

⚠ **CAUTION** Note that the save-on-shutdown feature works correctly only when the receiver has a valid time information when saving the SOS-BACKUP.

3.10 Clocks and time

This section introduces and explains the concepts of receiver clocks and time bases.

3.10.1 Receiver local time

The receiver is dependent on a local oscillator for both the operation of its radio parts and also for timing within its signal processing. No matter what nominal frequency the local oscillator has, u-blox receivers subdivide the oscillator signal to provide a 1-kHz reference clock signal, which is used to drive many of the receiver's processes. In particular, the measurement of satellite signals is arranged to be synchronized with the "ticking" of this 1-kHz clock signal.

When the receiver first starts, it has no information about how these clock ticks relate to other time systems; it can only count time in 1 millisecond steps. However, as the receiver derives information from the satellites it is tracking or from aiding messages, it estimates the time that each 1-kHz clock tick takes in the time base of the chosen GNSS system. This estimate of GNSS time based on the local 1-kHz clock is called receiver local time.

As receiver local time is a mapping of the local 1-kHz reference onto a GNSS time base, it may experience occasional discontinuities, especially when the receiver first starts up and the information it has about the time base is changing. Indeed, after a cold start, the receiver local time will initially indicate the length of time that the receiver has been running. However, when the

receiver obtains some credible timing information from a satellite or an aiding message, it will jump to an estimate of GNSS time.

3.10.2 Navigation epochs

Each navigation solution is triggered by the tick of the 1-kHz clock nearest to the desired navigation solution time. This tick is referred to as a **navigation epoch**. If the navigation solution attempt is successful, one of the results is an accurate measurement of time in the time base of the chosen GNSS system, called **GNSS system time**. The difference between the calculated GNSS system time and receiver local time is called **clock bias** (and **clock drift** is the rate at which this bias is changing).

In practice the receiver's local oscillator will not be as stable as the atomic clocks to which GNSS systems are referenced and consequently clock bias will tend to accumulate. However, when selecting the next navigation epoch, the receiver will always try to use the 1-kHz clock tick which it estimates to be closest to the desired fix period as measured in GNSS system time. Consequently the number of 1-kHz clock ticks between fixes will occasionally vary. This means that when producing one fix per second, there will normally be 1000 clock ticks between fixes, but sometimes, to correct drift away from GNSS system time, there will be 999 or 1001.

The GNSS system time calculated in the navigation solution is always converted to a time in both the GPS and UTC time bases for output.

Clearly when the receiver has chosen to use the GPS time base for its GNSS system time, conversion to GPS time requires no work at all, but conversion to UTC requires knowledge of the number of leap seconds since GPS time started (and other minor correction terms). The relevant GPS-to-UTC conversion parameters are transmitted periodically (every 12.5 minutes) by GPS satellites, but can also be supplied to the receiver via the UBX-MGA-GPS-UTC aiding message. By contrast when the receiver has chosen to use the GLONASS time base as its GNSS system time, conversion to GPS time is more difficult as it requires knowledge of the difference between the two time bases, but as GLONASS time is closely linked to UTC, conversion to UTC is easier.

When insufficient information is available for the receiver to perform any of these time base conversions precisely, predefined default offsets are used. Consequently plausible times are nearly always generated, but they may be wrong by a few seconds (especially shortly after receiver start). Depending on the configuration of the receiver, such "invalid" times may well be output, but with flags indicating their state (e.g. the "valid" flags in UBX-NAV-PVT).

☞ u-blox receivers employ multiple GNSS system times and/or receiver local times (in order to support multiple GNSS systems concurrently), so users should not use UBX messages reporting GNSS system time or receiver local time. It is recommended to use messages that report UTC time and other messages are retained only for backwards compatibility reasons.

3.10.3 iTOW timestamps

All the main UBX-NAV messages (and some other messages) contain an **iTOW** field which indicates the GPS time at which the navigation epoch occurred. Messages with the same iTOW value can be assumed to have come from the same navigation solution.

☞ With Priority navigation mode enabled, iTOW can be used as an epoch collector only for messages of the same group (e.g. priority navigation messages with the same iTOW value can be assumed to have come from the same navigation solution).

Note that iTOW values may not be valid (i.e. they may have been generated with insufficient conversion data) and therefore it is not recommended to use the iTOW field for any other purpose.

☞ The original designers of GPS chose to express time/date as an integer week number (starting with the first full week in January 1980) and a time of week (often abbreviated

to TOW) expressed in seconds. Manipulating time/date in this form is far easier for digital systems than the more conventional year/month/day, hour/minute/second representation. Consequently, most GNSS receivers use this representation internally, only converting to a more conventional form at external interfaces. The iTOW field is the most obvious externally visible consequence of this internal representation.

If reliable absolute time information is required, users are recommended to use the UBX-NAV-PVT navigation solution message which also contains additional fields that indicate the validity (and accuracy in UBX-NAV-PVT) of the calculated times (see also the GNSS times section below for further messages containing time information).

3.10.4 GNSS times

Each GNSS has its own time reference for which detailed and reliable information is provided in the messages listed in the table below.

Time reference	Message
GPS time	UBX-NAV-TIMEGPS
BeiDou time	UBX-NAV-TIMEBDS
GLONASS time	UBX-NAV-TIMEGLO
Galileo time	UBX-NAV-TIMEGAL
QZSS time	UBX-NAV-TIMEQZSS
UTC time	UBX-NAV-TIMEUTC

Table 22: GNSS times

3.10.5 Time validity

Information about the validity of the time solution is given in the following form:

- Time validity: Information about time validity is provided in the `valid` flags (e.g. `validDate` and `validTime` flags in the UBX-NAV-PVT message). If these flags are set, the time is known and considered valid for use. These flags are shown in table GNSS times in section GNSS times above as well as in the UBX-NAV-PVT message.
- Time validity confirmation: Information about confirmed validity is provided in the `confirmedDate` and `confirmedTime` flags in the UBX-NAV-PVT message. If these flags are set, the time validity can be confirmed by using an additional independent source, meaning that the probability of the time to be correct is very high. Note that information about time validity confirmation is only available if the `confirmedAvai` bit in the UBX-NAV-PVT message is set.

☞ `validDate` means that the receiver has knowledge of the current date. However, it must be noted that this date might be wrong for various reasons. Only when the `confirmedDate` flag is set, the probability of the incorrect date information drops significantly.

☞ `validTime` means that the receiver has knowledge of the current time. However, it must be noted that this time might be wrong for various reasons. Only when the `confirmedTime` flag is set, the probability of incorrect time information drops significantly.

☞ `fullyResolved` means that the UTC time is known without full seconds ambiguity. When deriving UTC time from GNSS time the number of leap seconds must be known, with the exception of GLONASS. It might take several minutes to obtain such information from the

GNSS payload. When the one second ambiguity has not been resolved, the time accuracy is usually in the range of ~20s.

3.10.6 UTC representation

UTC time is used in many NMEA and UBX messages. In NMEA messages it is always reported rounded to the nearest hundredth of a second. Consequently, it is normally reported with two decimal places (e.g. 124923.52). Although compatibility mode (selected using CFG-NMEA-COMPAT) requires three decimal places, rounding to the nearest hundredth of a second remains, so the extra digit is always 0.

UTC time is also reported within some UBX messages, such as UBX-NAV-TIMEUTC and UBX-NAV-PVT. In these messages date and time are separated into seven distinct integer fields. Six of these (year, month, day, hour, min and sec) have fairly obvious meanings and are all guaranteed to match the corresponding values in NMEA messages generated by the same navigation epoch. This facilitates simple synchronization between associated UBX and NMEA messages.

The seventh field is called nano and it contains the number of nanoseconds by which the rest of the time and date fields need to be corrected to get the precise time. So, for example, the UTC time 12:49:23.521 would be reported as: hour: 12, min: 49, sec: 23, nano: 521000000.

It is however important to note that the first six fields are the result of rounding to the nearest hundredth of a second. Consequently the nano value can range from -5000000 (i.e. -5 ms) to +994999999 (i.e. nearly 995 ms).

When the nano field is negative, the number of seconds (and maybe minutes, hours, days, months or even years) will have been rounded up. Therefore, some or all of them must be adjusted in order to get the correct time and date. Thus in an extreme example, the UTC time 23:59:59.9993 on 31st December 2011 would be reported as: year: 2012, month: 1, day: 1, hour: 0, min: 0, sec: 0, nano: -700000.

Of course, if a resolution of one hundredth of a second is adequate, negative nano values can simply be rounded up to 0 and effectively ignored.

Which master clock the UTC time is referenced to is output in the message UBX-NAV-TIMEUTC.

The preferred variant of UTC time can be specified using CFG-NAVSPG-UTCSTANDARD configuration item.

3.10.7 Leap seconds

Occasionally it is decided (by one of the international time keeping bodies) that, due to the slightly uneven spin rate of the Earth, UTC has moved sufficiently out of alignment with mean solar time (i.e. the Sun no longer appears directly overhead at 0 longitude at midday). A "leap second" is therefore announced to bring UTC back into close alignment. This normally involves adding an extra second to the last minute of the year, but it can also happen on 30th June. When this happens UTC clocks are expected to go from 23:59:59 to 23:59:60 and only then on to 00:00:00.

It is also theoretically possible to have a negative leap second, in which case there will only be 59 seconds in a minute and 23:59:58 will be followed by 00:00:00.

u-blox receivers are designed to handle leap seconds in their UTC output and consequently users processing UTC times from either NMEA or UBX messages should be prepared to handle minutes that are either 59 or 61 seconds long.

Leap second information can be polled from the u-blox receiver with the message UBX-NAV-TIMELS.

3.10.8 Real-time clock

u-blox receivers contain circuitry to support a real-time clock, which (if correctly fitted and powered) keeps time while the receiver is otherwise powered off. When the receiver powers up, it attempts to use the real-time clock to initialize receiver local time and in most cases this leads to appreciably faster first fixes.

3.10.9 Date

All GNSS frequently transmit information about the current time within their data message. In most cases, this is a time of week (often abbreviated to TOW), which indicates the elapsed number of seconds since the start of the week (midnight Saturday/Sunday). In order to map this to a full date, it is necessary to know the week and so the GNSS also transmit a week number, typically every 30 seconds. Unfortunately the GPS L1C/A data message was designed in a way that only allows the bottom 10 bits of the week number to be transmitted. This is not sufficient to yield a completely unambiguous date as every 1024 weeks (a bit less than 20 years), the transmitted week number value "rolls over" back to zero. Consequently, GPS L1 receivers cannot tell the difference between, for example, 1980, 1999 or 2019 etc.

Fortunately, although BeiDou and Galileo have similar representations of time, they transmit sufficient bits for the week number to be unambiguous for the foreseeable future (the first ambiguity will be in 2078 for Galileo and not until 2163 for BeiDou). GLONASS has a different structure, based on a time of day, but again transmits sufficient information to avoid any ambiguity during the expected lifetime of the system (the first ambiguous date will be in 2124). Therefore, u-blox 9 receivers using Protocol Version 24 and above regard the date information transmitted by GLONASS, BeiDou and Galileo to be unambiguous and, where necessary, use this to resolve any ambiguity in the GPS date.

☞ Customers attaching u-blox receivers to simulators should be aware that GPS time is referenced to 6th January 1980, GLONASS to 1st January 1996, Galileo to 22nd August 1999 and BeiDou to 1st January 2006; the receiver cannot be expected to work reliably with signals simulated before these dates.

3.10.9.1 GPS-only date resolution

In circumstances where only GPS L1C/A signals are available and for receivers with earlier firmware versions, the receiver establishes the date by assuming that all week numbers must be at least as large as a reference rollover week number. This reference rollover week number is hard-coded at compile time and is normally set a few weeks before the software is completed, but it can be overridden by CFG-NAVSPG-WKNROLLOVER configuration item to any value the user wishes.

The following example illustrates how this works: Assume that the reference rollover week number set in the firmware at compile time is 1524 (which corresponds to a week in calendar year 2009, but would be transmitted by the satellites as 500). In this case, if the receiver sees transmissions containing week numbers in the range of 500 ... 1023, these will be interpreted as week numbers 1524 ... 2047 (calendar year 2009 ... 2019), whereas transmissions with week numbers from 0 to 499 are interpreted as week numbers 2048 ... 2547 (calendar year 2019 ... 2028).

☞ It is important to set the reference rollover week number appropriately when supplying u-blox receivers with simulated signals, especially when the scenarios are in the past.

3.10.10 Time pulse

3.10.10.1 Introduction

u-blox receivers include a time pulse function providing clock pulses with configurable duration and frequency. The time pulse function can be configured using the CFG-TP-* configuration group. The UBX-TIM-TP message provides time information for the next pulse and the time source.

☞ For timepulse1 (TP1) NEO-M9L the WT pin should be disabled, i.e. CFG-SFODO-USE_WT_PIN = 0. Afterward, the time pulse (TP1) is available at the DIR pin of the NEO-M9L module.

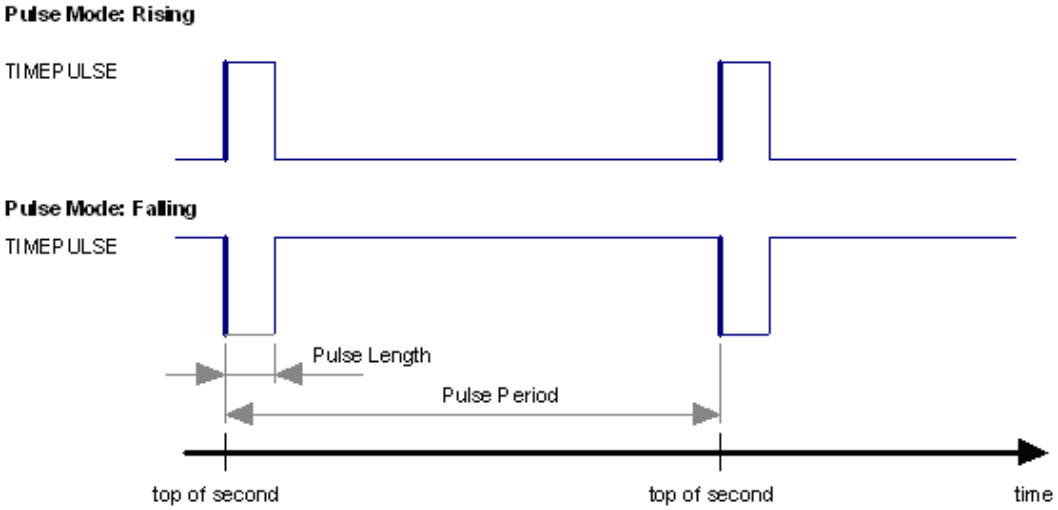


Figure 18: Time pulse

3.10.10.2 Recommendations

- The time pulse can be aligned to a wide variety of GNSS times or to variants of UTC derived from them (see the section on time bases). However, it is strongly recommended that the choice of time base is aligned with the available GNSS signals (so to produce GPS time or UTC(USNO), ensure GPS signals are available, and for GLONASS time or UTC(SU) ensure the presence GLONASS signals). This will involve coordinating the setting of CFG-SIGNAL-* configuration group with the choice of time pulse time base.
- When using time pulse for timing applications requiring absolute time accuracy, e.g. with requirements specifying offset to UTC, it is recommended to calibrate the user's full setup for TP output against a reference timing source. To achieve best absolute and consistent accuracy (e.g. for mass deployment), it is recommended that the user should calibrate each single setup and calibrate under different GNSS modes and different temperatures which are applicable to the user's application and operating requirements. The user should take the calibrated values and configure the compensation accordingly (see the section on Time pulse configuration)
- To get the best timing accuracy with the antenna, a fixed and *accurate* position is needed.
- If relative time accuracy between multiple receivers is required, do not mix receivers of different product families. If this is required, the receivers must be calibrated accordingly, by setting cable delay and user delay.
- The recommended configuration when using the UBX-TIM-TP message is to set both the measurement rate (CFG-RATE-MEAS) and the time pulse frequency (CFG-TP-*) to 1 Hz.

☞ Since the rate of UBX-TIM-TP is bound to 1 Hz, more than one UBX-TIM-TP message can appear between two pulses if the time pulse frequency is set larger than 1 Hz. In this case all UBX-TIM-TP messages in between time pulses T1 and T2 belong to T2 and the last UBX-TIM-TP before T2 reports the most accurate quantization error. In general, if the time pulse rate is not configured to 1 Hz, there will not be a single UBX-TIM-TP message for each time pulse.

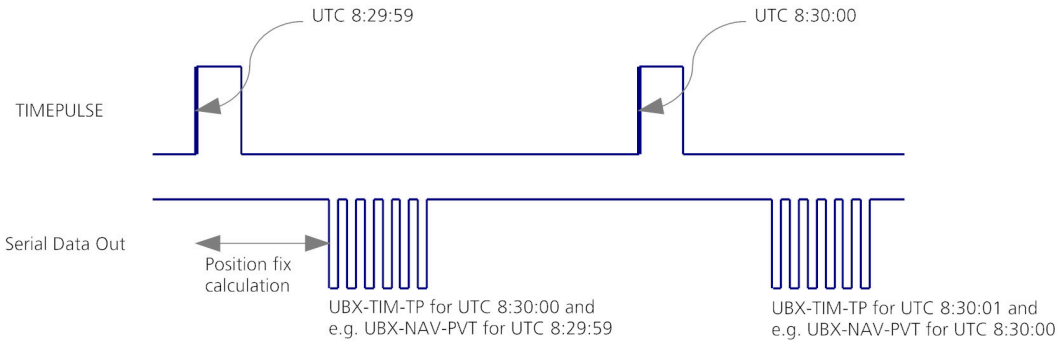


Figure 19: Time pulse and TIM-TP

3.10.10.3 GNSS time bases

GNSS receivers must handle a variety of different time bases as each GNSS has its own reference system time. What is more, although each GNSS provides a model for converting their system time into UTC, they all support a slightly different variant of UTC. So, for example, GPS supports a variant of UTC as defined by the US National Observatory, while BeiDou uses UTC from the National Time Service Center, China (NTSC). While the different UTC variants are normally closely aligned, they can differ by as much as a few hundreds of nanoseconds.

Although u-blox receivers can combine a variety of different GNSS times internally, the user must choose a single type of GNSS time and, separately, a single type of UTC for input (on EXTINT pins) and output (via the TIMEPULSE pin) and the parameters reported in corresponding messages.

The CFG-TP-TIMEGRID_TP* configuration item allows the user to choose between any of the supported GNSS (GPS, GLONASS, BeiDou, etc.) time bases and UTC. Also, the CFG-NAVSPG-UTCSTANDARD configuration item allows the user to select which variant of UTC the receiver should use. This includes an "automatic" option which causes the receiver to select an appropriate UTC version itself, based on the GNSS configuration, using, in order of preference, USNO if GPS is enabled, SU if GLONASS is enabled, NTSC if BeiDou is enabled and, finally, European if Galileo is enabled.

The receiver will assume that an input time pulse uses the same GNSS time base as specified for the time pulse output. So if the user selects GLONASS time for time pulse output, any time pulse input must also be aligned to GLONASS time (or to the separately chosen variant of UTC). Where UTC is selected for time pulse output, any GNSS time pulse input will be assumed to be aligned to GPS time.

☞ u-blox receivers allow users to independently choose GNSS signals used in the receiver (using CFG-SIGNAL-*) and the input/output time base (using CFG-TP-*). For example it is possible to instruct the receiver to use GPS and GLONASS satellite signals to generate BeiDou time. This practice will compromise time pulse accuracy if the receiver cannot measure the timing difference between the constellations directly and is therefore not recommended.

