GNSS

locate, communicate, accelerate

u-blox 7

Receiver Description

Including Protocol Specification V14

Abstract

The Receiver Description Including Protocol Specification describes the firmware features, specifications and configuration for u-blox 7 high performance high performance positioning modules.

The Receiver Description provides an overview and conceptual details of the supported features. The Protocol Specification details version 14 of the NMEA and UBX protocols and serves as a reference manual.

Document Information	
Title	u-blox 7 Receiver Description
Subtitle	Including Protocol Specification V14
Document type	Manual
Document number	GPS.G7-SW-12001-B 65525, 1 Feb 2013
Document status	Protocol version 14.00 (Public Release)

Table of Contents

Receiver Description

1 Overview

The Receiver Description Including Protocol Specification is an important resource for integrating and configuring u-blox positioning chips and modules. This document has a modular structure and it is not necessary to read it from the beginning to the end. There are 2 main sections: The Receiver Description and the Protocol Specification.

The Receiver Description describes the software aspects of system features and configuration of u-blox positioning technology. The Receiver Description is structured according to areas of functionality, with links provided to the corresponding NMEA and UBX messages, which are described in the Protocol Specification.

The Protocol Specification is a reference describing the software messages used by your u-blox GNSS (Global Navigation Satellite System: e.g. GPS, GLONASS, QZSS) receiver and is organized by the specific NMEA and UBX messages.

This document provides general information on u-blox GNSS receivers. Some information might not apply to certain products. Refer to the product Data Sheet and/or Hardware Integration Manual for possible restrictions or limitations.

2 Navigation Configuration Settings Description

This section relates to the configuration message `UBX-CFG-NAV5`.

2.1 Platform settings

u-blox positioning technology supports different dynamic platform models (see 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.

Dynamic Platform Models

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	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 vertical acceleration than a passenger car. No 2D position fixes supported.
Airborne <2g	Recommended for typical airborne environment. No 2D position fixes supported.
Airborne <4g	Only recommended for extremely dynamic environments. No 2D position fixes supported.

Dynamic Platform Model Details

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
Stationary	9000	10	6	Altitude and Velocity	Small
Pedestrian	9000	30	20	Altitude and Velocity	Small
Automotive	6000	84	15	Altitude and Velocity	Medium
At sea	500	25	5	Altitude and Velocity	Medium
Airborne <1g	50000	100	100	Altitude	Large
Airborne <2g	50000	250	100	Altitude	Large
Airborne <4g	50000	500	100	Altitude	Large

Dynamic platforms designed for high acceleration systems (e.g. airborne <2g) can result in a higher standard deviation in the reported position.

2.2 Navigation Input Filters

The navigation input filters in CFG-NAV5 mask the input data of the navigation engine.

These settings are already optimized. Do not change any parameters unless advised by u-blox support engineers.

Navigation Input Filter parameters

Parameter	*Description*
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.
fixedAlt and fixedAltVar	The fixed altitude is used if fixMode is set to 2D only. A variance greater than zero must also be supplied.
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.
cnoThreshNumSVs and cnoThresh	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.

See also comments in section Degraded Navigation below.

2.3 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 UBX-CFG-NAV5. In particular, both PDOP and accuracy values must lie below the respective limits.

Valid fixes are marked using the valid flag in certain NMEA messages (see Position Fix Flags in NMEA) and the gnssFixOK flag in UBX-NAV-PVT message.

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 normally be used.

The UBX-NAV-SOL and UBX-NAV-STATUS messages also report whether a fix is valid in their gpsFixOK and GPSfixOk flags. These messages have only been retained for backwards compatibility

and users are recommended to use the UBX-NAV-PVT message in preference.

The UBX-CFG-NAV5 message also defines TDOP and time accuracy values that are used in order to establish whether a fix is regarded as locked to GNSS or not and, as a consequence of this, which time pulse setting has to be used. Fixes that do not meet both criteria will be regarded as unlocked to GNSS and the corresponding time pulse settings of UBX-CFG-TP5 will be used to generate a time pulse.

2.4 Static Hold

Static Hold Mode allows the navigation algorithms to decrease the noise in the position output when the velocity is below a pre-defined 'Static Hold Threshold'. This reduces the position wander caused by environmental factors such as multi-path and improves position accuracy especially in stationary applications. By default, static hold mode is disabled.

If the speed drops below the defined 'Static Hold Threshold', the Static Hold Mode will be activated. Once Static Hold Mode has been entered, the position output is kept static and the velocity is set to zero until there is evidence of movement again. Such evidence can be velocity, acceleration, changes of the valid flag (e.g. position accuracy estimate exceeding the Position Accuracy Mask, see also section Navigation Output Filters), position displacement, etc.

2.5 Freezing the Course Over Ground

The receiver derives the course over ground from the GNSS velocity information. If the velocity cannot be calculated with sufficient accuracy (e.g., with bad signals) or if the absolute speed value is very low (under 0. 1m/s) then the course over ground value becomes inaccurate too. In this case the course over ground value is frozen, i.e. the previous value is kept and its accuracy is degraded over time. These frozen values will not be output in the NMEA messages NMEA-RMC and NMEA-VTG unless the NMEA protocol is explicitely configured to do so (see NMEA Protocol Configuration).

2.6 Degraded Navigation

Degraded navigation describes all navigation modes which use less than 4 Satellite Vehicles (SVs).

2.6.1 2D Navigation

If the receiver only has 3 SVs for calculating a position, the navigation algorithm uses a constant altitude to compensate for the missing fourth SV. When an SV is lost after a successful 3D fix (min. 4 SVs available), the altitude is kept constant at the last known value. This is called a 2D fix.

u-blox positioning technology does not calculate any solution with less than 3 SVs. Only u-blox timing receivers can, when stationary, calculate a timing solution with only 1 SV.

3 GNSS Configuration

The latest products from u-blox are multi-GNSS receivers capable of receiving and processing signals from multiple Global Navigation Satellite Systems (GNSS).

u-blox multi-GNSS receivers can acquire and track satellites from multiple GNSS systems and utilize them in positioning. u-blox multi-GNSS receivers can be configured to process either:

- GPS, SBAS (e.g. WAAS, EGNOS, MSAS) and QZSS L1 signals, centred on 1575.42MHz L1 frequency
- GLONASS L1 signals, centred on 1602.00MHz L1 frequency

Use the UBX-CFG-GNSS message to configure the u-blox receiver into the required mode of operation. This message allows the user to specify which GNSS signals should be processed along with limits on how many tracking channels should be allocated to each GNSS. The receiver will respond to such a request with a UBX-ACK-ACK message if it can support the requested configuration or a UBX-ACK-NAK message if not.

For maximum GPS coldstart sensitivity, ensure that the SBAS subsystem is enabled.

3.1 GLONASS

GLONASS is a GNSS operated by Russia. It has a number of significant differences when compared to GPS. In most cases u-blox receivers operate in a very similar manner when they are configured to use GLONASS signals instead of GPS. However some aspects of receiver output are likely to be noticeably affected:

- NMEA messages will change to use the GLONASS talker identifier **GL** (see section NMEA Protocol Configuration).
- UBX messages will report different satellite identity numbers (see section Satellite Numbering).
- Positioning accuracy with GLONASS only satellites may be worse than with only GPS satellites. This is because of reduced availability; the GLONASS constellation has less satellites (at the time of writing, nominally 24 for GLONASS instead of 32 for GPS). Additionally, GLONASS signals have a lower chipping rate which reduces accuracy.
- The identity of GLONASS satellites is determined by decoding specific parts of their data transmission. Therefore newly acquired GLONASS signals may be reported as coming from an "unknown" satellite until they are identified. From then on, satellites are reported using the correct satellite identity.
- As GLONASS uses a time base aligned directly to UTC, GLONASS receivers are affected by leap seconds, when the UTC time base is occasionally re-calibrated. As a consequence, users should be prepared for the receiver to restart itself if GLONASS signals are being tracked when a leap second occurs.

GPS receivers are unaffected by leap second changes as their time base (GPS time) is independent of leap seconds. GPS satellites periodically transmit information that allows the receiver to calculate UTC.

3.2 QZSS

QZSS is a GNSS operated by Japan Aerospace Exploration Agency (JAXA). It is intended as an enhancement to GPS which increases availability and positional accuracy. This can be achieved by the QZSS system transmitting GPS-compatible signals in the GPS bands.

NMEA messages will show the QZSS satellites only if configured accordingly (see section Satellite Numbering).

4 Satellite Numbering

4.1 NMEA

The NMEA protocol (V2.3) identifies satellites with a two digit number, reserving the numbers 1 to 32 for GPS, 33-64 for SBAS and 65-96 for GLONASS. So, for example, GLONASS SV4 is reported using number 68. u-blox receivers support this method in their NMEA output when "strict" SV numbering is selected. In most cases this is the default setting, but can be checked or set using `UBX-CFG-NMEA`.