☞ The information that allows GNSS times to be converted to the associated UTC times is only transmitted by the GNSS at relatively infrequent periods. For example GPS transmits UTC(USNO) information only once every 12.5 minutes. Therefore, if a time pulse is configured to use a variant of UTC time, after a cold start, substantial delays before the receiver has sufficient information to start outputting the time pulse can be expected.

3.10.10.4 Time pulse configuration

u-blox NEO-M9L receivers provide a time pulse (TIMEPULSE) signal with a configurable pulse period, length and polarity (rising or falling edge).

It is possible to define different signal behavior (i.e. output frequency and pulse length) depending on whether or not the receiver is locked to a reliable time source. Time pulse signal can be configured using the configuration group CFG-TP-*.

3.10.10.5 Configuring time pulse with CFG-TP-*

The configuration group CFG-TP-* can be used to change the time pulse settings, and includes the following parameters defining the pulse:

- **timepulse enable** - If this item is set, the time pulse is active.
- **frequency/period type** - Determines whether the time pulse is interpreted as frequency or period.
- **length/ratio type** - Determines whether the time pulse length is interpreted as length [us] or pulse ratio [%].
- **antenna cable delay** - Signal delay owning to RF components (e.g. antenna, cable, and splitter etc.) before the receiver input. This delay parameter affects the receiver calculation of GNSS time and it is used to compensate for the signal transit time prior to the receiver and any uncompensated delay from the receiver; a positive value compensates this delay, i.e. advances the time pulse
- **pulse frequency/period** - Frequency or pulse time period when locked mode is not configured or active.
- **pulse frequency/period lock** - Frequency or pulse time period, as soon as the receiver has calculated a valid time from a received signal. Only used if the corresponding item is set to use another setting in locked mode.
- **pulse length/ratio** - Length or duty cycle of the generated pulse, either specifies a time or ratio for the pulse to be on/off.
- **pulse length/ratio lock** - Length or duty cycle of the generated pulse, as soon as the receiver has calculated a valid time from a received signal. Only used if the corresponding item is set to use another setting in locked mode.
- **user delay** - A time offset of the TP output for adjustment in a user application. It adjusts the time pulse position only with respect to GNSS time. Configuring a positive value will add a delay, i.e. retard the pulse with respect to GNSS time. Conversely, a negative value will advance the pulse. This configuration is available for all supported TP outputs.
- **lock to GNSS freq** - If this item is set, uses the frequency gained from the GNSS signal information rather than the local oscillator's frequency.
- **locked other setting** - If this item is set, the alternative setting will be used as soon as the receiver can calculate a valid time. This mode can be used, for example, to disable time pulse if the time is not locked, or to indicate a lock with different duty cycles.
- **align to TOW** - If this item is set, pulses are aligned to the top of a second.
- **polarity** - If set, the first edge of the pulse is a rising edge (pulse polarity: rising).
- **grid UTC/GNSS** - Selection between UTC (0), GPS (1), GLONASS (2), BeiDou (3) and (4) Galileo timegrid. Also affects the time output by UBX-TIM-TP message.

☞ The maximum pulse length cannot exceed the pulse period.

☞ Time pulse settings shall be chosen in such a way that neither the high nor the low period of the output is less than 50 ns (except when disabling it completely), otherwise pulses can be lost.

3.10.10.5.1 Example

The example below shows the 1PPS TIMEPULSE signal generated on the time pulse output according to the specific parameters of the CFG-TP-* configuration group:

- **CFG-TP-TP1_ENA** = 1
- **CFG-TP-PERIOD_TP1** = 1 000 000 µs

- **CFG-TP-LEN_TP1** = 100 000 µs
- **CFG-TP-TIMEGRID_TP1** = 1 (GPS)
- **CFG-TP-PULSE_LENGTH_DEF** = 0 (Period)
- **CFG-TP-ALIGN_TO_TOW_TP1** = 1
- **CFG-TP-USE_LOCKED_TP1** = 1
- **CFG-TP-POL_TP1** = 1
- **CFG-TP-PERIOD_LOCK_TP1** = 100 000 µs
- **CFG-TP-LEN_LOCK_TP1** = 100 000 µs

The 1 Hz output is maintained whether or not the receiver is locked to GPS time. The alignment to TOW can only be maintained when GPS time is locked.

The TP1 is available at the DIR pin of the NEO-M9L module, with the WT pin disabled.

To configure the time pulse (TP2), signal output which is available at the time pulse pin of the NEO-M9L module, the following parameters of the CFG-TP-* configuration group need to changed:

- **CFG-TP-TP2_ENA** = 1
- **CFG-TP-PERIOD_TP2** = 1 000 000 µs
- **CFG-TP-LEN_TP2** = 100 000 µs
- **CFG-TP-TIMEGRID_TP2** = 1 (GPS)
- **CFG-TP-PULSE_LENGTH_DEF** = 0 (Period)
- **CFG-TP-ALIGN_TO_TOW_TP2** = 1
- **CFG-TP-USE_LOCKED_TP2** = 1
- **CFG-TP-POL_TP2** = 1
- **CFG-TP-PERIOD_LOCK_TP2** = 100 000 µs
- **CFG-TP-LEN_LOCK_TP2** = 100 000 µs

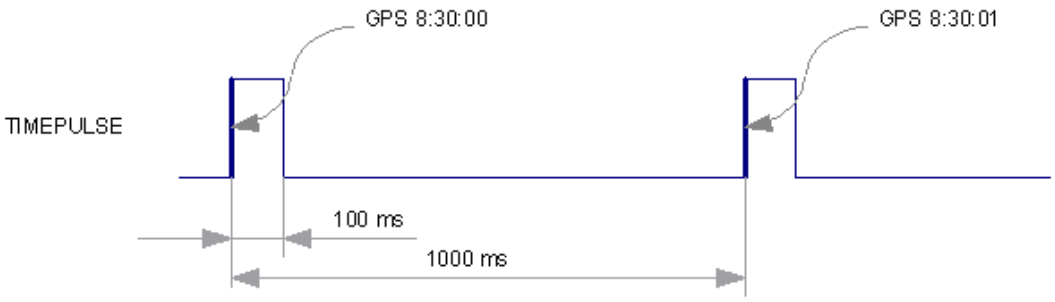


Figure 20: Time pulse signal with the example parameters

3.10.11 Time mark

The receiver can be used to provide an accurate measurement of the time at which a pulse was detected on the external interrupt pin. The reference time can be chosen by setting the time source parameter to UTC, GPS, GLONASS, BeiDou, Galileo or local time in the CFG-TP-* configuration group. The UTC standard can be set in the CFG-NAVSPG-* configuration group. The delay figures defined with CFG-TP-* are also applied to the results output in the UBX-TIM-TM2 message.

A UBX-TIM-TM2 message is output at the next epoch if

- The UBX-TIM-TM2 message is enabled, and
- A rising or falling edge was triggered since last epoch on one of the EXTINT channels.

The UBX-TIM-TM2 messages includes the time of the last time mark, new rising/falling edge indicator, time source, validity, number of marks and an accuracy estimate.

☞ Only the last rising and falling edge detected between two epochs is reported since the output rate of the UBX-TIM-TM2 message corresponds to the measurement rate configured with CFG-RATE-MEAS (see Figure 21 below).

☞ The WT pin on NEO-M9L can be used as an EXTINT. For this feature the WT pin should be disabled, i.e. CFG-SFODO-USE_WT_PIN = 0. Secondly, the CFG-TP-TP1_ENA needs to be enabled.

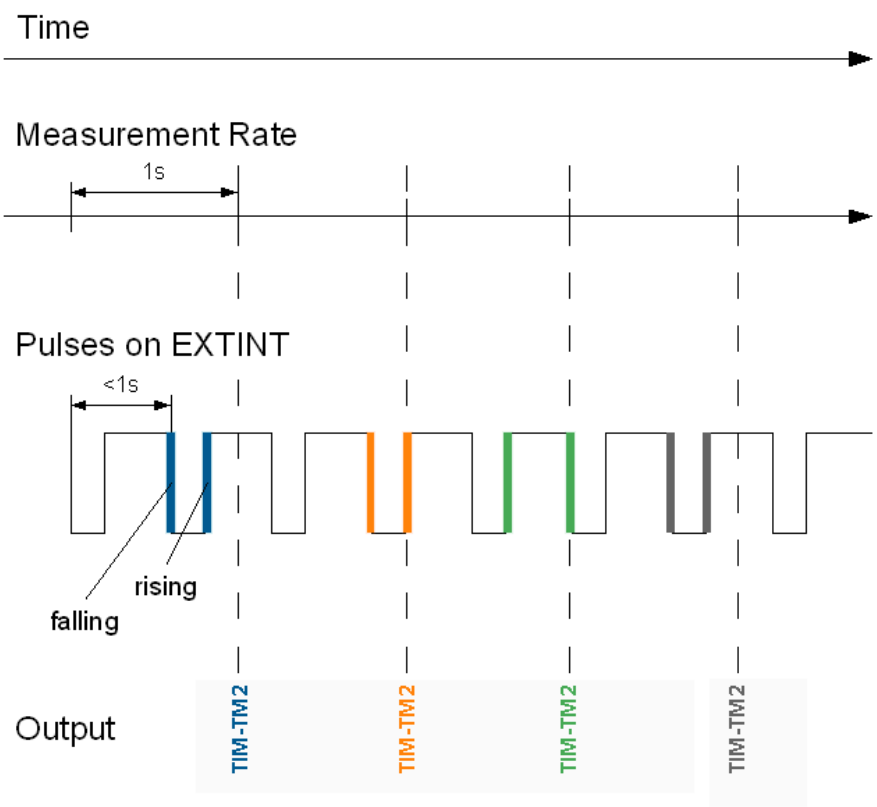


Figure 21: Time mark

3.11 Security

The security concept of NEO-M9L covers the air interface between the receiver and the GNSS satellites, the integrity of the receiver itself and the interface to the host system.

There are functions to monitor/detect certain security threats and report it to the host system. Other functions try to mitigate the threat and allow the receiver to operate normally.

The table below gives an overview about possible threats and which functionality is available to detect and/or mitigate it.

Threat	u-blox solution
Over air signal integrity	Spoofing detection / monitoring
	Jamming/interference detection / monitoring
GNSS receiver integrity	Secure boot
	Secure firmware update

Threat	u-blox solution
	Receiver configuration lock
Secure interface to host	Message authentication

Table 23: u-blox security options

3.11.1 Spoofing detection / monitoring

Spoofing is the process where a counterfeit GNSS signal is transmitted locally to prevent the true signal from being used and so produce an erroneous position fix and/or time solution.

The spoofing detection algorithm monitors multiple observed signal parameters for suspicious changes to identify external manipulation. A flag in UBX-NAV-STATUS message (flags2 - `spoofDetState`) alerts the user to potential spoofing activity.

A detection is successful when a signal is observed to transition from an initially genuine one to a spoofed version. Hence detection is not possible if the receiver is started under spoofing conditions. The detection algorithms rely on availability of signals from multiple GNSS constellations - the detection algorithm does not work in single-GNSS mode.

3.11.2 Jamming/interference detection / monitoring

The field `jamInd` of the UBX-MON-RF message can be used as an indicator for continuous wave (narrow-band) jammers/interference only. The interpretation of the value depends on the application. It is necessary to run the receiver in an unjammed environment to determine an appropriate threshold for the unjammed case. If the value rises significantly above this threshold, this indicates that a continuous wave jammer is present.

This monitoring function is always enabled.

The indicator reports any currently detected narrow-band interference over all currently configured signal bands.

3.11.2.1 Jamming/interference monitor (ITFM) / broadband interference monitoring

The field `flags` of the UBX-MON-RF message can be used as an indicator for both broadband and continuous wave (CW) jammers/interference. It is independent of the (CW only) jamming indicator described in Jamming/interference indicator above.

This monitor reports whether jamming has been detected or suspected by the receiver. The receiver monitors the background noise and looks for significant changes. Normally, with no interference detected, it will report "OK". If the receiver detects that the noise has risen above a preset threshold, the receiver reports "Warning". If in addition, there is no current valid fix, the receiver reports "Critical".

The monitor has four states as shown in the following table:

Value	Reported state	Description
0	Unknown	Jamming/interference monitor not enabled, uninitialized or antenna disconnected
1	OK	no interference detected
2	Warning	position OK but interference is visible (above the thresholds)
3	Critical	no reliable position fix and interference is visible (above the thresholds); interference is probable reason why there is no fix

Table 24: Jamming/interference monitor reported states

The monitor is disabled by default. The monitor is enabled by setting the CFG-ITFM-ENABLE configuration item. In this message it is also possible to specify the thresholds at which broadband and CW jamming are reported. These thresholds should be interpreted as the dB level above "normal". It is also possible to specify whether the receiver expects an active or a passive antenna.

☞ The monitoring algorithm relies on comparing the currently measured spectrum with a reference from when a good fix was obtained. Thus the monitor will only function when the receiver has had at least one (good) first fix, and will report "Unknown" before this time.

The monitor reports any currently detected interference over all currently configured signal bands.

3.11.3 GNSS receiver integrity

3.11.3.1 Secure boot

The NEO-M9L boots only with firmware images that are signed by u-blox. This prevents the execution of non-genuine firmware images run on the receiver.

3.11.3.2 Secure firmware update

The firmware image itself is encrypted and signed by u-blox. The NEO-M9L verifies the signature at each start.

3.11.3.3 Receiver configuration lock

The receiver configuration lock feature ensures that no configuration changes are possible once the feature is enabled. The configuration lock is enabled by setting the configuration item CFG-SEC-CFG_LOCK to "true".

The configuration lock can be applied to different configuration layers including the RAM, BBR, and flash memory. At startup, the receiver constructs the configuration database from different configuration layers and maintains it in the run-time RAM memory. When the configuration lock is set in the run-time RAM, the receiver configuration cannot be changed on any configuration layer.

☞ For more information on the configuration layers including the order of priority they are applied in, see the applicable interface description [2].

The configuration lock set on a configuration layer in volatile memory (RAM, BBR) is removed when the memory is cleared. However, the configuration lock set in non-volatile memory (flash memory) is permanent apart from one exception: during firmware upload to flash memory, the flash is erased during the process causing the configuration lock to be cleared. Refer to Firmware update to flash for more information on firmware update.

The configuration lock can also be applied to the OTP layer. The configuration lock on OTP layer is permanent and cannot be reversed. However, patches can still be applied to the OTP memory.

To test the lock functionality, set it on the RAM configuration layer. After a power cycle, the information on RAM layer is cleared and the lock is no longer set.

☞ It is recommended to apply the configuration lock on the same layer the configuration is stored.

An example of use case is that the host application locks the receiver configuration. A user communicating with the NEO-M9L through any of the available interfaces can poll, enable or send messages, but cannot change the configuration by sending UBX configuration messages.

3.11.4 Receiver message authentication

3.11.4.1 Introduction

The NEO-M9L features a security mechanism to sign its output and provide a way to authenticate the receiver-to-host communication channel.

This mechanism can be used to safeguard against man-in-the-middle attacks by third parties. The asymmetric signature mechanism, which is based on a pair of private and public keys, can be utilized by a host to verify that all output by NEO-M9L is originated by the intended NEO-M9L and not by a third party.

☞ This mechanism cannot be used to authenticate the host-to-receiver communication channel.

⚠ This mechanism does not guarantee protection against third-party attackers with physical access to the NEO-M9L.

3.11.4.2 Asymmetric signature: private and public keys

Asymmetric signing relies on a pair of private and public keys. The NEO-M9L asymmetric signing is based on the Elliptic Curve Digital Signature Algorithm (ECDSA). The ECDSA offers a variant of the Digital Signature Algorithm (DSA) which uses elliptic curve cryptography. The NEO-M9L uses the secp192r1 curve for signing its output.

Before configuring NEO-M9L to sign its output, it is required to generate a pair of private and public keys. The private and public keys must be generated using the ECDSA secp192r1 curve. NEO-M9L requires that the private key (d) is a 24-byte long value and that the public key is a 48-byte long value. The 48-byte value is commonly split into a pair of 24-byte long values (Px and Py).

In later steps, the private key (d) will be permanently written in the eFuse of the NEO-M9L and the public key (Px and Py) will be used by the host to verify the message stream sent by the NEO-M9L.

⚠ It is advisable not to save the private key on the host in any accessible form, otherwise a potential leaking of the private key can endanger the security of the receiver-to-host communication channel.

3.11.4.3 Writing the private key in the eFuse

The generated 24-byte long private key (d) must be permanently written in the eFuse of the NEO-M9L. To write the private key in the eFuse, the eFuse file 0xa6 (host interface signature config) must be written. This is done by the UBX-CFG-OTP message for file 0xa6. For the UBX-CFG-OTP message specification, see the interface description [2].

⚠ Writing a file for the private key in the eFuse is a one-time write-only irreversible operation. It is not possible to delete a private key file.

☞ For security reasons, it is not possible to read the private key from the NEO-M9L after it has been written in the eFuse.

The binary string for the desired UBX-CFG-OTP message can be generated either by following the message specification (for eFuse file 0xa6) or by using the UBX-CFG-OTP view in u-center (for u-blox 9, eFuse file 0xa6). Both options are outlined in the example later in this section. See [Example].

Once the UBX-CFG-OTP binary string is available, send it to the NEO-M9L. There should be a UBX-ACK-ACK for class 0x06 and ID 0x41 to confirm successful writing of the file. Further confirmation can be obtained by polling the UBX-CGF-OTP message in u-center (in UBX-CFG-OTP, then OTP Details), where the new file for the private key should now be listed as 'Asym. host i/f signing private key'. If a file for a private key already exists, then a new one cannot be written again. Any attempt to write another file for the private key will be rejected and result in a UBX-ACK-NAK.

3.11.4.4 Configuring the session ID

The session ID is a 24-byte long value that can be used to provide an extra level of security on the receiver-to-host communication channel. As will be described later, the session ID that is configured on a NEO-M9L is part of the data input used to calculate the resulting signature. As such, configuring the session ID can be used for enabling a mechanism of message short-term validity. In other words, the host can rotate the session ID as required to provide an extra layer of protection against malicious third parties.

The host can configure the session ID multiple times. It is configured via the CFG-SEC-ECCFGSESSIONID0, CFG-SEC-ECCFGSESSIONID1 and CFG-SEC-ECCFGSESSIONID2 configuration items, preferably at the start of a new communication session with the NEO-M9L. If the session ID is not configured, the session ID will be 0x0. Configuring the session ID is outlined in the example later in this section. See [Example].

☞ It is advisable to always configure the session ID to a value different from the default. For improved security, it is highly recommended to reconfigure the session ID frequently and using randomly-generated values.

3.11.4.5 Enabling signature output

The final step to fully use the asymmetric signature output on the NEO-M9L is to enable the UBX-SEC-ECSIGN message on the desired ports. This can be done per port via the CFG-MSGOUT-UBX_SEC_ECSIGN_* configuration items. The UBX-SEC-ECSIGN message contains all information related to the signature calculated by the NEO-M9L for the given configuration that has been done in the previous steps. For the UBX-SEC-ECSIGN message specification and the CFG-MSGOUT-UBX_SEC_ECSIGN_* configuration items, see the interface description [2].

Once configured, the NEO-M9L outputs the regular expected output plus the configured UBX-SEC-ECSIGN messages. For example, the output stream will look as follows:

- UBX-SEC-ECSIGN, Message1, Message2, ..., MessageN, UBX-SEC-ECSIGN, Message1, Message2, ..., MessageM, UBX-SEC-ECSIGN

UBX-SEC-ECSIGN is output according to its configured rate. For example, if CFG-MSGOUT-UBX_SEC_ECSIGN_UART1 is configured to 2, the message will be output every 2 seconds on UART1.