Unfortunately there is currently no standard way of identifying satellites from any other GNSS within the NMEA protocol. In order to support QZSS within current receivers and prepare for support of other systems (e.g. Galileo) in future receivers, an "extended" SV numbering scheme can be enabled (using `UBX-CFG-NMEA`). This uses the NMEA-defined numbers where possible, but adds other number ranges to support other GNSS. Note however that these non-standard extensions require 3 digit numbers, which may not be supported by some NMEA parsing software. For example QZSS satellites are reported using numbers in the range 193 to 197.

GLONASS satellites can be tracked before they have been identified. In NMEA output, such unknown satellite numbers are always reported as a null field (i.e. an empty string).

4.2 UBX

UBX protocol messages use two different numbering schemes. Many UBX messages (e.g. UBX-NAV-SVINFO) use a single byte for the satellite identifier (normally named "svId"). This uses similar numbering to the "extended" NMEA scheme and is merely an extension of the scheme in use for previous generations of u-blox receivers.

With ever increasing numbers of GNSS satellites, this scheme will have to be phased out in future u-blox receivers (as numbers greater than 255 will become necessary). Consequently, newer messages use a more sophisticated, flexible and future-proof approach. This involves having a separate *gnssId* to identify which GNSS type the satellite is part of and a simple *svId* which indicates which number the satellite is in that system. In nearly all cases, this means that the "svId" is the natural number associated with the satellite. For example the GLONASS SV4 is identified as *gnssId 6, svId 4*, while the GPS SV4 is *gnssId 0, svId 4*.

GNSS Identifiers

gnssId	*GNSS Type*
0	GPS
1	SBAS
5	QZSS
6	GLONASS

Other values will be added as support for other GNSS types is enabled in u-blox receivers.

GLONASS satellites can be tracked before they have been identified. In UBX messages, such unknown satellite numbers are always reported with svId 255.

4.3 Summary

A summary of all the SV numbering schemes is provided in the following table.

Satellite numbering

GNSS Type	*SV range*	*UBX gnssId:svId*	*UBX svId*	*NMEA (strict)*	*NMEA (extended)*
GPS	G1-G32	0:1-32	1-32	1-32	1-32
SBAS	S120-S158	1:120-158	120-158	33-64	33-64,152-158
QZSS	Q1-Q5	5:1-5	193-197	-	193-197
GLONASS	R1-R32, R?	6:1-32, 6:255	65-96, 255	65-96, null	65-96, null

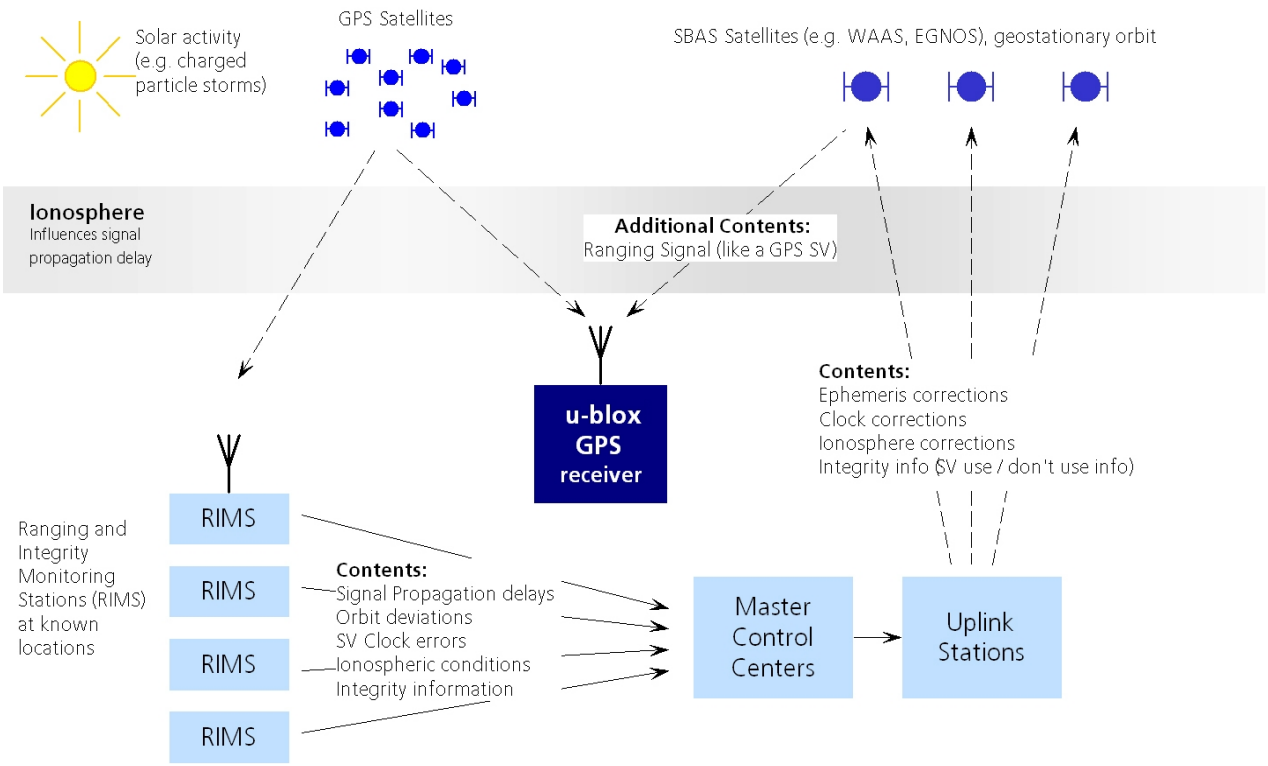
5 SBAS Configuration Settings Description