☞ The output rate of UBX-SEC-ECSIGN cannot be less than 1. In other words, the output cannot be signed more frequently than every 1 second. For performance considerations, depending on the output load of the used port, it is recommended to adapt the output rate of UBX-SEC-ECSIGN accordingly to reduce the frequency of signing.

All regular output on a port since the last UBX-SEC-ECSIGN is combined internally into a single chunk. This chunk is hashed with SHA-256. Then a signature is calculated (using ECSDA with curve secp192r1 and the private key) over the hash and the session ID.

The resulting 48-byte long signature will be output in the next UBX-SEC-ECSIGN message, which is output after the last combined message. This message will also contain information on the number of messages hashed since the last UBX-SEC-ECSIGN, the SHA-256 32-byte long hash of the combined messages and the configured session ID. The host can use all information in the latest UBX-SEC-ECSIGN message to verify that all messages received after the previous UBX-SEC-ECSIGN message originated from the configured NEO-M9L.

Enabling UBX-SEC-ECSIGN, the expected output behavior and expected output contents are outlined in the example later in this section. See [Example].

3.11.4.6 Signature verification

A host receiving a data stream from a port (R) of the NEO-M9L, where the asymmetric signing mechanism has been enabled, can now verify the receiver-to-host communication channel.

In order to check the authenticity of the received data, the host needs to keep track of the following:

- B = A buffer containing all the binary data output from port R, from the previously received UBX-SEC-ECSIGN and until the last received UBX-SEC-ECSIGN. The UBX-SEC-ECSIGN messages should not be part of the buffer data.
- N = The number of messages received by the host from port R, from the previously received UBX-SEC-ECSIGN and until the last received UBX-SEC-ECSIGN. The UBX-SEC-ECSIGN messages should not be counted as part of N.
- P, Px, Py = The public key (P) that matches the private key (d), which was earlier written to the NEO-M9L. The public key is expected to be a 48-byte long value; commonly split into a pair of 24-byte long values (Px and Py).
- S = The latest configured session ID (S) of the NEO-M9L. This is expected to be a 24-byte long value.

Once a UBX-SEC-ECSIGN is received by the host, the host needs to do the following to verify the NEO-M9L output since the previously received UBX-SEC-ECSIGN:

- Verify that the msgNum field in the last received UBX-SEC-ECSIGN is equal to N
- Compute an SHA-256 32-byte long hash (H) of the combined binary data in B. Verify that the finalHash field in the last received UBX-SEC-ECSIGN is identical with the hash (H) calculated by the host.
- Verify that the sessionId field in the last received UBX-SEC-ECSIGN is identical to S
- Calculate the signature verification input (signatureInput) as follows, based on the hash (H) and the session ID (S):
  - hashAndId = concatenate the hash (H) and the session ID (S)
  - hashAndIdSha256 = calculate an SHA-256 32-byte long hash on hashAndId
  - hashAndIdSha256Part1 = the first 8 bytes of hashAndIdSha256
  - hashAndIdSha256Part2 = the next 16 bytes of hashAndIdSha256
  - hashAndIdSha256Part3 = the last 8 bytes of hashAndIdSha256
  - hashAndIdSha256PartsXor = hashAndIdSha256Part1 XOR hashAndIdSha256Part3
  - signatureInput = concatenate hashAndIdSha256PartsXor and hashAndIdSha256Part2
- Run an ECDSA verification (with curve secp192r1) using:
  - the signature verification input (signatureInput)
  - the public key (Px and Py)
  - the ecdsaSignature field in the last received UBX-SEC-ECSIGN
- The ECDSA verification must pass

If all the above conditions are met on the host, the receiver-to-host communication channel can be considered as verified in the time frame between the two last received UBX-SEC-ECSIGN messages.

☞ The above verification procedure can also be applied at startup. The only difference is that the first received UBX-SEC-ECSIGN message will cover all output since startup. The host must take care to remember the last configured session ID (S) or to ensure that the last configured session ID (S) has been saved in the non-volatile storage.

The above host-side verification procedure of the NEO-M9L output is outlined in the example below. See [Example].

3.11.4.7 Example

All steps described so far are necessary for setting up the asymmetric signature mechanism and authenticating the receiver-to-host communication channel successfully. Here follows a simplified example demonstrating the minimum steps that need to be taken for configuration and verification. Note that it does not include further security enhancements, such as rotating the session ID.

Asymmetric signature (private and public keys): Generate an ECDSA F(p) key pair using a secp192r1 curve. This can be done with the library/tool of your choice. One available calculation tool that can be used is SCV Cryptomanager (http://www.cryptomanager.com/). For the purposes of this example, the following keys were generated:

- Private (d): F2 15 C8 47 6E B5 13 FA BF FD 27 1D 92 57 40 BF 45 E0 29 A3 77 67 8A 35
- Public (Px): C7 85 96 D5 35 C9 7D 6B 27 2A BE 6B 9B D0 88 60 2A 77 DE DE 09 A7 9C 16
- Public (Py): 1D 8A 9D F4 31 C3 42 67 82 70 F8 95 1E 33 77 2A C3 1B 6E 57 4D 23 AA BD

Writing the private key in the eFuse: Generate a UBX-CFG-OTP message to write the private key (d) in the eFuse file 0xa6. This can be done in u-center or by manually constructing the message following the UBX-CFG-OTP message specification.

Constructing the message in u-center: Select the UBX-CFG-OTP message view. Select version u-blox 9 and ID 0xa6.

- Enter the private key (d) as a 24-byte hex value space separated: F2 15 C8 47 6E B5 13 FA BF FD 27 1D 92 57 40 BF 45 E0 29 A3 77 67 8A 35
- The resulting binary string is: B5 62 06 41 24 00 04 01 A6 98 C0 4B 10 A0 28 EF 12 05 F2 15 C8 47 6E B5 13 FA BF FD 27 1D 92 57 40 BF 45 E0 29 A3 77 67 8A 35 53 FF

Manually constructing the message: Following the message specification in the interface description [2], the message fields should get populated as follows:

- Message header: 0xb5 0x62, Class: 0x06, ID: 0x41, Length: 0x24
- Payload.version: 0x04, Payload.operation: 0x01, Payload.fileID: 0xa6, Payload.lengthMask: 0x98
- Payload.CRC: 0xa0104bc0
- Payload.key: 0x0512ef28
- Payload.privateKey: 0xF2 15 C8 47 6E B5 13 FA BF FD 27 1D 92 57 40 BF 45 E0 29 A3 77 67 8A 35
- Message checksum: 0x53 0xFF
- The resulting binary string is: B5 62 06 41 24 00 04 01 A6 98 C0 4B 10 A0 28 EF 12 05 F2 15 C8 47 6E B5 13 FA BF FD 27 1D 92 57 40 BF 45 E0 29 A3 77 67 8A 35 53 FF
- The CRC field can be obtained by calculating an IEEE-802.3 32-bit CRC over the fields version, operation, fileID, lengthMask and privateKey. The polynomial should be set to 0x04c11db7, which is the default polynomial for IEEE-802.3 32-bit CRC. The initial value (seed) should be set to 0xb55db55d. The final XOR value should be 0x0. The input and output must be reflected. The CRC calculation can be done with the library/tool of your choice. Here (http://www.sunshine2k.de/articles/coding/crc/understanding_crc.html) is one available online resource that can be used.

⚠ Writing a file for the private key in the eFuse is a one-time write-only irreversible operation. It is not possible to delete a private key file.

Once the UBX-CFG-OTP binary string is available, send it to the NEO-M9L. There should be a UBX-ACK-ACK for class 0x06 and ID 0x41 to confirm successful writing of the file. Further confirmation can be obtained by polling the UBX-CGF-OTP message in u-center (OTP Details in UBX-CFG-OTP), where the new file for the private key should now be listed as 'Asym. host i/f signing private key'. If a file for a private key already exists, a new one cannot be written again. Any attempt to write another file for the private key will be rejected and result in a UBX-ACK-NAK.

Configuring the session ID: Set a session ID by setting the configuration items CFG-SEC-ECCFGSESSIONID0, CFG-SEC-ECCFGSESSIONID1 and CFG-SEC-ECCFGSESSIONID2. For the purposes of this example, we will set them to:

- CFG-SEC-ECCFGSESSIONID0 = 0x1111111111111111
- CFG-SEC-ECCFGSESSIONID1 = 0x2222222222222222
- CFG-SEC-ECCFGSESSIONID2 = 0x3333333333333333

Enabling signature output: Enable the UBX-SEC-ECSIGN message on the port where the port output is to be signed. This is done by setting the configuration items CFG-MSGOUT-UBX_SEC_ECSIGN_*. For the purposes of this example, we will set the UBX-SEC-ECSIGN to be output every 2 seconds on UART1 as follows:

- CFG-MSGOUT-UBX_SEC_ECSIGN_UART1 = 2

Monitor the output on the desired port to observe that there is the regular port output plus the newly enabled UBX-SEC-ECSIGN message at the configured rate. For the purposes of this example, the output on UART1 will look as follows (assuming only one regular UBX message is enabled):

Time	Message	Comment
15:34:18	UBX-SEC-ECSIGN	
15:34:18	UBX-MON-COMMS	
15:34:19	UBX-MON-COMMS	
15:34:20	UBX-SEC-ECSIGN	This is signing the 2 previous messages
15:34:20	UBX-MON-COMMS	
15:34:21	UBX-MON-COMMS	
15:34:22	UBX-SEC-ECSIGN	This is signing the 2 previous messages

Table 25: Example output on UART1 when output rate of UBX-SEC-ECSIGN on UART1 is set to 2 and only one regular UBX message is enabled on UART1 at rate 1

Signature verification: For the purposes of this example we will focus on the last three messages to show how we can use the contents of the last UBX-SEC-ECSIGN message to verify the last two UBX-MON-COMMS messages. In other words, how to verify the messages from the previously received UBX-SEC-ECSIGN. Here are the binary contents of these last three messages:

Time	Message	Binary contents
15:34:20	UBX-MON-COMMS	B5 62 0A 36 80 00 00 03 00 00 00 01 05 FF 00 01 00 00 19 16 03 00 00 56 00 00 0C 41 00 00 00 00 00 00 30 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 00 88 E4 1D 00 00 09 00 00 23 E8 49 00 08 2C 00 00 48 51 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1C 00 00 00 01 01 00 00 0C C8 49 00 00 00 00 00 5E 72 17 00 08 0C 00 00 47 1A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1E 2A
15:34:21	UBX-MON-COMMS	B5 62 0A 36 80 00 00 03 00 00 00 01 05 FF 00 01 00 00 A1 16 03 00 00 56 00 00 0C 41 00 00 00 00 00 00 30 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 00 D0 EB 1D 00 00 09 00 00 0F 02 4A 00 06 2C 00 00 69 51 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1C 00 00 00 01 01 00 00 F8 E1 49 00 00 00 00 00 A6 79 17 00 07 0C 00 00 4F 1A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 76 F3
15:34:22	UBX-SEC-ECSIGN	B5 62 27 04 6C 00 01 00 02 00 AD CD C2 68 E6 9F 16 F0 EB D7 A6 71 5E 16 4D AD D6 AB A4 03 E5 EC 16 C0 6C 97 39 9F EA 6F DD 85 11 11 11 11 11 11 11 22 22 22 22 22 22 22 22 33 33 33 33 33 33 33 B0 D7 BD C7 AC 81 19 42 FF 53 07 66 9B 76 FB 38 57 2B A0 63 56 D8 ED 7F 55 5F 74 E5 20 13 EC D4 76 18 EF 53 11 D3 DF AC C0 1E 0A E6 ED 89 F3 72 42 5E

Table 26: Example binary contents of the last three messages

While the host has been receiving these messages, it should have kept track of the following values:

Value description	Value contents for this example
B = A buffer containing all the binary data output from port R, since the previously received UBX-SEC-ECSIGN and until the last received UBX-SEC-ECSIGN. The UBX-SEC-ECSIGN messages should not be part of the buffer data.	B = B5 62 0A 36 80 00 00 03 00 00 00 01 05 FF 00 01 00 00 19 16 03 00 00 56 00 00 0C 41 00 00 00 00 00 00 30 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 00 88 E4 1D 00 00 09 00 00 23 E8 49 00 08 2C 00 00 48 51 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1C 00 00 00 01 01 00 00 0C C8 49 00 00 00 00 00 5E 72 17 00 08 0C 00 00 47 1A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1E 2A B5 62 0A 36 80 00 00 03 00 00 00 01 05 FF 00 01 00 00 A1 16 03 00 00 56 00 00 0C 41 00 00 00 00 00 00 30 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 00 D0 EB 1D 00 00 09 00 00 0F 02 4A 00 06 2C 00 00 69 51 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1C 00 00 00 01 01 00 00 F8 E1 49 00 00 00 00 00 A6 79 17 00 07 0C 00 00 4F 1A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 76 F3
N = The number of messages received by the host from port R, since the previously received UBX-SEC-ECSIGN and until the last received UBX-SEC-ECSIGN. The UBX-SEC-ECSIGN messages should not be counted as part of N.	N = 2
P, Px, Py = The public key (P) that matches the private key (d), which was earlier written to the NEO-M9L. The public key is expected to be a 48-byte long value; commonly split into a pair of 24-byte long values (Px and Py).	Px = C7 85 96 D5 35 C9 7D 6B 27 2A BE 6B 9B D0 88 60 2A 77 DE DE 09 A7 9C 16 Py = 1D 8A 9D F4 31 C3 42 67 82 70 F8 95 1E 33 77 2A C3 1B 6E 57 4D 23 AA BD
S = The latest configured session ID (S) of the NEO-M9L. This is expected to be a 24-byte long value.	S = 11 11 11 11 11 11 11 11 22 22 22 22 22 22 22 22 33 33 33 33 33 33 33 33

Table 27: Example values to be tracked by the host for the signature verification

Given the last received UBX-SEC-ECSIGN, the host now can verify the messages received since the previously received UBX-SEC-ECSIGN. Based on the binary contents of the last received UBX-SEC-ECSIGN, we can verify as follows:

- The field msgNum in the last received UBX-SEC-ECSIGN is 0x0002.
  - This is equal to N = 2 that the host has been keeping track of.
- The field finalHash in the last received UBX-SEC-ECSIGN is AD CD C2 68 E6 9F 16 F0 EB D7 A6 71 5E 16 4D AD D6 AB A4 03 E5 EC 16 C0 6C 97 39 9F EA 6F DD 85.
  - Performing an SHA-256 32-byte long hash of the binary data in the buffer B of the host yields the same hash as the field finalHash.
- The field sessionId in the last received UBX-SEC-ECSIGN is 11 11 11 11 11 11 11 11 22 22 22 22 22 22 22 22 22 33 33 33 33 33 33 33 33.
  - This is identical to the last configured session ID (S) that the host has been keeping track of.

Before the final verification of the ecdsaSignature field, we need to extract the signature verification input using the finalHash and sessionId fields from the last received UBX-SEC-ECSIGN message:

Action	Result
Get the finalHash field from the last received UBX-SEC-ECSIGN message	finalHash = AD CD C2 68 E6 9F 16 F0 EB D7 A6 71 5E 16 4D AD D6 AB A4 03 E5 EC 16 C0 6C 97 39 9F EA 6F DD 85
Get the sessionId field from the last received UBX-SEC-ECSIGN message	sessionId = 11 11 11 11 11 11 11 11 22 22 22 22 22 22 22 22 33 33 33 33 33 33 33 33
Concatenate finalHash and sessionId	hashAndId = AD CD C2 68 E6 9F 16 F0 EB D7 A6 71 5E 16 4D AD D6 AB A4 03 E5 EC 16 C0 6C 97 39 9F EA 6F DD 85 11 11 11 11 11 11 11 11 22 22 22 22 22 22 22 22 33 33 33 33 33 33 33 33
Calculate an SHA-256 32-byte hash on hashAndId	hashAndIdSha256 = DD 17 07 34 C9 9E 5D 6C 0B 91 D1 D8 BF E8 0B 7B 0F E0 0D E3 EE C0 E7 6D 04 4B 2D F3 84 97 E4 12
Get the first 8 bytes of hashAndIdSha256	hashAndIdSha256Part1 = DD 17 07 34 C9 9E 5D 6C
Get the next 16 bytes of hashAndIdSha256	hashAndIdSha256Part2 = 0B 91 D1 D8 BF E8 0B 7B 0F E0 0D E3 EE C0 E7 6D

Action	Result
Get the last 8 bytes of hashAndIdSha256	hashAndIdSha256Part3 = 04 4B 2D F3 84 97 E4 12
Do a XOR between hashAndIdSha256Part1 and hashAndIdSha256Part3	hashAndIdSha256PartsXor = D9 5C 2A C7 4D 09 B9 7E
Concatenate hashAndIdSha256PartsXor and hashAndIdSha256Part2 to form the signature input	signatureInput = D9 5C 2A C7 4D 09 B9 7E 0B 91 D1 D8 BF E8 0B 7B 0F E0 0D E3 EE C0 E7 6D

Table 28: Example steps to extract the signature verification input using the finalHash and sessionId fields from the last received UBX-SEC-ECSIGN message

The final step is to verify the field ecdsaSignature in the last received UBX-SEC-ECSIGN. For that you need to run an ECDSA verification using a secp192r1 curve. This can be done with the library/tool of your choice. One available calculation tool that can be used is SCV Cryptomanager (http://www.cryptomanager.com/). The following input parameters must be provided for the verification:

- Input = D9 5C 2A C7 4D 09 B9 7E 0B 91 D1 D8 BF E8 0B 7B 0F E0 0D E3 EE C0 E7 6D
- Public (Px) = C7 85 96 D5 35 C9 7D 6B 27 2A BE 6B 9B D0 88 60 2A 77 DE DE 09 A7 9C 16
- Public (Py) = 1D 8A 9D F4 31 C3 42 67 82 70 F8 95 1E 33 77 2A C3 1B 6E 57 4D 23 AA BD
- Signature part 1 (the first half of ecdsaSignature) = B0 D7 BD C7 AC 81 19 42 FF 53 07 66 9B 76 FB 38 57 2B A0 63 56 D8 ED 7F
- Signature part 2 (the second half of ecdsaSignature) = 55 5F 74 E5 20 13 EC D4 76 18 EF 53 11 D3 DF AC C0 1E 0A E6 ED 89 F3 72
- Running the ECDSA verification using a secp192r1 curve shows that the field ecdsaSignature is verified

In this example, all conditions have been met on the host. Therefore, the receiver-to-host communication channel can be considered as verified in the time frame between the two last received UBX-SEC-ECSIGN messages.

3.12 eFuse OTP layer configuration

eFuse OTP is a form of one-time programmable (OTP) memory and consists of an array of fuses. The changes made to the bits (i.e. burning the fuse) can only be done once and there is no way to undo any changes. This memory is used for configuration settings and ROM patches that need to be stored permanently in the receiver. For saving information in the eFuse OTP, refer to the message UBX-CFG-OTP in NEO-M9L Interface description [2].

☞ Use the u-center application to compose the UBX-CFG-OTP message for storing the desired settings.

3.12.1 eFuse OTP organization in u-blox 9

The eFuse OTP consists of 4096 bits. It is divided into two sections: a fixed section (eFuse OTP fixed section) and a dynamic section (eFuse OTP dynamic section). The fixed section consists of 256 bits, where each bit or a combination of bits is a defined setting. The capacity of the dynamic section is 3840 bits, its content is written on request by customers and organized in files. As there is no fixed organization of the dynamic section, it should be filled cautiously, as it is used for ROM patches that may be released at a later time.

Settings written to the eFuse OTP are not applied to the receiver until reset is performed. It is possible to write several settings to the eFuse OTP before applying a reset.