5.1 SBAS (Satellite Based Augmentation Systems)

SBAS (Satellite Based Augmentation System) is an augmentation technology for GPS, which calculates GPS integrity and correction data with RIMS (Ranging and Integrity Monitoring Stations) on the ground and uses geostationary satellites to broadcast GPS integrity and correction data to GPS users. The correction data is transmitted on the GPS L1 frequency (1575.42 MHz), and therefore no additional receiver is required to make use of the correction and integrity data.

Currently, there are no operational augmentation systems for any GNSS other than GPS. Consequently this section only addresses GPS.

SBAS Principle



There are several compatible SBAS systems available or in development all around the world:

- WAAS (Wide Area Augmentation System) for North America has been in operation since 2003.
- MSAS (Multi-Functional Satellite Augmentation System) for Asia has been in operation since 2007.
- EGNOS (European Geostationary Navigation Overlay Service) has been in operation since 2009.
- GAGAN (GPS Aided Geo Augmented Navigation), developed by the Indian government is at the time of writing in test mode.

SBAS support allows u-blox GPS technology to take full advantage of the augmentation systems that are currently available (WAAS, EGNOS, MSAS), as well as those being tested and planned (such as GAGAN).

With SBAS enabled the user benefits from additional satellites for ranging (navigation). u-blox GPS technology uses the available SBAS Satellites for navigation just like GPS satellites, if the SBAS satellites offer this service.

To improve position accuracy SBAS uses different types of correction data:

- **Fast Corrections** for short-term disturbances in GPS signals (due to clock problems, etc).
- **Long-term corrections** for GPS clock problems, broadcast orbit errors etc.
- **Ionosphere corrections** for Ionosphere activity

Another benefit of SBAS is the use of GPS integrity information. In this way SBAS Control stations can 'disable' the use of GPS satellites within a 6 second alarm time in case of major GPS satellite problems. If integrity monitoring is enabled, u-blox GPS technology only uses satellites, for which integrity information is available.

For more information on SBAS and associated services please refer to

- RTCA/DO-229D (MOPS). Available from www.rtca.org
- gps.faa.gov for information on WAAS.
- www.esa.int for information on EGNOS.
- www.essp-sas.eu for information about European Satellite Services Provider (ESSP), the EGNOS operations manager.

- www.isro.org for information on GAGAN.

SBAS satellites tracked (as of March 2012)

Identification	*Position*	*GPS PRN*	*SBAS Provider*
AMR	98° W	133	WAAS
PanAmSat Galaxy XV	133.1° W	135	WAAS
TeleSat Anik F1R	107.3° W	138	WAAS
Inmarsat 3F2 AOR-E	15.5° W	120	EGNOS
Artemis	21.5° W	124	EGNOS
Inmarsat 3F5 IOR-W	25° E	126	EGNOS
MTSAT-1R	140° E	129	MSAS
MTSAT-2	145° E	137	MSAS
Inmarsat 4 F1	55.1° E	127	GAGAN

5.2 SBAS Features

This u-blox SBAS implementation is, in accordance with standard RTCA/DO-229D, a class Beta-1 equipment. All timeouts etc. are chosen for the En Route Case. Do not use this equipment under any circumstances for safety of life applications!

u-blox receivers are capable of receiving multiple SBAS signals in parallel, even from different SBAS systems (WAAS, EGNOS, MSAS, etc.). They can be tracked and used for navigation simultaneously. Every SBAS satellite tracked utilizes one vacant receiver tracking channel. Only the number of receiver channels limits the total number of satellites used. Each SBAS satellite, which broadcasts ephemeris or almanac information, can be used for navigation, just like a normal GPS satellite.