3.12.2 eFuse OTP fixed section

The following bits are defined in the fixed section of the eFuse OTP:

Name	Options	Description
DC/DC enable		With this option, the internal DC/DC converter can be enabled
IO monitor		The IO monitor value defines the threshold when the receiver is released from the reset state to running state.
Oscillator type		The oscillator type configuration sets the LDO_X_OUT voltage to supply the TCXO
V_IO voltage range	V_IO range high (2.7 V - 3.3 V)	Supply voltage for PIOs and input voltage for clock and backup domains
eFuse protection (not reversible!)	eFuse OTP is not write protected (default) eFuse OTP is write protected	If the bit is permanently set to zero, the OTP memory content (fixed and dynamic section) is protected and cannot be modified

Table 29: Defined bits for fixed section of the eFuse OTP

☞ Use the u-center application to compose the UBX-CFG-OTP message to write to the reserved bits.

☞ If the eFuse protection bit is permanently set to zero, the content cannot be modified. That implies that future firmware patches cannot be stored.

3.12.3 eFuse OTP dynamic section

Settings in the dynamic section are grouped together in files. Files are available in the dynamic section of the eFuse OTP memory:

ID	Name	Length (bytes)	Payload
0x30	Receiver configuration	13	The receiver configuration (see UBX-CFG-OTP Write file 0x30)
0xA6	ECSIGN configuration	36	The configuration for the host interface signature (see UBX-CFG-OTP Write file 0xa6)

Table 30: Available files in the dynamic section of the eFuse OTP memory

☞ Refer to the u-blox NEO-M9L Interface description [2] for more information about UBX-CFG-OTP message.

3.13 u-blox protocol feature descriptions

3.13.1 Broadcast navigation data

This section describes the data reported via UBX-RXM-SFRBX.

UBX-RXM-SFRBX reports the broadcast navigation data message the receiver has collected from each tracked signal. When enabled, a separate message is generated each time the receiver decodes a complete subframe of data from a tracked signal. The data bits are reported as received, including preambles and error checking bits as appropriate. However, because there is considerable variation in the data structure of the different GNSS signals, the form of the reported data also varies. This document uses the term "subframe", but other GNSS data structures might use different terms, for example, GLONASS uses "strings" and Galileo uses "pages".

3.13.1.1 Parsing navigation data subframes

Each UBX-RXM-SFRBX message contains a subframe of data bits appropriate for the relevant GNSS, delivered in a number of 32-bit words, as indicated by numWords field.

Due to the variation in data structure between different GNSS, the most important step in parsing a UBX-RXM-SFRBX message is to identify the form of the data. This should be done by reading the

gnssId field, which indicates which GNSS the data was decoded from. In almost all cases, this is sufficient to indicate the structure. Because of this, the following sections are organized by GNSS. However, in some cases the identity of the GNSS is not sufficient, and this is described, where appropriate, in the following sections.

In most cases, the data does not map perfectly into a number of 32-bit words and, consequently, some of the words reported in UBX-RXM-SFRBX messages contain fields marked as "Pad". These fields should be ignored and no assumption should be made about their contents.

UBX-RXM-SFRBX messages are only generated when complete subframes are detected by the receiver and all appropriate parity checks have passed.

Where the parity checking algorithm requires data to be inverted before it is decoded (e.g. GPS L1C/A), the receiver carries this out before the message output. Therefore, users can process data directly and do not need to worry about repeating any parity processing.

The meaning of the content of each subframe depends on the sending GNSS and is described in the relevant interface control documents (ICD).

3.13.1.2 GPS

NEO-M9L is designed to receive and track the L1C/A signals provided at 1575.42 MHz by the Global Positioning System (GPS).

3.13.1.2.1 GPS L1C/A

For GPS L1C/A signals, there is a fairly straightforward mapping between the reported subframe and the structure of subframe and words described in the GPS ICD. Each subframe comprises ten data words, which are reported in the same order they are received.

Each word is arranged as follows:

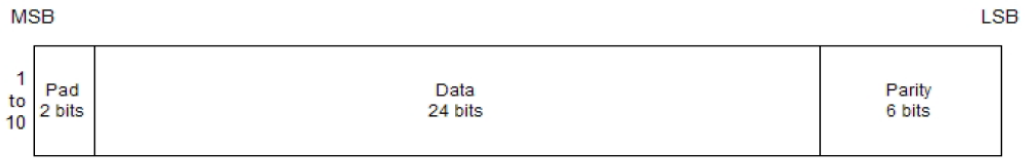


Figure 22: GPS L1C/A subframe word

3.13.1.3 GLONASS

The receivers are designed to receive and track the L1OF signals GLONASS provides at 1602 MHz +k*562.5 kHz, where k is the satellite's frequency channel number (k = -7,...,5, 6). The ability to receive and track GLONASS L1OF satellite signals allows design of GLONASS receivers where required by regulations.

The GLONASS words are arranged as follows:

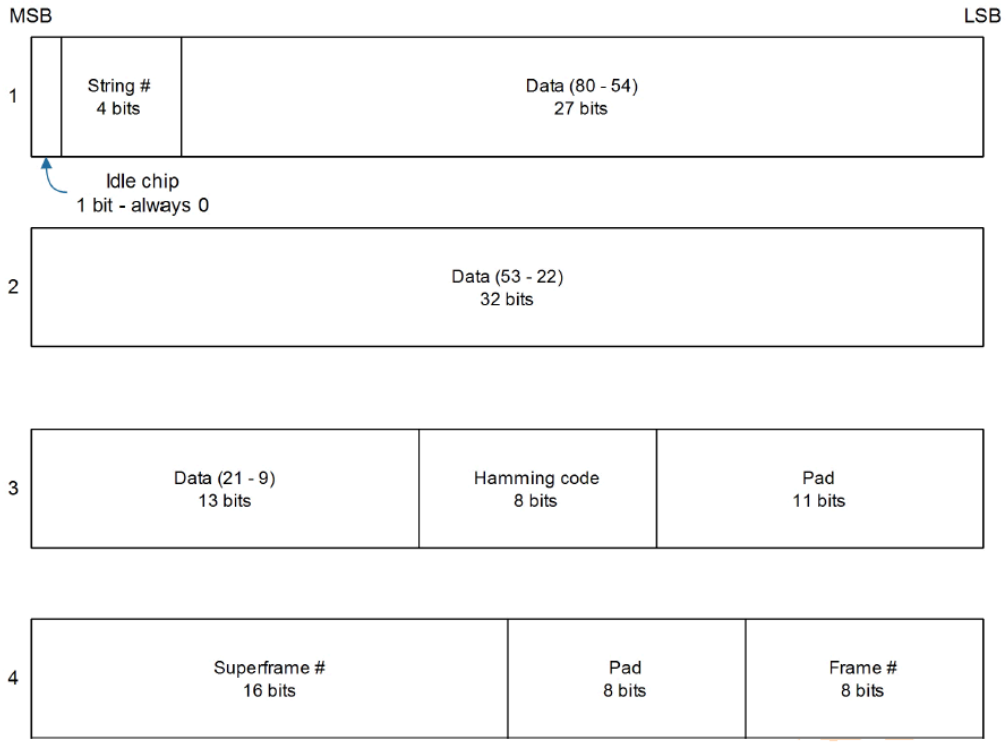


Figure 23: GLONASS navigation message data

3.13.1.4 BeiDou

u-blox M9 receivers can receive and process the B1I signals broadcast at 1561.098 MHz from the BeiDou Navigation Satellite System. The ability to receive and track BeiDou signals in conjunction with another constellation results in higher coverage, improved reliability and better accuracy.

Each word is arranged as follows:

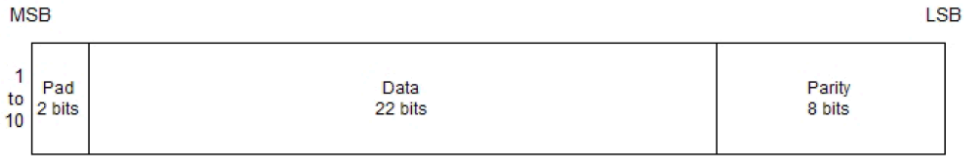


Figure 24: BeiDou subframe word

Note that as the BeiDou data words only comprise 30 bits, the 2 most significant bits in each word reported by UBX-RXM-SFRBX are padding and should be ignored.

3.13.1.5 Galileo

u-blox M9 receivers running can receive and track the E1-B/C signals centered on the GPS L1 frequency band. GPS and Galileo signals can be processed concurrently together with BeiDou and GLONASS signals, enhancing coverage, reliability and accuracy. The SAR return link message (RLM) parameters for both short and long versions are decoded by the receiver and made available to users via UBX proprietary messages and via standard NMEA RLM sentences.

3.13.1.5.1 Galileo E1-B

For the Galileo E1-B signal, each reported subframe contains a pair of I/NAV pages as described in the Galileo ICD. Galileo pages can either be "Nominal" or "Alert" pages. For Galileo "Nominal" pages the eight words are arranged as follows:

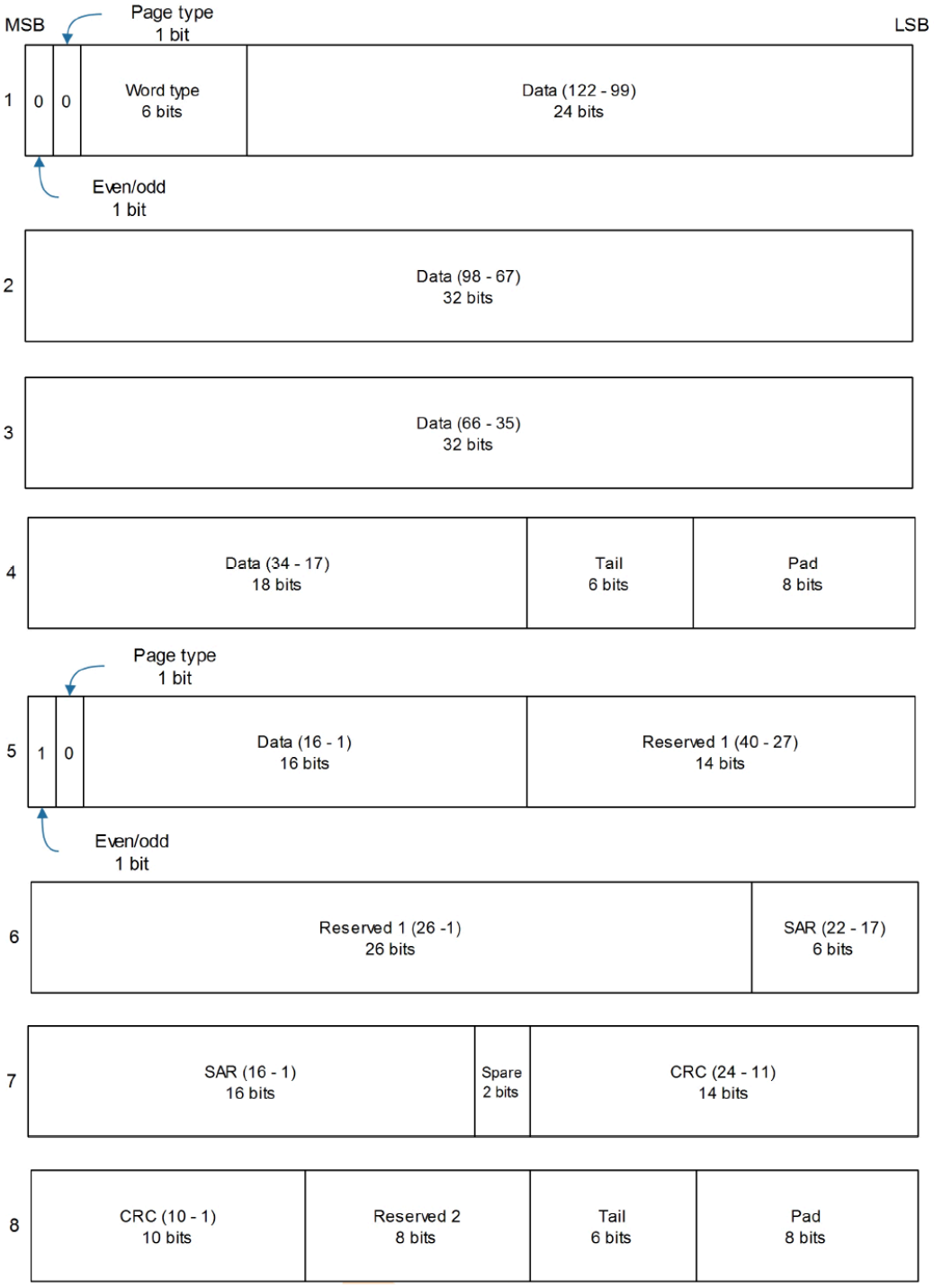


Figure 25: Galileo E1-B subframe words

Alert pages are reported in very similar manner, but the page type bits will have value 1 and the structure of the eight words will be slightly different (as indicated by the Galileo ICD).

3.13.1.6 SBAS

For SBAS (L1C/A) signals each reported subframe contains eight 32-bit data words to deliver the 250 bits transmitted in each SBAS data block.

The eight words are arranged as follows:

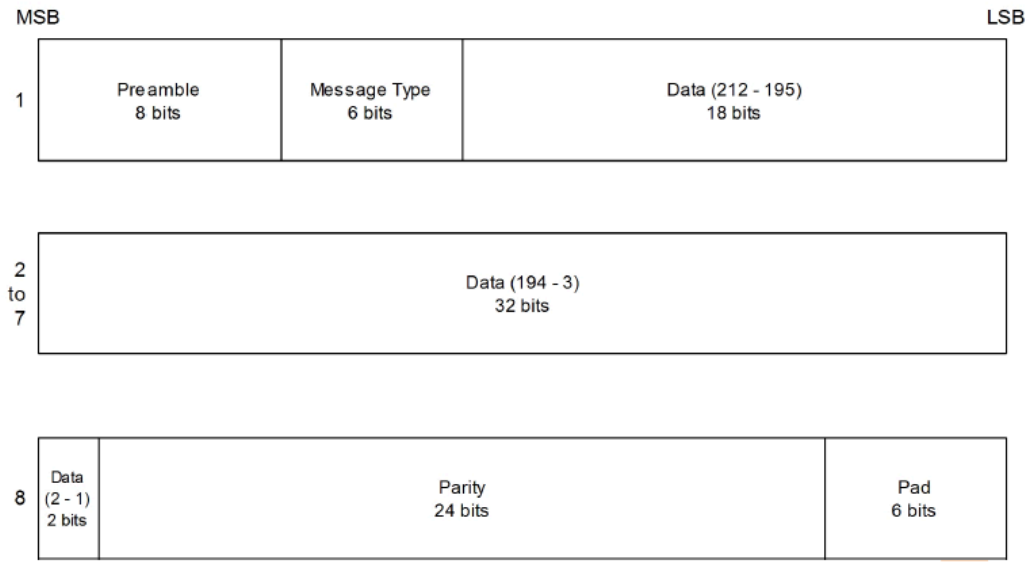


Figure 26: SBAS subframe words

3.13.1.7 QZSS

The structure of the data delivered by QZSS L1C/A signals is effectively identical to that of GPS (L1C/A).

3.13.1.8 Summary

The following table gives a summary of the different data message formats reported by the UBX-RXM-SFRBX message:

GNSS	Signal	gnssId	sigId	numWords	period
GPS	L1C/A	0	0	10	6s
SBAS	L1C/A	1	0	9	1s
Galileo	E1 B	2	1	8	2s
BeiDou	B1I D1	3	0	10	6s
BeiDou	B1I D2	3	1	10	0.6s
QZSS	L1C/A	5	0	10	6s
GLONASS	L1OF	6	0	3	2s

Table 31: Data message formats reported by UBX-RXM-SFRBX

3.14 Forcing a receiver reset

Typically, in GNSS receivers, a distinction is made between cold, warm, and hot start, depending on the type of valid information the receiver has at the time of the restart.

- **Cold start:** In cold start mode, the receiver has no information from the last position (e.g. time, velocity, frequency etc.) at startup. Therefore, the receiver must search the full time and frequency space, and all possible satellite numbers. If a satellite signal is found, it is tracked to decode the ephemeris (18-36 seconds under strong signal conditions), whereas the other channels continue to search satellites. Once there is a sufficient number of satellites with valid ephemeris, the receiver can calculate position and velocity data. Other GNSS receiver manufacturers call this startup mode **Factory startup**.
- **Warm start:** In warm start mode, the receiver has approximate information for time, position, and coarse satellite position data (Almanac). In this mode, after power-up, the receiver normally needs to download ephemeris before it can calculate position and velocity data. As the ephemeris data usually is outdated after 4 hours, the receiver will typically start with a warm start if it has been powered down for more than 4 hours. In this scenario, several augmentations are possible. See Multiple GNSS assistance.
- **Hot start:** In hot start mode, the receiver was powered down only for a short time (4 hours or less), so that its ephemeris is still valid. Since the receiver does not need to download ephemeris again, this is the fastest startup method.

Using the UBX-CFG-RST message, you can force the receiver to reset and clear data, in order to see the effects of maintaining/losing such data between restarts. For this, the UBX-CFG-RST message offers the `navBbrMask` field, where hot, warm and cold starts can be initiated, and also other combinations thereof.

The reset type can also be specified. This is not related to GNSS, but to the way the software restarts the system.

- **Hardware reset** uses the on-chip watchdog, in order to electrically reset the chip. This is an immediate, asynchronous reset. No Stop events are generated.
- **Controlled software reset** terminates all running processes in an orderly manner and, once the system is idle, restarts operation, reloads its configuration and starts to acquire and track GNSS satellites.
- **Controlled software reset (GNSS only)** only restarts the GNSS tasks, without reinitializing the full system or reloading any stored configuration.
- **Hardware reset (after shutdown)** uses the on-chip watchdog. This is a reset after shutdown.
- **Controlled GNSS stop** stops all GNSS tasks. The receiver will not be restarted, but will stop any GNSS-related processing.
- **Controlled GNSS start** starts all GNSS tasks.

3.15 Firmware upload

NEO-M9L is supplied with firmware. u-blox may release updated images containing, for example, security fixes, enhancements, bug fixes, etc. Therefore it is important that customers implement a firmware update mechanism in their system.

The messages in the UBX-UPD class provide an interface to transfer memory data from the host to the receiver and vice versa. Additionally, there are messages to perform specific sequences such as the flash memory erase and write routines, reading the flash organization, or calculating a checksum over a memory region. This class also includes some special messages to reboot the receiver in a special mode, or to control queue sizes at the receiver.

u-blox can provide a platform-independent sample code for updating the firmware.

3.15.1 Firmware update to flash

The following sequence diagram describes the procedure before transferring to flash:

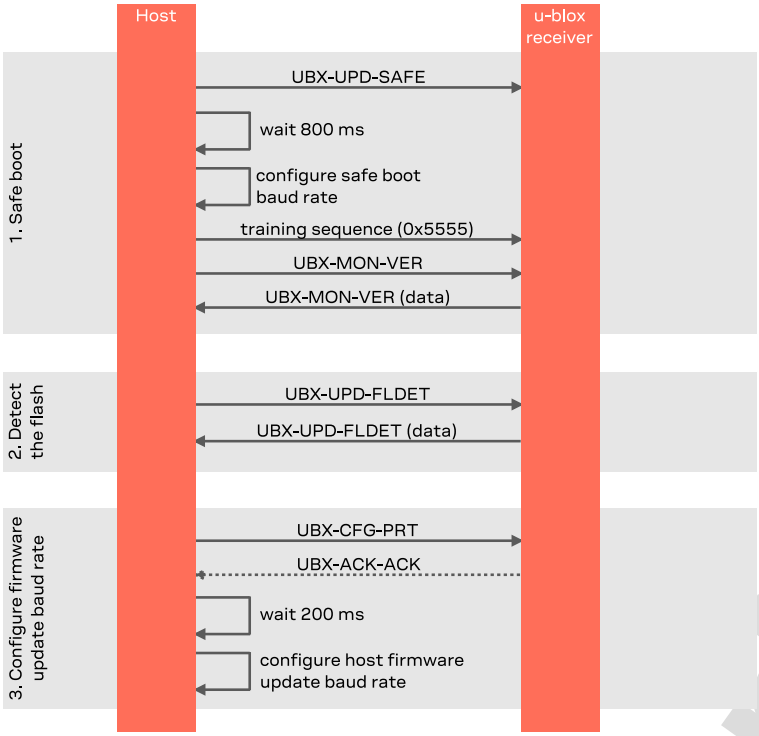


Figure 27: Procedure before transferring to flash

A firmware image is a binary file containing the software to be run by the receiver. A firmware update is the process of transferring a firmware image to the receiver and storing it in non-volatile flash memory.

While the erase and write procedures are running, the flash memory (including processor instructions) cannot be read. For this reason, the receiver must be booted from the internal ROM memory prior to erasing or writing the flash. The message UBX-UPD-SAFE prompts the receiver to perform a reset, executing code from ROM. As the receiver starts from ROM, stored configurations will not be applied, including the baud rate for the serial ports.

As the receiver is running out of the internal ring oscillator, a training sequence must be sent to synchronize the baud rate (which is expected to be at 9600 baud). This consists of sequence 0x55,0x55. Note that this message is not encapsulated in a UBX message, but consists only of these two bytes. After sending the training sequence, the host has to wait for at least 2 ms before sending messages to the receiver. To make sure that communication still functions, the version can be polled by sending the UBX-MON-VER message. See *[1.Safeboot]* in Figure 27.

Since different flash types are supported, the flash organization (block sizes and number of blocks) can be determined by reading the unique manufacturer and device IDs from the flash. The organization structure can then be looked up at the host. The IDs are read by processing the receiver's response to the UBX-UPD-FLDET message. See *[2.Detect the flash]* in Figure 27.

The following sequence diagram describes the erasing and transferring to flash:

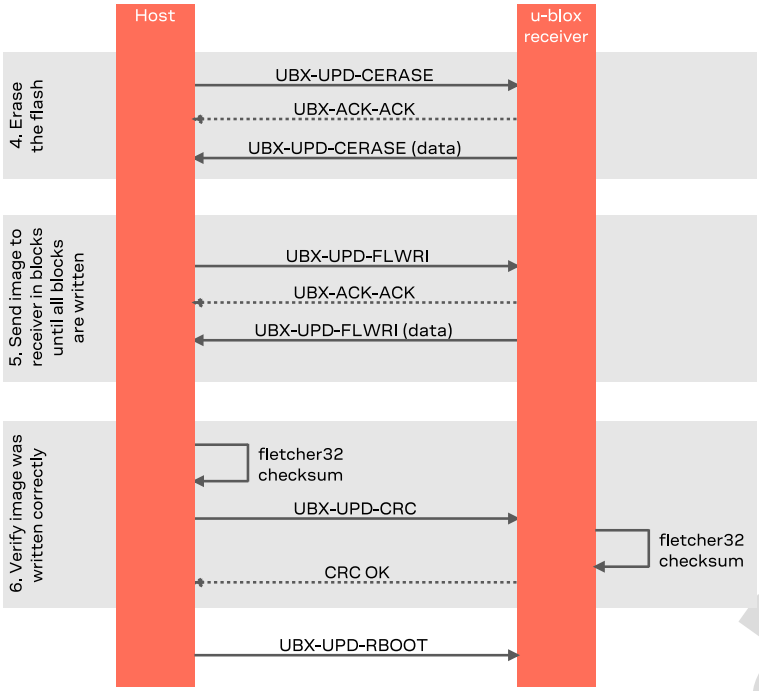


Figure 28: Erasing and transferring sequence

With the flash organization available, the main firmware update procedure can begin. Flash memory needs to be erased before it can be written.

The simplest procedure is to first delete all sectors necessary for the firmware to fit by using the UBX-UPD-CERASE. See *[4.Erase the flash]* in Figure 28.

Then, beginning with the first packet, each packet is transferred to the receiver by using the UBX-UPD-FLWRI command until the whole image is written to the flash. See *[5. Send the image...]* in Figure 28. A faster alternative (as can also be seen in the sample code) is to transfer write-packets into the receiver's queue while the receiver waits for the flash sectors to be written.

See the verification description from the section Firmware verification below. See *[6.Verify...]* in Figure 28.

If unsuccessful, the process can be repeated. If the update has completed, the receiver can be rebooted by sending the UBX-UPD-RBOOT command.

3.15.2 Firmware update sample code

The firmware update sample shows the whole firmware update process implementation. A platform-dependent code has been added for Windows-based and Linux-based operating systems and should be easily adaptable for any other platform. The code was written for little-endian systems, and can also be adapted to work on big-endian systems. The sample code includes code to use the following communication interfaces:

- Serial (RS-232)
- USB
- I2C via Aardvark hardware (a USB to I2C converter board)
- SPI via Aardvark hardware (a USB to SPI converter board)

The sample code for the firmware update written in C programming language is available from u-blox. Contact your nearest u-blox Field Application Engineer to get a copy.

3.15.3 Firmware verification

Before starting the new firmware, the content can be verified against the image by using the UBX-UPD-CRC (version 1) message. This message checks the validity of the uploaded image without any knowledge about the image structure. This is done by computing a fletcher32 checksum of the binary on the host, and then sending that computed checksum to the receiver, which will do the same and compare it to the received checksum. See the table below for details about the field values. The result will then be returned in UBX-UPD-CRC message.

Field	Value
version	0x01
region	0x00(RAM) or 0x01(flash)
addr	0x00800000(RAM) or 0x48(flash)
size	actual size of binary
crcA	First word of fletcher32 checksum of binary
crcB	Second word of fletcher32 checksum of binary

Table 32: Fields of UBX-UPD-CRC (version 1)

3.16 Production test

The NEO-M9L module contains firmware in the internal flash memory. The receiver is pre-configured to be ready for use. However, additional configurations can be done during production.

Proposed steps to configure and start up the NEO-M9L in production:

1. System monitoring which includes:
   - Verify that the correct firmware image has been uploaded (for example by polling UBX-MON-VER content).
   - Verify the configuration which has been set by polling the configuration items and compare them with the expected result.
   - Verify the correct behavior of the selected power modes (tracking modes and backup modes). Plus test the correct behavior of the supply domains (battery, LDOs etc).
   - Verify the functionality of PIOs as described in Verification of PIO pins.
   - Verify the functionality of additional PIOs such as TIMEPULSE or EXTINT pins.
   - Verify the functionality of sensors as described in Sensor production test.
2. Test the GNSS performance as described in Test GNSS performance.

3.16.1 Verification of PIO pins

The UBX-CFG-PIO and UBX-MON-HW3 messages allow setting and inspection of PIO pins. The message UBX-CFG-PIO is used to set PIO pin test mode for the receiver and the state of PIO pins. It specifies one of two requests.

Request 1 (SET-PIN) sets the receiver into a test mode in which PIO pin state can be changed (for example by being driven low or pulled high). Subsequent SET-PIN messages will change the PIO configuration while leaving the receiver in test mode.

Request 0 (EXIT-TEST) sets the receiver out of the test mode and restores the PIO state that existed before the test mode was entered.

The reply message to a poll for UBX-MON-HW3 reports the current PIO pin state. The responseType -field of the poll reply tells if the receiver is in the test mode or not. In test mode the PIO pin state may have been influenced by SET-PIN requests.

3.16.2 Sensor production test

A self-test can be triggered using UBX-CFG-SPT and its result retrieved by polling UBX-MON-SPT. To start the test, the correct sensor ID must be specified. After the test has been started the UBX-MON-SPT messages can be polled and the field `testState` indicates if the test is completed. If this field indicates that the test failed, the measurement data may be invalid.

Self-test implementation in the receiver does not make pass/fail decisions. This is the tester's responsibility. UBX-MON-SPT may contain data that indicates pass/fail decision if the built-in self-test of the sensor device returns this information.

Self-test in MEMS sensors is typically implemented with an offset on measured dynamics. It must be verified if movement and vibration on the production site have a tolerable impact on the tests, and the limits must be set taking these effects into account.

3.16.3 Test GNSS performance

The message UBX-MON-PT2 reports some characteristic values of the receiver. It does not report a pass or fail, but transmits the measured values which can be compared to expected ranges by the host test system.

The message can be polled or enabled to be sent periodically. It is activated automatically with 1 Hz update rate in the Production Test mode which can be activated by the UBX-CFG-PT2 message. In this mode, the receiver tries to find and track the satellite signals by the UBX-CFG-PT2 message. It is recommended to use only 1 satellite signal per GNSS and to set it at the center frequency. Additionally, the clock frequency can be evaluated by providing an external calibrated clock (for example the GPS simulator's reference frequency). The receiver's Power-On Self-Test (POST) status reports the hardware state, the other measurement values provide quality indicators concerning the whole signal processing chain.

A standard in-circuit production test for the user application will use the UBX-MON-PT2 protocol message and will need access to a serial interface, for example I2C, SPI or UART. If the EXTINT signal is accessible in production test, frequency aiding may be used in order to reduce test time.

Value	Description	Min	Typ.	Max	Comments
rfPga	RF gain amplifier setting		~45		Gives indication about the gain of the whole RF path. If additional external LNA or active antenna is used, the value becomes lower.
clkDriftTrk	Drift of the receiver clock	-2'500 ppb		+2'500 ppb	It is mandatory that the frequency of the TCXO is within these limits at room temperature. Make sure that the used single channel simulator has no Doppler offset.
carrphDevMax	Carrier phase deviation		0.004 cycles		Gives some information about the stability of the TCXO. A typical value for a tracked satellite with C/N0 ratio of about 40 dBHz equals 0.004 cycles. For lower C/N0 the value becomes higher.
cnoMin and cnoMax	C/N0 ratio		40 dBHz		A signal level of –130 dBm results in a C/N0 of about 40 dBHz.
postStatus	Power on self test				Value depends on receiver setup, but on a similar design the value always has to be the same for all devices.

Table 33: Limits for MON-PT2 measurements

4 Design

This section provides information to help carry out a successful schematic and PCB design integrating the NEO-M9L.

4.1 Pin assignment

The pin assignment of the NEO-M9L module is shown in Figure 29. The defined configuration of the PIOs is listed in Table 34.

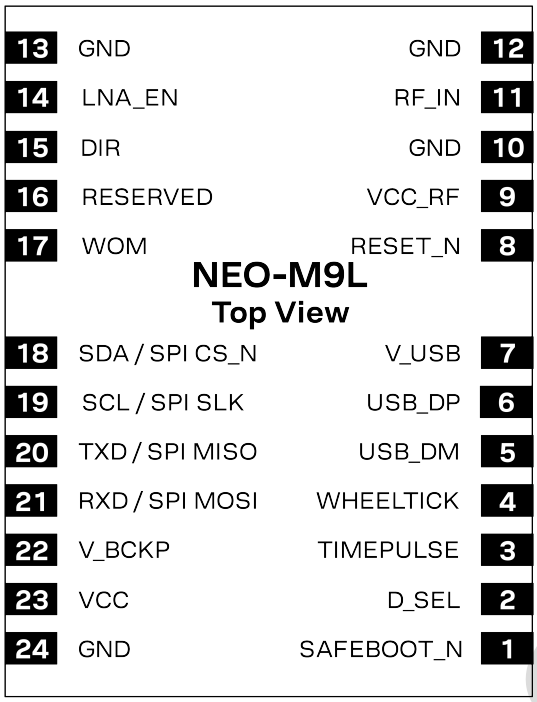


Figure 29: NEO-M9L pin assignment

Pin no.	Name	I/O	Description
1	SAFEBOOT_N	I	SAFEBOOT_N (used for FW updates and reconfiguration, leave open)
2	D_SEL	I	Interface select (open or VCC = UART + I2C; GND = SPI)
3	TIMEPULSE	O	TIMEPULSE (1 PPS, TP2)
4	WHEELTICK	I	Wheel-tick input
5	USB_DM	I/O	USB data (DM)
6	USB_DP	I/O	USB data (DP)
7	V_USB	I	USB supply
8	RESET_N	I	RESET (active low)
9	VCC_RF	O	Voltage for external LNA
10	GND	I	Ground
11	RF_IN	I	GNSS signal input
12	GND	I	Ground
13	GND	I	Ground
14	LNA_EN	O	Antenna/LNA control
15	DIR	I	Direction input for speed pulse

Pin no.	Name	I/O	Description
16	Reserved	-	Reserved
17	WOM	O	Wake on motion interrupt
18	SDA / SPI CS_N	I/O	I2C data if D_SEL = VCC (or open); SPI chip select if D_SEL = GND
19	SCL / SPI SLK	I/O	I2C clock if D_SEL = VCC (or open); SPI clock if D_SEL = GND
20	TXD / SPI MISO	O	UART output if D_SEL = VCC (or open); SPI MISO if D_SEL = GND
21	RXD / SPI MOSI	I	UART input if D_SEL = VCC (or open); SPI MOSI if D_SEL = GND
22	V_BCKP	I	Backup voltage supply
23	VCC	I	Supply voltage
24	GND	I	Ground

Table 34: NEO-M9L pin assignment

4.2 Power supply

The u-blox NEO-M9L module has three power supply pins: VCC, V_BCKP and V_USB.

4.2.1 VCC: Main supply voltage

The **VCC** pin is connected to the main supply voltage. During operation, the current drawn by the module can vary by some orders of magnitude. For this reason, it is important that the supply circuitry be able to support the peak power for a short time (see the applicable data sheet [1] for specification).

The module integrates a DC/DC converter, which allows reduced power consumption.

☞ When switching from backup mode to normal operation or at startup, u-blox NEO-M9L modules must charge the internal capacitors in the core domain. In certain situations, this can result in a significant current draw. For low-power applications using backup mode, it is important that the power supply or low ESR capacitors at the module input can deliver this current/charge.

☞ To reduce peak current during power on, users can employ an LDO that has a built-in current limiter.

☞ Do not add any series resistance greater than 0.2 Ω to the VCC supply as it will generate input voltage noise due to dynamic current conditions.

☞ For the NEO-M9L module the equipment must be supplied by an external limited power source in compliance with the clause 2.5 of the standard IEC 60950-1.

4.2.2 V_BCKP: Backup supply voltage

The V_BCKP pin can be used to provide power to maintain the real-time clock (RTC) and battery-backed RAM (BBR) when VCC is removed.

V_BCKP can be provided with a battery. If V_BCKP is provided during a VCC outage, the receiver may be able to perform a hot or warm start. Especially if the RTC and BBR contents are still current, for example, after a short VCC outage. The fusion filter data is also stored in BBR and NEO-M9L can restart in fusion mode.

If V_BCKP is not provided, the module performs a cold start at power up. Also sensors will need to be recalibrated.

If a host is connected to NEO-M9L, V_BCKP can be partially emulated by using UBX-UPD-SOS functionality. BBR data can saved to the host and restored at startup. See the applicable Interface description for more information.

- Avoid high resistance on the **V_BCKP** line: During the switch from main supply to backup supply, a short current adjustment peak can cause a high voltage drop on the pin with possible malfunctions.
- If no backup supply voltage is available, connect the **V_BCKP** pin to **VCC**.
- Allow all I/O including UART and other interfaces to float or connect to a high impedance in HW backup mode (V_BCKP supplied when VCC is removed). See the Interfaces section.

Real-time clock (RTC)

The real-time clock (RTC) is driven by a 32-kHz oscillator using an RTC crystal. If VCC is removed while a battery is connected to **V_BCKP**, most of the receiver is switched off leaving the RTC and BBR powered. This operating mode is called Hardware Backup Mode which enables time keeping and all relevant data to be saved to allow a hot or warm start.

4.2.3 NEO-M9L power supply

The NEO-M9L requires a low-noise, low-dropout voltage, and a very low source impedance power supply of 3.3 V typically. No inductors or ferrite beads should be used from LDO to the module VCC pin. The peak currents need to be taken into account for the source supplying the LDO for the module.

A power supply fed by 5 V is shown in the figure below. This example circuit is intended only for the module supply.

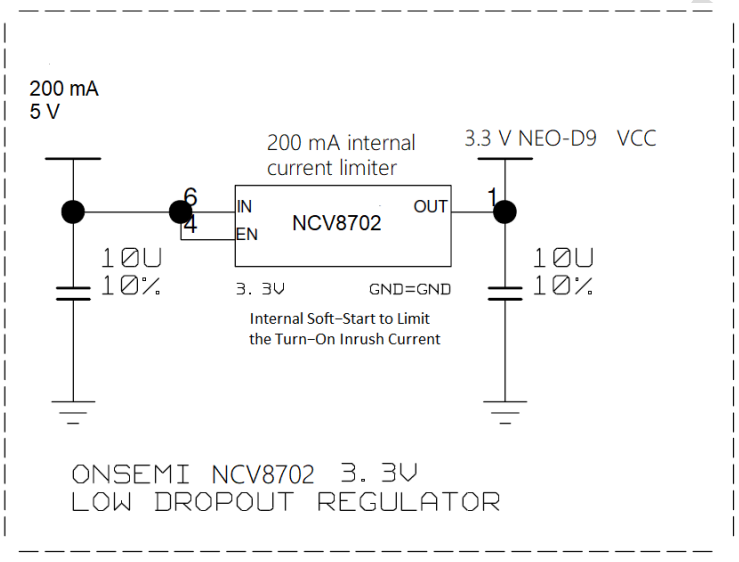


Figure 30: NEO-M9L power supply

4.3 NEO-M9L minimal design

The minimal electrical circuit for NEO-M9L operation using the UART1 interface is shown in Figure 31 below.

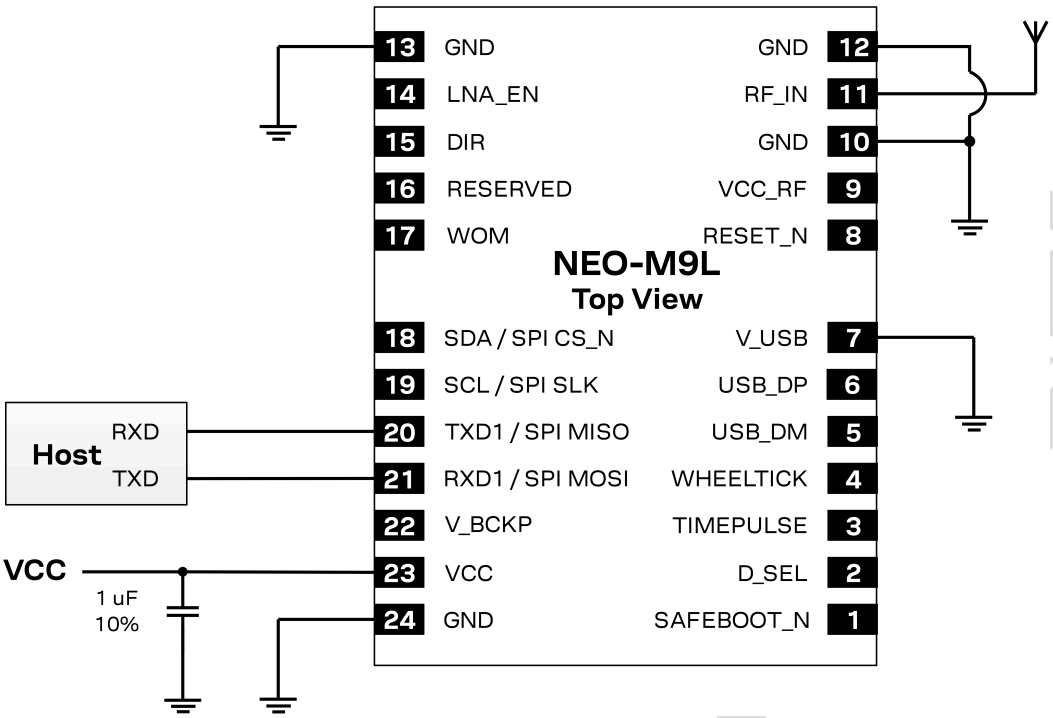


Figure 31: Minimal NEO-M9L design

For a minimal design with the NEO-M9L GNSS modules, the following functions and pins should be considered:

- Connect the power supply to VCC and V_BCKP.
- If hot or warm start operations are needed, connect a backup battery to V_BCKP.
- If USB is not used connect V_USB to ground.
- Ensure an optimal ground connection to all ground pins of the NEO-M9L GNSS module.
- Choose the required serial communication interfaces (UART, USB, SPI or I2C) and connect the appropriate pins to your application. If SPI is used D_SEL must be connected to ground.
- SAFEBOOT_N pin can be used to enable future firmware update.