For receiving correction data, the u-blox GPS receiver 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/or 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 (Testmode allowed/disallowed, Integrity enabled/disabled) and the signal link quality to the satellite.

In case corrections are available from the chosen SBAS satellite and used in the navigation calculation, the DGPS flag is set in the receiver's output protocol messages (see `NAV-PVT`, `NAV-SOL`, `NAV-STATUS`, `NAV-SVINFO`, `NMEA Position Fix Flags description`). The message `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 RIMS stations of a region are combined to a TEC (Total Electron Content) Map. This map is transferred to the receiver via the satellites to allow a correction of the ionosphere error on each received satellite.

Supported SBAS messages

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

Supported SBAS messages continued

Message Type	*Message Content*	*Source*
24	Mixed Fast / Long term Corrections	Primary
25	Long term Corrections	Primary
26	Ionosphere Delays	Primary

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, be careful that EGNOS satellites do not take preference over WAAS satellites, the satellites from the EGNOS system should be disallowed using the PRN Mask.

Example 2: SBAS Receiver in Europe

Some WAAS satellites 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 disallow using unwanted SBAS satellites.

The EGNOS SBAS system does not provide the satellite ranging function.

5.3 SBAS Configuration

To configure the SBAS functionalities use the UBX proprietary message `UBX-CFG-SBAS` (SBAS Configuration).

SBAS Configuration parameters

Parameter	*Description*
Mode - SBAS Subsystem	Enables or disables the SBAS subsystem
Mode - Allow test mode usage	Allow / Disallow SBAS usage from satellites in Test Mode (Message 0)
Services/Usage - Ranging	Use the SBAS satellites for navigation
Services/Usage - Apply SBAS correction data	Combined enable/disable switch for Fast-, Long-Term and Ionosphere Corrections
Services/Usage - Apply integrity information	Use integrity data
Number of tracking channels	Should be set using `UBX-CGF-GNSS`. The field in `UBX-CFG-SBAS` is no longer supported.
PRN Mask	Allows selectively enabling/disabling SBAS satellites (e.g. restrict SBAS usage to WAAS-only).

By default SBAS is enabled with three prioritized SBAS channels and it will use any received SBAS satellites (except for those in test mode) for navigation, ionosphere parameters and corrections.

6 Clocks and Time

6.1 Receiver Local Time

The receiver is dependent on a local oscillator (normally a TCXO or Crystal oscillator) for both the operation of its radio parts and also for timing within its signal processing. No matter what the nominal frequency the local oscillator is (e.g. 26MHz), u-blox receivers subdivide the oscillator signal to provide a 1kHz reference clock signal which is used to drive many of the receiver's processes. In particular the measurement of satellite signals is arranged to happen synchronised with the "ticking" of this 1kHz 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 of these 1kHz clock ticks takes place in the time-base of the relevant GNSS system. In previous versions of the firmware for u-blox receivers this was always the GPS time-base, but in the latest firmware it could be GPS or GLONASS and in the future it could also be other GNSS systems (such as Galileo, Compass.... etc). This estimate of GNSS time based on the local 1kHz clock is called **receiver local time**.

As receiver local time is a mapping of the local 1kHz 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 receiver local time will indicate the length of time that the receiver has been running. However, when the receiver obtains some credible timing information from a satellite or aiding message, it will jump to an estimate of GNSS time.

6.2 Navigation Epochs

Each navigation solution is triggered by the tick of the 1kHz 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 the **clock bias** (and the **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 1kHz clock tick which it estimates to be closest to the desired fix period as measured in GNSS system time. Consequently the number of 1kHz clock ticks between fixes will occasionally vary (so 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-AID-HUI` 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 conversion to UTC is easier (as GLONASS time is closely linked to UTC).

Where insufficient information is available for the receiver to perform any of these time-base conversions precisely, pre-defined 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`).

Future u-blox receivers are likely to employ multiple GNSS system times and/or receiver local times (in order to support multiple GNSS systems in parallel), so users should not rely on UBX messages that report GNSS system time or receiver local time being supported in future. It is therefore recommended to give preference to those messages that report UTC time.

6.3 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.

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. If reliable absolute time information is required, users are recommended to use the UBX-NAV-TIMEUTC, UBX-NAV-TIMEGPS, UBX-NAV-PVT or UBX-NAV-SOL messages, which contain additional fields that indicate the validity and accuracy of the calculated times.

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 GPS/GNSS receivers use this representation internally, only converting to a more "conventional forms" at external interfaces. The iTOW field is the most obvious externally visible consequence of this internal representation.

6.4 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). What is more, although compatibility mode (selected using UBX-CFG-NMEA) requires three decimal places, rounding to the nearest hundredth of a second remains, so the extra digit is always 0.

UTC time is 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 synchronisation 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 will need to 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.

6.5 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 and UBX messages should be prepared to handle minutes that are either 59 or 61 seconds long.

Note that the behavior of GLONASS signals during leap seconds is not well defined. As a consequence, users should be prepared for the receiver to restart itself if GLONASS signals are being tracked when a leap second occurs.

6.6 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 initialise receiver local time and in most cases this leads to appreciably faster first fixes.

7 Serial Communication Ports Description

u-blox positioning technology comes with a highly flexible communication interface. It supports the NMEA and the proprietary UBX protocols, and is truly multi-port and multi-protocol capable. Each protocol (UBX, NMEA) can be assigned to several ports at the same time (multi-port capability) with individual settings (e.g. baud rate, message rates, etc.) for each port. It is even possible to assign more than one protocol (e.g. UBX protocol and NMEA at the same time) to a single port (multi-protocol capability), which is particularly useful for debugging purposes.

To enable a message on a port the UBX and/or NMEA protocol must be enabled on that port using the UBX proprietary message `CFG-PRT`. This message also allows changing port-specific settings (baud rate, address etc.). See `CFG-MSG` for a description of the mechanism for enabling and disabling messages.

The following table shows the port numbers used. Note that any numbers not listed are reserved for future use.

Port Number assignment

Port #	*Electrical Interface*
0	DDC (I²C compatible)
1	UART 1
3	USB
4	SPI

7.1 TX-ready indication

This feature enables each port to define a corresponding pin, which indicates if bytes are ready to be transmitted. 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 (note that this is not necessarily equal to all bytes transmitted, i.e. after the pin has become inactive, up to 16 bytes can still need to be transferred to the host).

The TX-ready pin can be selected from all PIOs which are not in use (see `MON-HW` for a list of the PIOs and their mapping), each TX-ready pin is exclusively for one port and cannot be shared. If the PIO is invalid or already in use, only the configuration for the 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 polled and the settings of TX-ready feature verified (will be set to disabled/all zero if settings invalid).

The threshold should not be set above 2 kB, as the internal message buffer limit can be reached before this, resulting in the TX-ready pin never being set as messages are discarded before the threshold is reached.

7.2 Extended TX timeout

If the host does not communicate over SPI or DDC 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 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.

7.3 UART Ports

One or two Universal Asynchronous Receiver/Transmitter (UART) ports are featured, that can be used to transmit GNSS measurements, monitor status information and configure the receiver. See our online product descriptions for availability.

The serial ports consist of an RX and a TX line. Neither handshaking signals nor hardware flow control signals are available. These serial ports operate in asynchronous mode. The baud rates can be configured individually for each serial port. However, there is no support for setting different baud rates for reception and transmission or for different protocols on the same port.

Possible UART Interface Configurations

Baud Rate	*Data Bits*	*Parity*	*Stop Bits*
4800	8	none	1
9600	8	none	1
19200	8	none	1
38400	8	none	1
57600	8	none	1
115200	8	none	1

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 losses, the baudrate and communication speed or the number of enabled messages should be selected so that the expected number of bytes can be transmitted in less than one second.

See `CFG-PRT for UART` for a description of the contents of the UART port configuration message.

7.4 USB Port

One Universal Serial Bus (USB) port is featured. See the Data Sheet of your specific product for availability. This port can be used for communication purposes and to power the positioning chip or module.

The USB interface supports two different power modes:

- In *Self Powered Mode* the receiver is powered by its own power supply. **VDDUSB** is used to detect the availability of the USB port, i.e. whether the receiver is connected to a USB host.

- In *Bus Powered Mode* the device is powered by the USB bus, therefore no additional power supply is needed. See the table below for the default maximum current that can be drawn by the receiver. See `CFG-USB` for a description on how to change this maximum. Configuring Bus Powered Mode indicates that the device will enter a low power state with disabled GNSS functionality when the host suspends the device, e.g. when the host is put into stand-by mode.

Maximum Current in Bus Powered Mode

Generation	*Max Current*
u-blox 7	50 mA

The voltage range for **VDDUSB** *is specified from 3.0V to 3.6V, which differs slightly from the specification for VCC*

7.5 DDC Port

A Display Data Channel (DDC) bus is implemented, which is a 2-wire communication interface compatible with the I²C standard (Inter-Integrated Circuit). See our online product selector matrix for availability.