4.4 WT and DIR interface example

This section shows an example design for interfacing the WT and DIR pins.

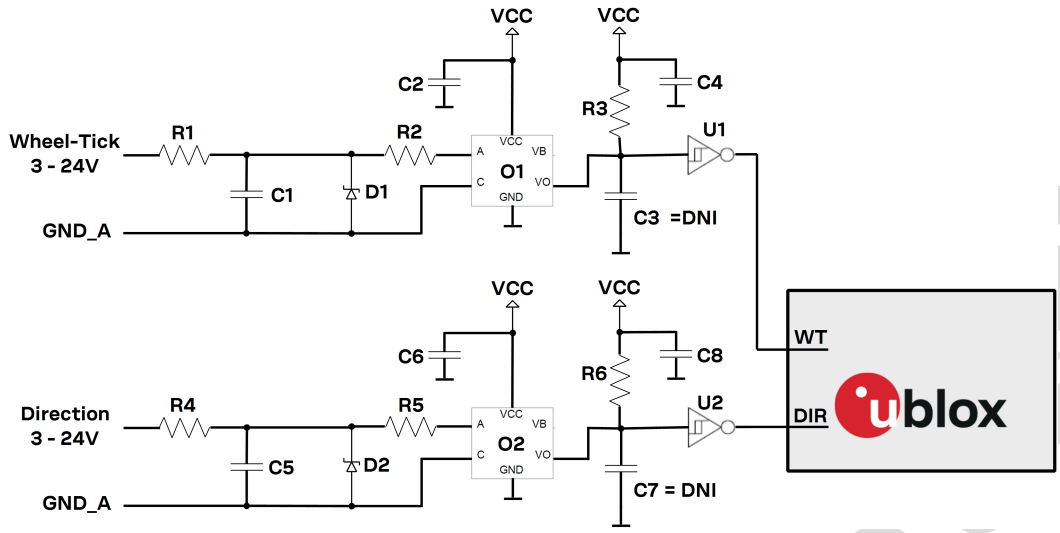


Figure 32: Example WT and DIR design

ID	Description
R1	RES THICK FILM CHIP 0603 3K3 5% 0.1W
R2	RES THICK FILM CHIP 0603 470R 5% 0.1W
R3	RES THICK FILM CHIP 0603 910R 5% 0.1W
R4	RES THICK FILM CHIP 0603 3K3 5% 0.1W
R5	RES THICK FILM CHIP 0603 470R 5% 0.1W
R6	RES THICK FILM CHIP 0603 910R 5% 0.1W
C1	CAP CER X7R 0603 1N 10% 25V
C2	CAP CER X7R 0603 100N 10% 10V
C3	CAP CER X7R 0603 22N 10% 10V
C4	CAP CER X7R 0603 100N 10% 10V
C5	CAP CER X7R 0603 1N 10% 25V
C6	CAP CER X7R 0603 100N 10% 10V
C7	CAP CER X7R 0603 22N 10% 10V
C8	CAP CER X7R 0603 100N 10% 10V
D1	VOLTAGE REGULATOR DIODE FAIRCHILD BZX84 SOT23 6V2 0.2A
D2	VOLTAGE REGULATOR DIODE FAIRCHILD BZX84 SOT23 6V2 0.2A
O1	OPTOCOUPLER LVTTL/LVCMOS COMPATIBLE AVAGO HCPL-070L-000E SO8
O2	OPTOCOUPLER LVTTL/LVCMOS COMPATIBLE AVAGO HCPL-070L-000E SO8
U1	TINY LOGIC UHS INVERTER WITH SCHMITT TRIGGER FAIRCHILD NC7SZ14 SOT23-5
U2	TINY LOGIC UHS INVERTER WITH SCHMITT TRIGGER FAIRCHILD NC7SZ14 SOT23-5

Table 35: WT and DIR example

4.5 Antenna

GNSS signals operate with very low signal levels, ranging from -130 dBm to about -167 dBm. This is already a quite challenging parameter for GNSS receiver performance. In addition to that out-of-band interferers such as GSM, CDMA, WCDMA, LTE, Wi-Fi or Bluetooth wireless systems have a much higher signal level. This makes the receiver layout even more challenging.

Performance of the GNSS antenna and the related front-end RF signal path components have a decisive impact on the performance of the overall GNSS application.

The NEO-M9L has an internal DC block and 50 Ω impedance matching for GNSS signal input, so there is no need to add these components to the RF path.

The antenna must have sufficient bandwidth to receive all needed GNSS constellations[2], and 50 Ω impedance at GNSS frequencies.

4.5.1 Antenna design with passive antenna

A design using a passive antenna requires attention to the layout of the RF section. Typically, a passive antenna is located near electronic components; therefore, care should be taken to reduce electrical noise that may interfere with the antenna performance. The antenna must have sufficient bandwidth to receive all needed GNSS constellations. Also, the antenna should have a suitable ground plane to achieve good performance. Passive antennas do not require a DC bias voltage and can be directly connected to the RF input pin RF_IN.

The example circuit in Figure 33 shows a setup for a design with a good passive GNSS antenna.

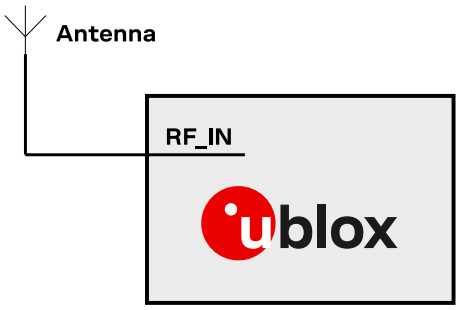


Figure 33: Passive antenna example circuit

4.5.2 Antenna design with external LNA or active antenna

Passive antenna may be inadequate if the antenna gain is low or if the antenna must be placed far away from the module. For such designs the NEO-M9L module supports controlling an additional external LNA or an active antenna. The antenna control signal pin LNA_EN is active high to enable the external LNA or the active antenna when the receiver needs GNSS signal. If needed, the VCC_RF output pin can be used to supply filtered DC bias voltage to these external components. In this case, the supply voltage of the NEO-M9L module must match the external components' working voltage (for example, 3.3 V). If the VCC_RF voltage does not match with the supply voltage of the active antenna, use a filtered external supply.

The external LNA or the active antenna with an integrated LNA requires a power supply that contributes an additional current of typically 5 to 20 mA to the system's power consumption budget.

Active antennas for GNSS application are usually powered through a DC bias on the RF cable. A simple bias-T as in Figure 34 can be used to add this DC current to the RF signal line. The bias voltage can be supplied either by the VCC_RF pin of the module or by an external antenna supply.

[2] GNSS L-band frequencies: 1559...1563 MHz; 1573...1578 MHz; 1598...1606 MHz;

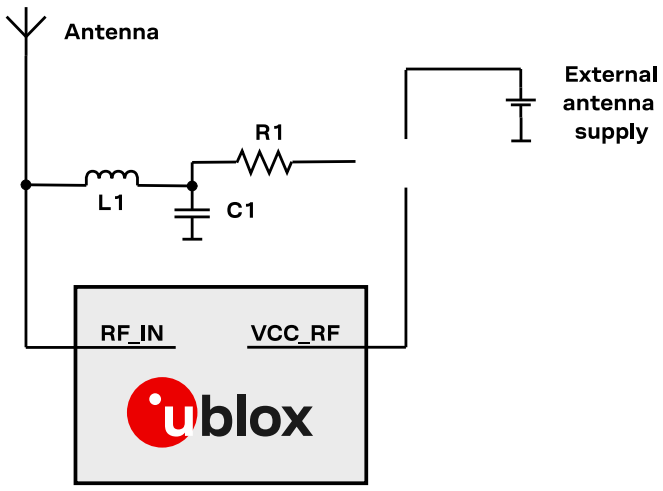


Figure 34: NEO-M9L active antenna example circuit (bias-T)

Part	Recommended value	Description
C1	X7R 100 nF 10% 16 V	Bias-T capacitor removes high frequency noise coming from the DC supply.
L1	27 nH, impedance > 500 Ω at GNSS frequency, rated current > 300 mA	Bias-T inductor isolates RF from the DC path.
R1	10 Ω 10% 0.25 W	Current limiter series resistor in the event of a short circuit.

Table 36: Components in active antenna example circuit (bias-T)

Design recommendations:

- If the application has a passive antenna with sufficient gain but the antenna is located far away from the module, an additional external LNA must be placed close to the passive antenna.
- If the application needs an active antenna, it is recommended to use an OEM active antenna module meeting u-blox's specification.
- A suitable ground plane is required for the antenna to achieve good performance.
- Active antenna may have strong gain. For maximum external gain, see the Data sheet [1] in Related documents.
- A series current limiting resistor is required to prevent short circuits destroying the bias-t inductor.
- If the VCC_RF voltage of the NEO-M9L module does not match the supply voltage of the active antenna, use a filtered external supply instead of the VCC_RF.
- The antenna circuit should include filtering to ensure adequate protection from the nearby transmitters' interface. Take care in designing the circuit of antennas placed close to cellular or Wi-Fi transmitting antennas.
- An ESD protection diode should also be connected to the input.

It is recommended to use active current limiting. If active current limiting is not used, take into account the following important points:

☞ The bias-t inductor and current limiting resistor must be selected to be reliable with a short circuit on the antenna feed. In cases where the module supplies the voltage via VCC_RF, a higher value resistor will be needed to ensure the module supply inductor is protected. The current should be limited to below 150 mA at the module supply voltage under short circuit conditions. For example, a value of 19 Ω or more is required at a module supply of 3.3 V to limit short circuit current to 150 mA. The DC resistance of the bias-t inductor is assumed to be 1-2 Ω and the module internal feed inductor is assumed to be 1.2 Ω.

☞ The power dissipation in the resistor and inductor needs to be taken into account based on the supply voltage and the short circuit current.

⚠ The NEO-M9L conformity to Radio Equipment Directive (RED) 2014/53/EU has been assessed with active GNSS antenna B3G02G-S3-01-A. Similar antenna or RF front-end characteristics should be used for the module.

To achieve the best performance, u-blox recommends using an active antenna or an external LNA with this module:

Parameter	Specification	
Active antenna recommendations	Minimum gain[3]	15 dB
	Maximum gain[3]	30 dB
	Noise figure	<2 dB
Antenna element specification	L1 band antenna gain [4]	1559 - 1606 MHz: 3 dBic typ.

Table 37: Antenna specifications for NEO-M9L modules

4.6 EOS/ESD precautions

⚠ To avoid overstress damage during production or in the field it is essential to observe strict EOS/ESD/EMI handling and protection measures.

⚠ To prevent overstress damage at the RF_IN of your receiver, never exceed the maximum input power as specified in the applicable data sheet [1].

When integrating GNSS receivers into wireless systems, pay special attention to electromagnetic and voltage susceptibility issues. Wireless systems include components which can produce Electrostatic Discharge (ESD), Electrical Overstress (EOS) and Electro-Magnetic Interference (EMI). CMOS devices are more sensitive to such influences because their failure mechanism is defined by the applied voltage, whereas bipolar semiconductors are more susceptible to thermal overstress. The following design guidelines help in designing robust yet cost-effective solutions.

4.6.1 ESD protection measures

⚠ GNSS receivers are sensitive to Electrostatic Discharge (ESD). Special precautions are required when handling. Most defects caused by ESD can be prevented by following strict ESD protection rules for production and handling. When implementing passive antenna patches or external antenna connection points, then additional ESD measures as shown in the figure below can also avoid failures in the field.

3 Including passive losses (filters, cables, connectors etc.)

4 Measured with a ground plane d=150 mm

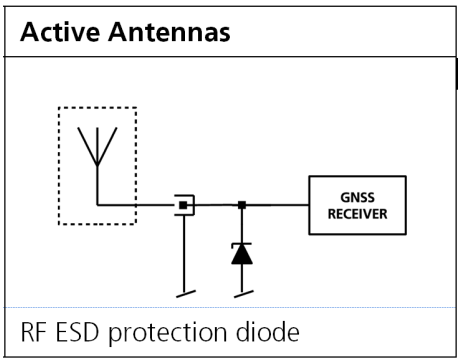


Figure 35: RF ESD precautions

4.6.2 EOS precautions

Electrical overstress (EOS) usually describes situations when the maximum input power exceeds the maximum specified ratings. EOS failure can happen if RF emitters are close to a GNSS receiver or its antenna. EOS causes damage to the chip structures. If the RF_IN is damaged by EOS, it is hard to determine whether the chip structures have been damaged by ESD or EOS.

EOS protection measures as shown in the figure below are recommended for any designs combining wireless communication transceivers (e.g. GSM, GPRS) and GNSS in the same design or in close proximity.

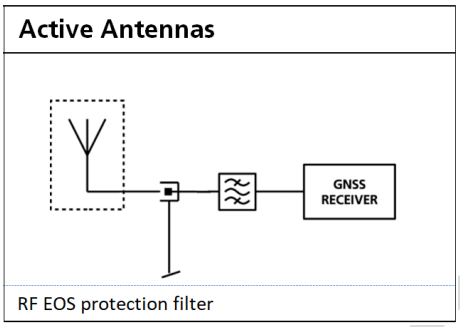


Figure 36: Active antenna EOS protection

4.6.3 Safety precautions

The NEO-M9L must be supplied by an external limited power source in compliance with the clause 2.5 of the standard IEC 60950-1. In addition to external limited power source, only Separated or Safety Extra-Low Voltage (SELV) circuits are to be connected to the module including interfaces and antennas.

☞ For more information about SELV circuits see section 2.2 in Safety standard IEC 60950-1.

4.7 Electromagnetic interference on I/O lines

Any I/O signal line with a length greater than approximately 3 mm can act as an antenna and may pick up arbitrary RF signals transferring them as noise into the receiver. This specifically applies to unshielded lines, in which the corresponding GND layer is remote or missing entirely, and lines close to the edges of the printed circuit board.

If, for example, a cellular signal radiates into an unshielded high-impedance line, it is possible to generate noise in the order of volts and not only distort receiver operation but also damage it

permanently. Another type of interference can be caused by noise generated at the PIO pins that emits from unshielded I/O lines. Receiver performance may be degraded when this noise is coupled into the GNSS antenna.

EMI protection measures are particularly useful when RF emitting devices are placed next to the GNSS receiver and/or to minimize the risk of EMI degradation due to self-jamming. An adequate layout with a robust grounding concept is essential in order to protect against EMI.

☞ Intended Use: In order to mitigate any performance degradation of a radio equipment under EMC disturbance, system integration shall adopt appropriate EMC design practice and not contain cables over three meters on signal and supply ports.

4.7.1 General notes on interference issues

Received GNSS signal power at the antenna is very low. At the nominal received signal strength (-128 dBm) it is below the thermal noise floor of -111 dBm. Due to this fact, a GNSS receiver is susceptible to interference from nearby RF sources of any kind. Two cases can be distinguished:

- Out-of-band interference: Typically any kind of wireless communications system (e.g. LTE, GSM, CDMA, 3G, WLAN, Bluetooth, etc.) may emit its specified maximum transmit power in close proximity to the GNSS receiving antenna, especially if such a system is integrated with the GNSS receiver. Even at reasonable antenna selectivity, destructive power levels may reach the RF input of the GNSS receiver. Also, larger signal interferers may generate intermodulation products inside the GNSS receiver front-end that fall into the GNSS band and contribute to in-band interference.
- In-band interference: Although the GNSS band is kept free from intentional RF signal sources by radio-communications standards, many devices emit RF power into the GNSS band at levels much higher than the GNSS signal itself. One reason is that the frequency band above 1 GHz is not well regulated with regards to EMI, and even if permitted, signal levels are much higher than GNSS signal power. Notably, all types of digital equipment, such as PCs, digital cameras, LCD screens, etc. tend to emit a broad frequency spectrum up to several GHz of frequency. Also wireless transmitters may generate spurious emissions that fall into GNSS band.

As an example, GSM uses power levels of up to 2 W (+33 dBm). The absolute maximum power input at the RF input of the GNSS receiver can be +15 dBm. The GSM specification allows spurious emissions for GSM transmitters of up to +36 dBm, while the GNSS signal is less than -128 dBm. By simply comparing these numbers it is obvious that interference issues must be seriously considered in any design of a GNSS receiver. Different design goals may be achieved through different implementations:

- The primary focus is to prevent damaging the receiver from large input signals. Here the GNSS performance under interference conditions is not important and suppression of the signal is permitted. It is sufficient to just observe the maximum RF power ratings of all of the components in the RF input path.
- GNSS performance must be guaranteed even under interference conditions. In such a case, not only the maximum power ratings of the components in the receiver RF path must be observed. Further, non-linear effects like gain compression, NF degradation (desensitization) and intermodulation must be analyzed.

☞ Pulsed interference with a low-duty cycle such as GSM may be destructive due to the high peak power levels.

4.7.2 In-band interference mitigation

With in-band interference, the signal frequency is very close to the GNSS frequency. Such interference signals are typically caused by harmonics from displays, micro-controller operation, bus systems, etc. Measures against in-band interference include:

- Maintaining a good grounding concept in the design
- Shielding
- Layout optimization
- Low-pass filtering of noise sources, e.g. digital signal lines
- Remote placement of the GNSS antenna, far away from noise sources
- Adding an LTE, CDMA, GSM, WCDMA, BT band-pass filter before antenna

4.7.3 Out-of-band interference

Out-of-band interference is caused by signal frequencies that are different from the GNSS carrier frequency. The main sources are wireless communication systems such as LTE, GSM, CDMA, WCDMA, Wi-Fi, BT, etc.

Measures against out-of-band interference include maintaining a good grounding concept in the design and adding a GNSS band-pass filter into the antenna input line to the receiver.

For GSM applications, such as typical handset design, an isolation of approximately 20 dB can be reached with careful placement of the antennas. If this is insufficient, an additional SAW filter is required on the GNSS receiver input to block the remaining GSM transmitter energy.

4.8 Layout

This section details layout and placement requirements of the NEO-M9L module.

4.8.1 Placement

GNSS signals at the surface of the Earth are below the thermal noise floor. A very important factor in achieving maximum GNSS performance is the placement of the receiver on the PCB. The placement used may affect RF signal loss from antenna to receiver input and enable interference into the sensitive parts of the receiver chain, including the antenna itself. When defining a GNSS receiver layout, the placement of the antenna with respect to the receiver, as well as grounding, shielding and interference from other digital devices are crucial issues and need to be considered very carefully.

Signal loss on the RF connection from antenna to receiver input must be minimized as much as possible. Hence, the connection to the antenna must be kept as short as possible.