Unlike all other interfaces, the DDC is not able to communicate in full-duplex mode, i.e. TX and RX are mutually exclusive. u-blox receivers act as a slave in the communication setup, therefore they cannot initiate data transfers on their own. The host, which is always master, provides the data clock (SCL), and the clock frequency is therefore not configurable on the slave.

The clock rate on the SCL line generated by the master must not exceed 400 kHz (fast-mode).

The receiver's DDC address is set to 0x42 by default. This address can be changed by setting the `mode` field in `CFG-PRT for DDC` accordingly.

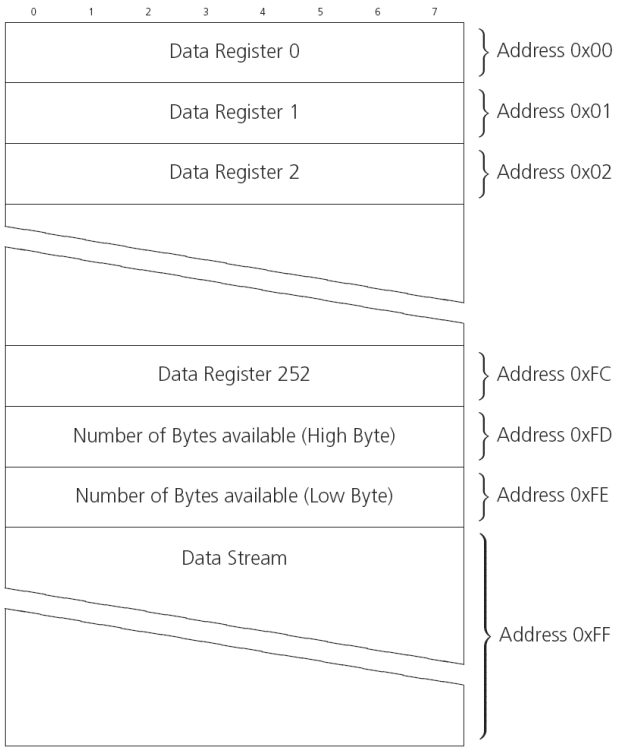
As the receiver will be run in slave mode and the physical layer lacks a handshake mechanism to inform the master about data availability, a layer has been inserted between the physical layer and the UBX and NMEA layer. The DDC implements a simple streaming interface that allows the constant polling of data, discarding everything that is not parseable. This means that the receiver returns 0xFF if no data is available. The TX-ready feature can be used to inform the master about data availability and can be used as a trigger for data transmission.

7.5.1 Read Access

To allow both polled access to the full message stream and quick access to the key data, the register layout depicted in Figure *DDC Register Layout* is provided. The data registers 0 to 252, at addresses 0x00 to 0xFC, each 1 byte in size, contain information to be defined at a later point in time. At addresses 0xFD and 0xFE, the currently available number of bytes in the message stream can be read. At address 0xFF, the message stream is located. Subsequent reads from 0xFF return the messages in the transmit buffer, byte by byte. If the number of bytes read exceeds the number of bytes indicated, the payload is padded using the value 0xFF.

The registers 0x00 to 0xFC will be defined in a later firmware release. Do not use them, as they don't provide any meaningful data!