Ensure that RF critical circuits are clearly separated from any other digital circuits on the system board. To achieve this, position the receiver digital part closer to the digital section of the system PCB and have the RF section and antenna placed as far as possible away from the other digital circuits on the board.

A proper GND concept shall be followed: The RF section shall not be subject to noisy digital supply currents running through its GND plane.

4.8.2 Thermal management

During design-in do not place the receiver near sources of heating or cooling. The receiver oscillator is sensitive to sudden changes in ambient temperature which can adversely impact satellite signal tracking. Sources can include co-located power devices, cooling fans or thermal conduction via the PCB. Take into account the following questions when designing in the receiver.

- Is the receiver placed away from heat sources?
- Is the receiver placed away from air-cooling sources?
- Is the receiver shielded by a cover/case to prevent the effects of air currents and rapid environmental temperature changes?

⚠ High temperature drift and air vents can affect the GNSS performance. For best performance, avoid high temperature drift and air vents near the receiver.

4.8.3 Package footprint, copper and paste mask

Copper and solder mask dimensioning recommendations for the NEO-M9L module packages are provided in this section.

☞ These are recommendations only and not specifications. The exact copper, solder and paste mask geometries, distances, stencil thickness and solder paste volumes must be adapted to the specific production processes (e.g. soldering etc.) of the customer.

Refer to the applicable data sheet [1] for the mechanical dimensions.

4.8.3.1 Footprint

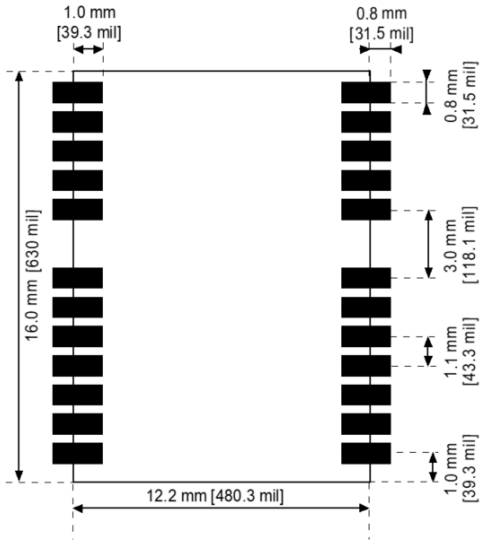


Figure 37: NEO-M9L suggested footprint (i.e. copper mask)

4.8.3.2 Paste mask

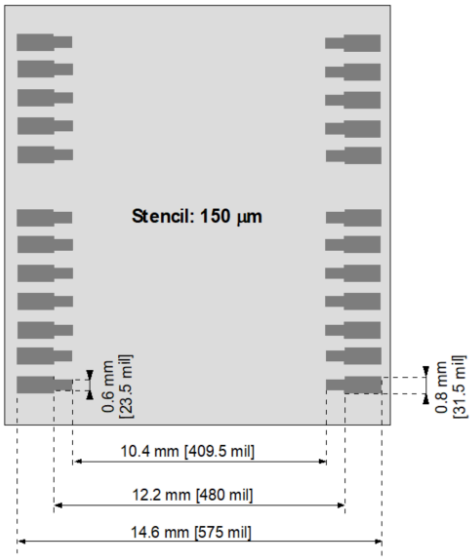


Figure 38: NEO-M9L suggested paste mask

4.8.4 Layout guidance

The presented layout guidance reduces the risk of performance issues at design level.

4.8.4.1 RF In trace

The RF In trace has to work in the middle L-band frequencies.

A grounded co-planar RF trace is recommended as it provides the maximum shielding from noise with adequate vias to the ground layer.

Focus on these layout guidelines:

- RF input co-planar waveguide referenced to layer 2, matched to 50 Ω.
- Place RF input matching component close to RF_IN pin.
- Use vias to 3D grounding at ground plane.
- No stubs must occur on the RF input trace.

The RF trace must be shielded by vias to ground along the entire length of the trace and the NEO-M9L RF_IN pad should be surrounded by vias as shown in the figure below.

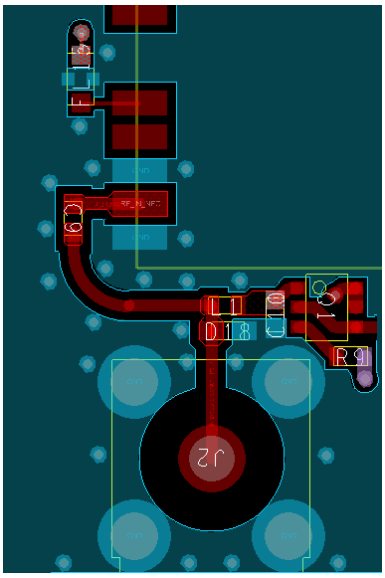


Figure 39: RF input trace

The RF_IN trace on the top layer should be referenced to a suitable ground layer.

4.8.4.2 VCC pad

The VCC pad for the NEO-M9L module needs to have as low an impedance as possible with large vias to the lower power layer of the PCB. The VCC pad needs a large pad and the decoupling capacitor must be placed as close as possible. This is shown in the figure below.

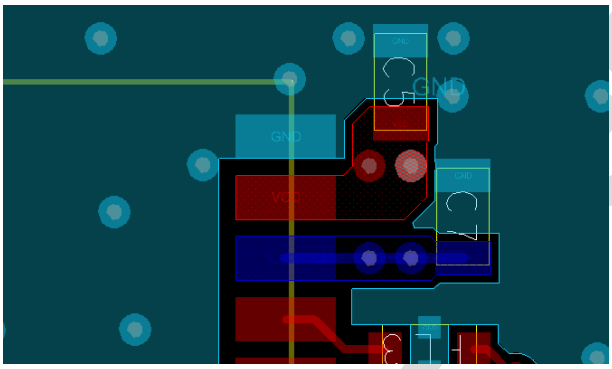


Figure 40: VCC pad

4.9 Design guidance

4.9.1 General considerations

Check power supply requirements and schematic:

- Is the power supply voltage within the specified range and noise-free?
- If USB is not used, connect the V_USB pin to ground.
- For USB devices: Is the V_USB voltage within the specified range?
- It is recommended to have a separate LDO for V_USB that is enabled by the module VCC. This is to comply with the USB self-powered specification.
- If USB is used, is there a 1 uF capacitor right near the V_USB pin? This is just for the V_USB pin.
- Is there a 1 uF cap right next to the module VCC pin?

- Compare the peak current consumption of the NEO-M9L GNSS module with the specification of your power supply.
- GNSS receivers require a stable power supply. Avoid series resistance (less than 0.2 Ω) in your power supply line (the line to VCC) to minimize the voltage ripple on VCC. See the NEO-M9L Power supply section in the Design chapter for more information on the power supply requirements.
- Allow all I/O to Float/High impedance (High-Z) when VCC is not applied.

4.9.2 Backup battery

Check backup supply requirements and schematic:

- For achieving a minimal time to first fix (TTFF) after a power down (warm starts, hot starts), make sure to connect a backup battery to V_BCKP.
- Verify that your battery backup supply can provide the battery backup current specified in the applicable data sheet.
- Allow all I/O including UART and other interfaces to Float/High impedance in HW backup mode (battery backup connected with VCC removed).

4.9.3 RF front-end circuit options

The first stages of the signal processing chain are crucial to the overall receiver performance.

When an RF input connector is employed this can provide a conduction path for harmful or destructive electrical signals. If this is a likely factor the RF input should be protected accordingly.

Additional points on the RF input

- What is the expected quality of the signal source (antenna)?
- What is the external active antenna signal power?
- What is the bandwidth and filtering of the external active antenna?
- Does the external antenna and any optional filtering components meet the group delay variation requirements?

Are destructive RF power levels expected to reach the RF input? Is interference from wireless transmitters expected?

- What are the characteristics of these signals (duty cycle, frequency range, power range, spectral purity)?
- What is the expected GNSS performance under interference conditions?

Is there a risk of RF input exposure to excessive ESD stress?

- In the field: Can the user access the antenna connector?
- PCB / system assembly: Is there risk that statically charged parts (e.g. patch antennas) may be discharged through the RF input?

The following subsections provide several options addressing the various questions above:

☞ In some applications, such as cellular transceivers, interference signals may exceed the maximum power rating of the RF_IN input. To avoid device destruction use of external input protection is mandatory.

☞ During assembly of end-user devices which contain passive patch antennas, an ESD discharge may occur during production when pre-charged antennas are soldered to the GNSS receiver board. In such cases, use of external protection in front of RF_IN is mandatory to avoid device destruction.

ESD discharge cannot be avoided during assembly and / or field use. Note that SAW filters are susceptible to ESD damage. To provide additional robustness an ESD protection diode may be placed at the antenna RF connector to GND.

4.9.4 Antenna/RF input

Check RF input requirements and schematic:

- Make sure the antenna is not placed close to noisy parts of the circuitry and does not face any other noisy elements (for example microcontroller, display).
- ESD protection on the RF input is mandatory.
- Possible external antenna bias-t inductor must be L-band frequency selected with high impedance in the GNSS band.
- Ensure RF trace is tuned for 50 Ω to ensure proper GNSS bandwidth.

4.9.5 Ground pads

Ensure the ground pads of the module are connected to ground.

If a patch type antenna is used, a sufficient antenna ground plane is required.

4.9.6 Schematic design

For a **minimal design** with the NEO-M9L GNSS modules, consider the following functions and pins:

- Connect the power supply to VCC and V_BCKP.
- V_USB: If USB is used it is recommended V_USB is to be powered as per USB self-powered mode specification.
- If USB is not used connect V_USB to ground.
- Ensure an optimal ground connection to all ground pins of the NEO-M9L GNSS module.
- Choose the required serial communication interfaces (UART, USB, SPI or I2C) and connect the appropriate pins to your application.
- If you need hot or warm start in your application, connect a backup battery to V_BCKP.

4.9.7 Layout design-in guideline

- Is the receiver placed away from heat sources?
- Is the receiver placed away from air-cooling sources?
- Is the receiver shielded by a cover/case to prevent the effects of air currents and rapid environmental temperature changes?
- Is the receiver placed as recommended in the Layout and Layout guidance?
- Assure a low serial resistance on the VCC power supply line (choose a line width > 400 um).
- Keep the power supply line as short as possible.
- Add a ground plane underneath the module to reduce interference. This is especially important for the RF input line.
- For improved shielding, add as many vias as possible around the micro strip/co-planar waveguide, around the serial communication lines, underneath the module, etc.

5 Product handling

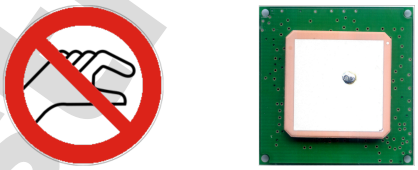
5.1 ESD handling precautions

NEO-M9L contains highly sensitive electronic circuitry and is an Electrostatic Sensitive Device (ESD). Observe precautions for handling! Failure to observe these precautions can result in severe damage to the GNSS receiver!

- Unless there is a galvanic coupling between the local GND (i.e. the work table) and the PCB GND, then the first point of contact when handling the PCB must always be between the local GND and PCB GND.
- Before mounting an antenna patch, connect ground of the device.

- When handling the RF pin, do not come into contact with any charged capacitors and be careful when contacting materials that can develop charges (e.g. patch antenna ~10 pF, coax cable ~50-80 pF/m or soldering iron).

- To prevent electrostatic discharge through the RF input, do not touch any exposed antenna area. If there is any risk that such exposed antenna area is touched in non-ESD protected work area, implement proper ESD protection measures in the design.



- When soldering RF connectors and patch antennas to the receiver's RF pin, make sure to use an ESD-safe soldering iron (tip)

5.2 Soldering

Soldering paste

Use of "no clean" soldering paste is highly recommended, as it does not require cleaning after the soldering process. The paste in the example below meets these criteria.

- Soldering paste: OM338 SAC405 / Nr.143714 (Cookson Electronics)
- Alloy specification: Sn 95.5/ Ag 4/ Cu 0.5 (95.5% tin/ 4% silver/ 0.5% copper)
- Melting temperature: 217 °C
- Stencil thickness: The exact geometry, distances, stencil thicknesses and solder paste volumes must be adapted to the customer's specific production processes (e.g. soldering).

Reflow soldering

A convection-type soldering oven is highly recommended over the infrared-type radiation oven. Convection-heated ovens allow precise control of the temperature, and all parts will heat up evenly, regardless of material properties, thickness of components and surface color.

As a reference, see “IPC-7530 Guidelines for temperature profiling for mass soldering (reflow and wave) processes”, published in 2001.

Preheat phase

During the initial heating of component leads and balls, residual humidity will be dried out. Note that the preheat phase does not replace prior baking procedures.

- Temperature rise rate: max 3 °C/s. If the temperature rise is too rapid in the preheat phase, excessive slumping may be caused
- Time: 60 – 120 s. If the preheat is insufficient, rather large solder balls tend to be generated. Conversely, if performed excessively, fine balls and large balls will be generated in clusters
- End temperature: 150 – 200 °C. If the temperature is too low, non-melting tends to be caused in areas containing large heat capacity

Heating - reflow phase

The temperature rises above the liquidus temperature of 217 °C. Avoid a sudden rise in temperature as the slump of the paste could become worse.

For professional grade modules

- Limit time above 217 °C liquidus temperature: 40 – 60 s
- Peak reflow temperature: 245 °C

For automotive grade modules

- Limit time above 217 °C liquidus temperature: 60 – 150 s
- Peak reflow temperature: 250 °C

Cooling phase

A controlled cooling prevents negative metallurgical effects of the solder (solder becomes more brittle) and possible mechanical tensions in the products. Controlled cooling helps to achieve bright solder fillets with a good shape and low contact angle.

For professional grade modules

- Temperature fall rate: max 4 °C/s

For automotive grade modules

- Temperature fall rate: max 6 °C/s

☞ To avoid falling off, the modules should be placed on the topside of the motherboard during soldering.

The final soldering temperature chosen at the factory depends on additional external factors such as the choice of soldering paste, size, thickness and properties of the base board, etc. Exceeding the maximum soldering temperature in the recommended soldering profile may permanently damage the module.

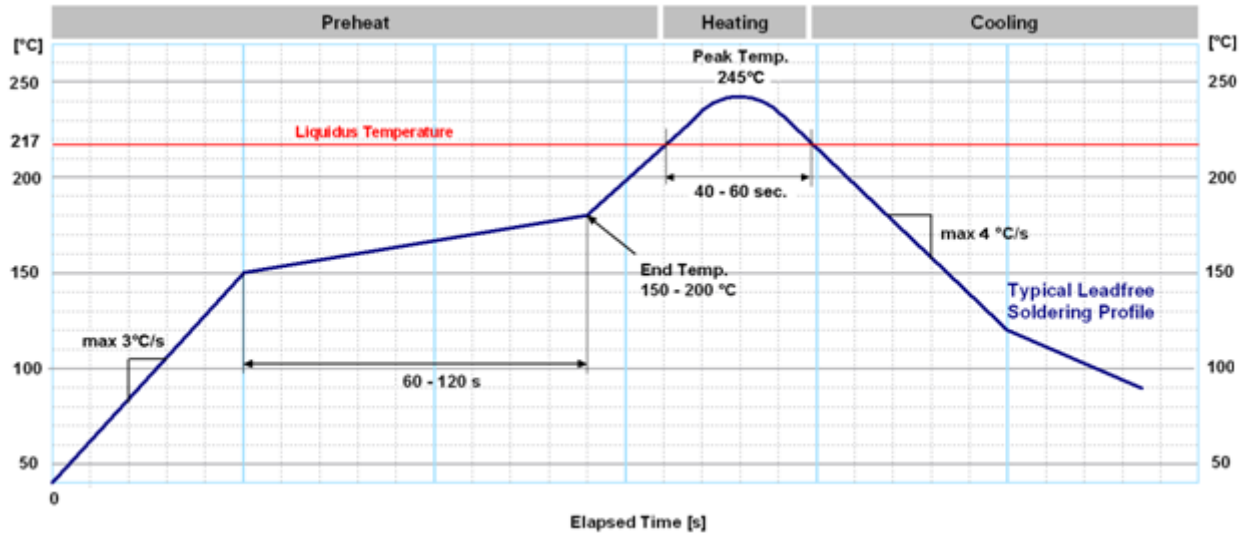


Figure 41: Soldering profile for professional grade NEO-M9L

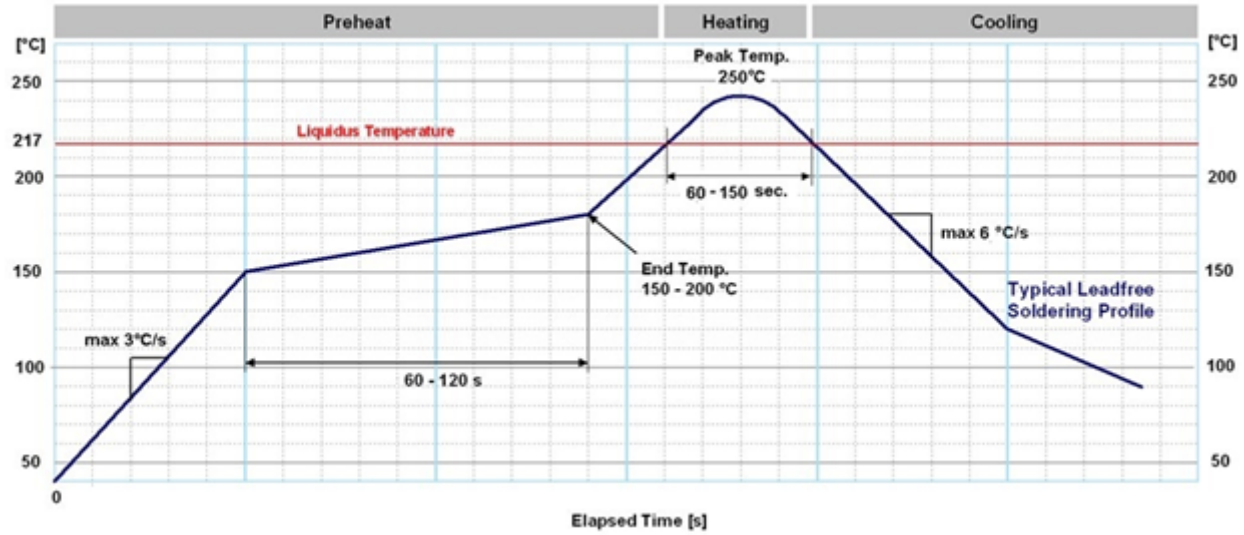


Figure 42: Soldering profile for automotive grade NEO-M9L

☞ Modules **must not** be soldered with a damp heat process.

Optical inspection

After soldering the module, consider optical inspection.

Cleaning

☞ Do not clean with water, solvent, or ultrasonic cleaner:

- Cleaning with water will lead to capillary effects where water is absorbed into the gap between the baseboard and the module. The combination of residues of soldering flux and encapsulated water leads to short circuits or resistor-like interconnections between neighboring pads.
- Cleaning with alcohol or other organic solvents can result in soldering flux residues flowing underneath the module, into areas that are not accessible for post-cleaning inspections. The solvent will also damage the sticker and the printed text.

- Ultrasonic cleaning will permanently damage the module, in particular the quartz oscillators.

The best approach is to use a "no clean" soldering paste and eliminate the cleaning step after the soldering.

Repeated reflow soldering

☞ Repeated reflow soldering processes or soldering the module upside down are not recommended.

A board that is populated with components on both sides may require more than one reflow soldering cycle. In such a case, the process should ensure the module is only placed on the board submitted for a single final upright reflow cycle. A module placed on the underside of the board may detach during a reflow soldering cycle due to lack of adhesion.

The module can also tolerate an additional reflow cycle for rework purposes.

Wave soldering

Base boards with combined through-hole technology (THT) components and surface-mount technology (SMT) devices require wave soldering to solder the THT components. Only a single wave soldering process is encouraged for boards populated with modules.

Rework

We do not recommend using a hot air gun because it is an uncontrolled process and can damage the module.

⚠ Use of a hot air gun can lead to overheating and severely damage the module. Always avoid overheating the module.

After the module is removed, clean the pads before reapplying solder paste, placing and reflow soldering a new module.

⚠ Never attempt a rework on the module itself, e.g. by replacing individual components. Such actions will immediately void the warranty.

Conformal coating

Certain applications employ a conformal coating of the PCB using HumiSeal® or other related coating products. These materials affect the RF properties of the GNSS module and it is important to prevent them from flowing into the module. The RF shields do not provide 100% protection for the module from coating liquids with low viscosity. Apply the coating carefully.

☞ Conformal coating of the module will void the warranty.

Casting

If casting is required, use viscose or another type of silicon pottant. The OEM is strongly advised to qualify such processes in combination with the module before implementing this in the production.

☞ Casting will void the warranty.

Grounding metal covers

Attempts to improve grounding by soldering ground cables, wick or other forms of metal strips directly onto the EMI covers is done at the customer's own risk. The numerous ground pins should be sufficient to provide optimum immunity to interferences and noise.

☞ u-blox makes no warranty for damages to the module caused by soldering metal cables or any other forms of metal strips directly onto the EMI covers.

Use of ultrasonic processes

Some components on the module are sensitive to ultrasonic waves. Use of any ultrasonic processes (cleaning, welding etc.) may cause damage to the GNSS receiver.

☞ u-blox offers no warranty against damages to the module caused by ultrasonic processes.

5.3 Tapes

Figure 43 shows the feed direction and illustrates the orientation of the NEO-M9Ls on the tape:

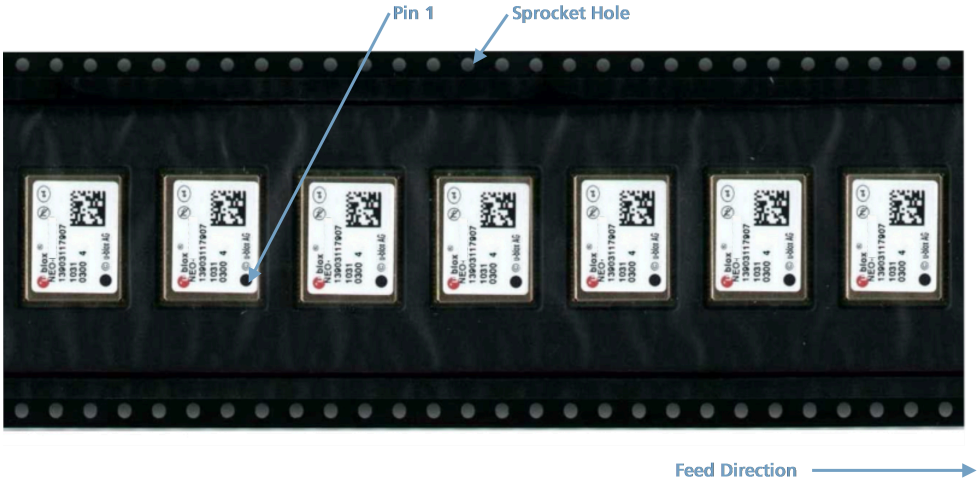


Figure 43: Orientation of NEO-M9L on the tape

The dimensions of the tapes for NEO-M9L are specified in Figure 44 (measurements in mm).

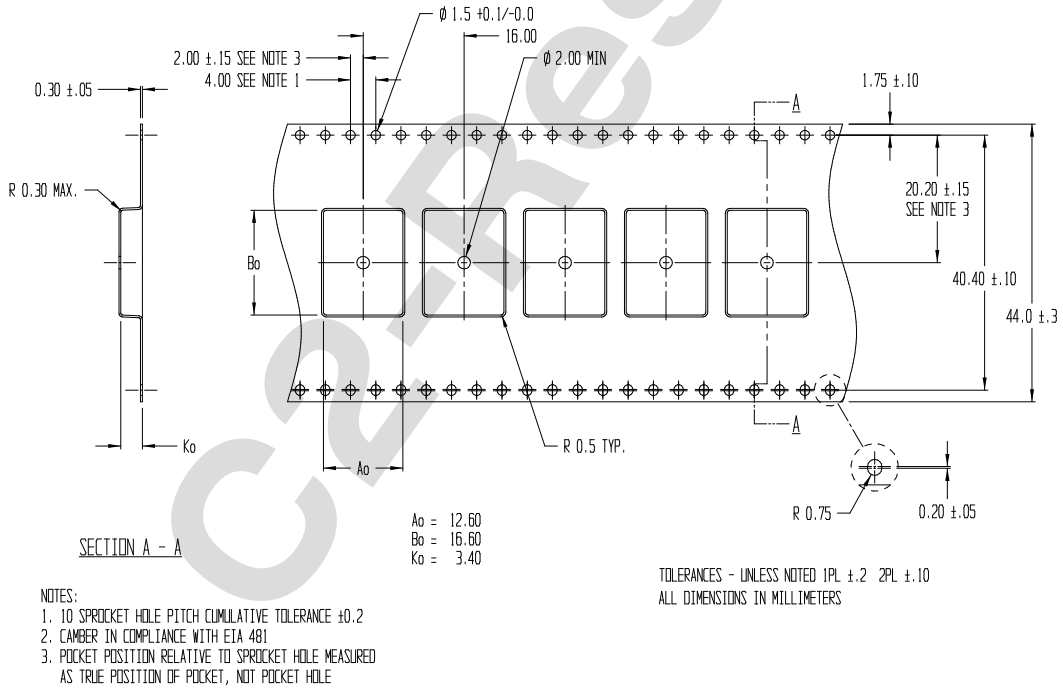


Figure 44: NEO-M9L tape dimensions (mm)

5.4 Reels

The NEO-M9L receivers are deliverable in quantities of 250 pieces on a reel. The receivers are shipped on reel type B, as specified in the u-blox Package Information Guide [3].

5.5 Moisture sensitivity levels

The moisture sensitivity level (MSL) for NEO-M9L is specified in the table below.

Package	MSL level
LCC	4 (NEO-M9L-20B), 3 (NEO-M9L-01A and NEO-M9L-20A),

Table 38: MSL level

☞ For MSL standard see IPC/JEDEC J-STD-020, which can be downloaded from www.jedec.org.

☞ For more information regarding moisture sensitivity levels, labeling, storage and drying, see the u-blox Package Information Guide [3].

Appendix

A Glossary

Abbreviation	Definition
ACH	Advanced calibration handling
ADR	Automotive dead reckoning
ANSI	American National Standards Institute
ARP	Antenna reference point
BeiDou	Chinese navigation satellite system
BBR	Battery-backed RAM
CDMA	Code-division multiple access
CRP	Configurable reference point
ECDSA	Elliptic Curve Digital Signature Algorithm
EMC	Electromagnetic compatibility
EMI	Electromagnetic interference
EOS	Electrical overstress
EPA	Electrostatic protective area
ESD	Electrostatic discharge
Galileo	European navigation satellite system
GLONASS	Russian navigation satellite system
GND	Ground
GNSS	Global navigation satellite system
GPS	Global Positioning System
GSM	Global System for Mobile Communications
I2C	Inter-integrated circuit bus
IEC	International Electrotechnical Commission
IMU	Inertial measurement unit
IRP	Inertial measurement unit reference point
PCB	Printed circuit board
QZSS	Quasi-Zenith Satellite System
RF	Radio frequency
SBAS	Satellite-based Augmentation System
SV	Space vehicle, a satellite
TDOP	Time dilution of precision
UBX	u-blox
UDR	Untethered dead reckoning
VRP	Vehicle reference point
WOM	Wake on motion

B Providing odometer data over the software sensor interface

This section describes the steps required for using the software sensor interface to provide odometer data from the host system to the receiver. It also outlines the aspects to be considered in a typical setup.

B.1 Configuration

B.1.1 Disable odometer hardware interface

By default, the CFG-SFODO-DIS_AUTOSW is set to 1 meaning that the receiver will automatically use odometer data provided over the software sensor interface instead of hardware interface.

In some designs, for example after power cycle of the system, the host may not be able to readily provide the data over the software sensor interface. In the meantime, the receiver uses signals on the hardware interface. To avoid any discrepancy, we highly recommend disabling the hardware interface by setting CFG-SFODO-USE_WT_PIN to 0 (in all layers).

This setting configures the receiver firmware to ignore signals on WT_PIN and prevents reading the state of the DIR_PIN. Effectively the receiver will use odometer data provided in the UBX-ESF-MEAS messages on the specified communication port.

If the design includes an IMU sensor directly connected to the UBX DR chip to provide 3-axis acceleration, 3-axis gyroscope, and gyroscope temperature data, the host shall not provide such IMU data over the software sensor interface in a normal use case.

B.1.2 Frequency

Configure the nominal frequency of odometer data by setting CFG-SFODO-FREQUENCY. The recommended frequency is 10 Hz.

If the receiver detects that the frequency of the data is significantly different from the configured value, faults will be reported in the UBX-ESF-STATUS output message. See chapter Sensor data in UBX-ESF-MEAS for more information.

B.1.3 Latency

Configure the latency of odometer data by setting CFG-SFODO-LATENCY. The latency is the time between sampling the measurement by the sensor and making the data available to the receiver. It results from data processing and transfer delays in the data path.

The latency should be as close to a constant value as possible.The latency should be kept to an absolute minimum for real time use of the data in the navigation filter. The latency should not exceed 100 ms.

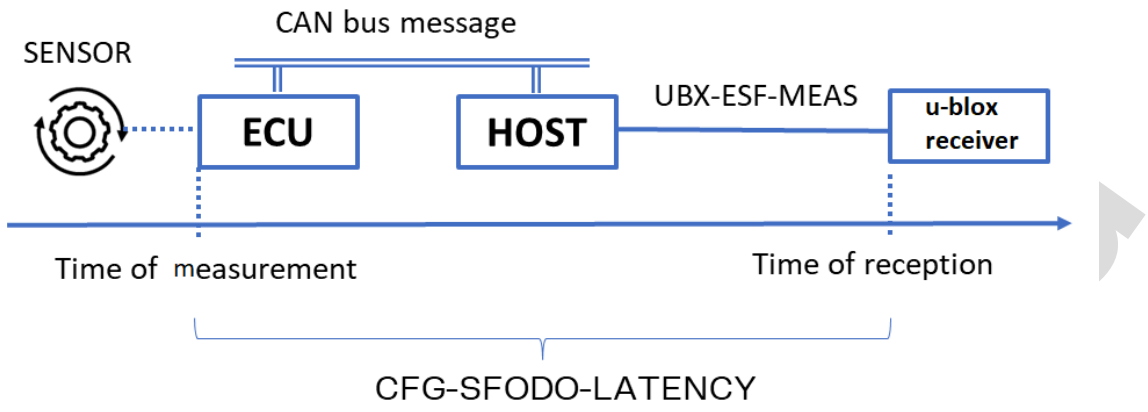


Figure 45: Example of odometer measurement data path

B.1.4 Speed data vs. wheel tick data

Odometer data is typically available either as speed or wheel tick data. If both are available, we recommend providing the one of better quality. Small quantization and dead band together with fast direction sense is preferred. The data should refer as close as possible to the VRP.

It is possible, although not recommended to provide both speed and wheel tick data. If you want the receiver to use speed data instead of wheel tick data, set the CFG-SFODO-USE_SPEED to 1 .

B.1.4.1 Using speed data

The signed vehicle speed data published e.g., on CAN bus is often a good and straightforward choice for dataType = 11 in UBX-ESF-MEAS (see Sensor data types). When only individual wheel speed data is available, a good approximation of the speed at VRP is the average of rear left and rear right wheel speeds.

Some vehicles may suffer from speed dead band, that is, zero speed is reported if the true speed is below a certain threshold speed. Crawling at speed below the threshold goes unnoticed and may lead to significant position offset over time. To reduce the net effect on position error, configure CFG-SFODO-SPEED_BAND. In ideal case the dead band is equal to speed quantization.

If the RMS error of speed measurements is known, set it in CFG-SFODO-QUANT_ERROR.

B.1.4.2 Using wheel tick data

In case the wheel tick data describes the distance traveled at VRP well, it is good for single tick dataType =10. Note that commercial vehicles seldom have a wheel tick sensor in this location.

More often the wheel tick data is available for each individual wheel. When available, provide the rear left wheel ticks (dataType = 8) and rear right wheel ticks (dataType = 9) and set CFG-SFODO-COMBINE_TICKS to 1. Receiver firmware will estimate the wheel tick count at VRP.

Do not count and provide single tick (dataType = 10) as a sum of rear wheel ticks. The individual rear wheel tick counts roll-over independent of each other making the sum unpredictable.

Typically, the available wheel tick data has limited value range and resolution. This means that the wheel tick data rolls over after reaching the maximum value. When the maximum value of the wheel tick counter is known, set it in CFG-SFODO-COUNT_MAX and set CFG-SFODO-DIS_AUTOCOUNTMAX to 1. The automatic maximum count estimator works correctly only if the actual maximum value equals 2^N-1, where N is positive integer value. In certain CAN

implementations the highest counter values denote error codes and effectively the actual maximum value is not equal to 2^N-1.

If the wheel tick scaling factor to metric scale is known, set it in CFG-SFODO-FACTOR. This will slightly speed up the initial wheel tick calibration.

If the wheel tick quantization (length of wheel tick) is known, set it in CFG-SFODO-QUANT_ERROR. Note that large wheel tick quantization (dm level) reduces performance.

A typical caveat in providing wheel tick data is that it rolls over faster than the UBX-ESF-MEAS is provided to the receiver (10 Hz recommended), making the data ambiguous. The rate at which the wheel tick rolls over depends on the wheel tick maximum count, wheel tick resolution, and vehicle speed. As an example, let us assume that on the CAN bus the maximum count is available as an 8-bit value (max 2^8-1= 255), 1-bit resolution is 1 cm and we output data at 10 Hz rate. In this case when the vehicle speed exceeds 255*0.01m*10Hz= 25.5 m/s, the data becomes ambiguous. This would probably cause an issue at highway speeds. Fortunately, the odometer data, e.g., on the CAN bus is in most vehicles provided at a higher rate than recommended for UBX-ESF-MEAS rate. A typical update rate on CAN is 100 Hz, meaning the speed at which the wheel tick data becomes ambiguous is 255 m/s. The solution is then to extend the range of the wheel tick data, e.g., to 16-bit values in the host application. The host shall monitor roll-overs at 100 Hz and add the roll-overs to the wheel tick data output in UBX-ESF-MEAS at 10 Hz to the receiver. The CFG-SFODO-COUNT_MAX shall be set to 2^16-1.

B.1.4.3 About direction of travel

In some cases, the speed data, e.g., on the CAN bus is unsigned or there is no wheel rotation (direction) sense information available. In such cases the gear setting, or reverse light status information can be used to estimate if the vehicle is moving forward or backwards and then set the sign of speed or direction bit in the wheel tick data accordingly.

The use of gear setting, or reverse light status can result in wrong direction of motion information in certain scenarios.

B.2 Sensor data in UBX-ESF-MEAS

The host can provide sensor data to the receiver in UBX-ESF-MEAS messages over a communication port. See the Interface description [2] for specification of the UBX-ESF-MEAS format.

This chapter explains the population of the incoming message, that is, messages formatted and transferred from the host to the receiver.

timeTag shall describe the time of measurement. DataTypes measured simultaneously can be packed and transferred in one UBX-ESF-MEAS message with common timeTag. If different dataTypes are measured at different times, they should be sent in separate messages.

The time frame is typically the one maintained by the host. In practice, it is read from the host's ms-counter unit when sampling the measurement.

The receiver does not try to model the clock of the host. The timeTags of a particular dataType measurements are only monitored for sanity.

The receiver discards faulty measurements. Faults are reported in the UBX-ESF-STATUS flags.

Bad TTAG faults are reported in UBX-ESF-STATUS faults if:

- Successive measurements are more than 100 s apart (based on supplied timeTag).
- Successive measurements are very close to each other (supplied timeTag delta less than 5 ms).
- Negative time tag difference is detected.

- Consecutive timeTag and calibTtag (see First byte reception section for more information) differences are significantly different.

u-blox highly recommends monitoring of the UBX-ESF-STATUS faults for bad and missing measurements. This is especially useful during the system development phase.

flags shall be set to the value of numMeas in the message (optional). Bits 3...0 shall be set to zero.

id can be set to any value. It is not of particular interest to the receiver.

dataField (in the repeated group) shall the set to the measurement value in the specified units.

dataType shall denote the type of measurement.

Figure 46 shows a simplified example design in a system, where speed data is published on vehicle's CAN bus in a speed message at 100 Hz (every 10 ms). The host processor reads the CAN data continuously and at reception of every 10th speed message (every 100 ms):

- Initializes UBX-ESF-MEAS data payload to all zeroes (payload denoted here as ESF)
- Reads the ms-counter value and copies the value to ESF.timeTag.
- Writes ESF.flags = 1
- Converts speed value from the CAN message to ESF.dataField
- Sets value of dataType = 11
- Computes and sets the CRC in the UBX-ESF-MEAS message
- Outputs the UBX-ESF-MEAS to NEO-M9L

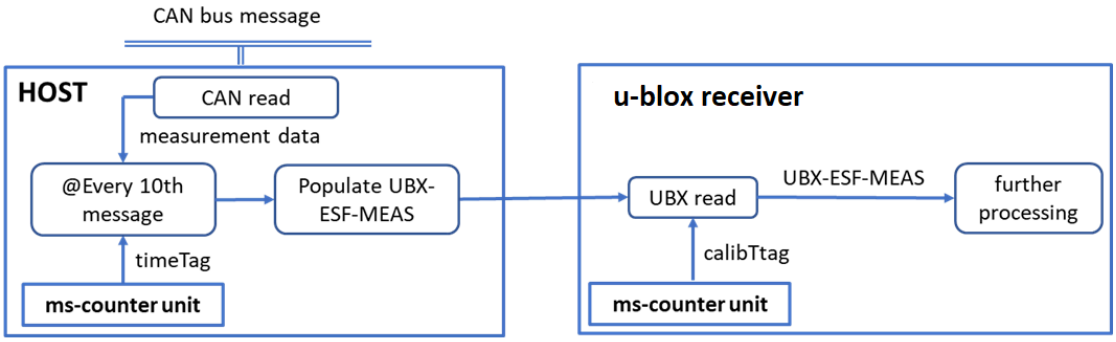


Figure 46: Example of odometer data transfer and processing

Related documents

[1] NEO-M9L-01A Data sheet, UBX-20016394
NEO-M9L-20A Data sheet, UBX-21028129
NEO-M9L-20B Data sheet, UBX-21028142, C2-Restricted
[2] M9 ADR-5.15 Interface description, UBX-22037101
M9 ADR-5.15 Interface description, UBX-22037099, C2-Restricted
[3] Packaging information for u-blox chips, modules, and antennas, UBX-14001652

☞ For regular updates to u-blox documentation and to receive product change notifications please register on our homepage https://www.u-blox.com.

Revision history

Revision	Date	Name	Status / comments
R01	21-Dec-2021	ssid	Advance information - NEO-M9L-20A
R02	19-Apr-2022	ssid	Initial production - NEO-M9L-20A and NEO-M9L-01A
R03	12-Oct-2022	njaf	ADR 5.15 FW update - Initial production Active antenna specifications added

Contact

For further support and contact information, visit us at www.u-blox.com/support.