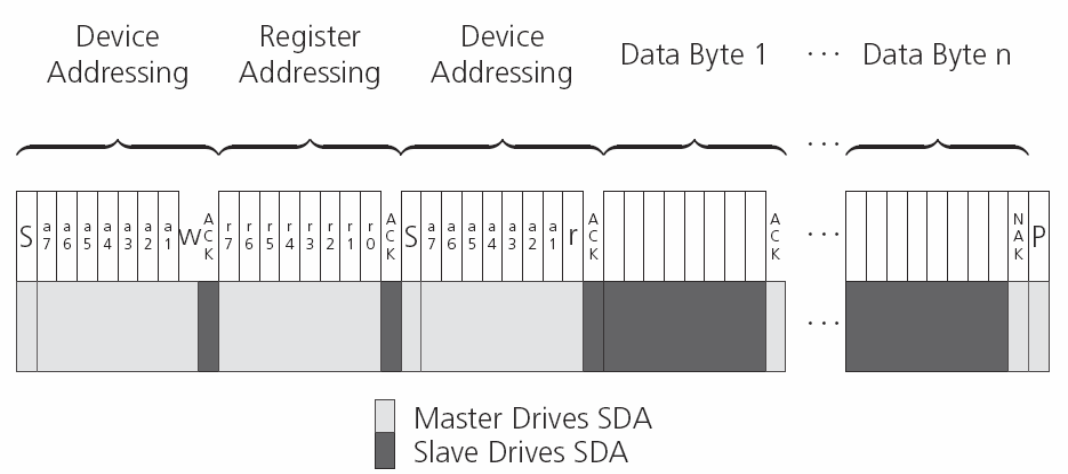
DDC Register Layout



7.5.1.1 Random Read Access

Random read operations allow the master to access any register in a random manner. To perform this type of read operation, first the register address to read from must be written to the receiver (see Figure *DDC Random Read Access*). 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. Next, the 8-bit address of the register to be read must be written to the bus. Following the receiver's acknowledge, 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. After every byte being read, the internal address counter is incremented by one, saturating at 0xFF. This saturation means, that, after having read all registers coming after the initially set register address, the raw message stream can be read.

DDC Random Read Access

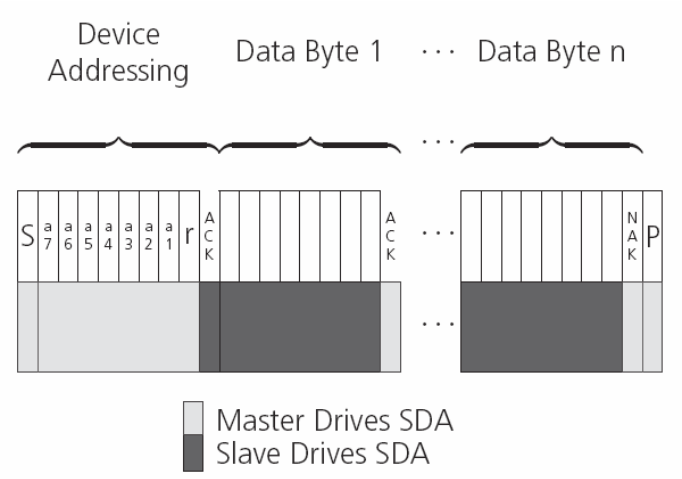


7.5.1.2 Current Address Read

The receiver contains an address counter that maintains the address of the last register accessed, internally incremented by one. Therefore, if the previous read access was to address `n` (where `n` is any legal address), the next current address read operation would access data from address `n+1` (see Figure *DDC Current Address Read Access*). Upon receipt of the device address with the `RW` bit set to one, the receiver issues an acknowledge and 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.

To allow direct access to streaming data, the internal address counter is initialized to 0xFF, meaning that current address reads without a preceding random read access return the raw message stream. The address counter can be set to another address at any point using a random read access.

DDC Current Address Read Access

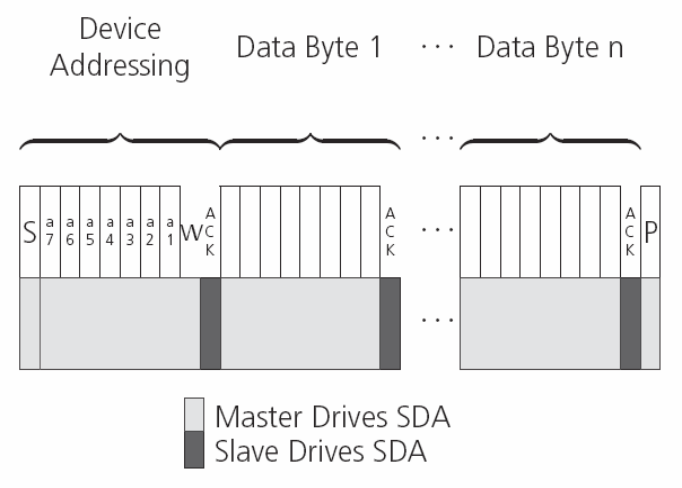


7.5.2 Write Access

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 section Read Access is not writable. 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. Now, 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.

DDC Write Access



7.6 SPI Port

A Serial Peripheral Interface (SPI) bus is available with selected receivers. See our online product descriptions for availability.

SPI is a four-wire synchronous communication interface. In contrast to UART, the master provides the clock signal, which therefore doesn't need to be specified for the slave in advance. Moreover, a baud rate setting is not applicable for the slave. SPI modes 0-3 are implemented and can be configured using the field **mode.spiMode** in **CFG-PRT for SPI** (default is SPI mode 0).

The SPI clock speed is limited depending on hardware and firmware versions!

7.6.1 Maximum SPI clock speed

u-blox 7

Firmware Version	*Max SPI speed*
1.00	5.5 MHz

7.6.2 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). 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.

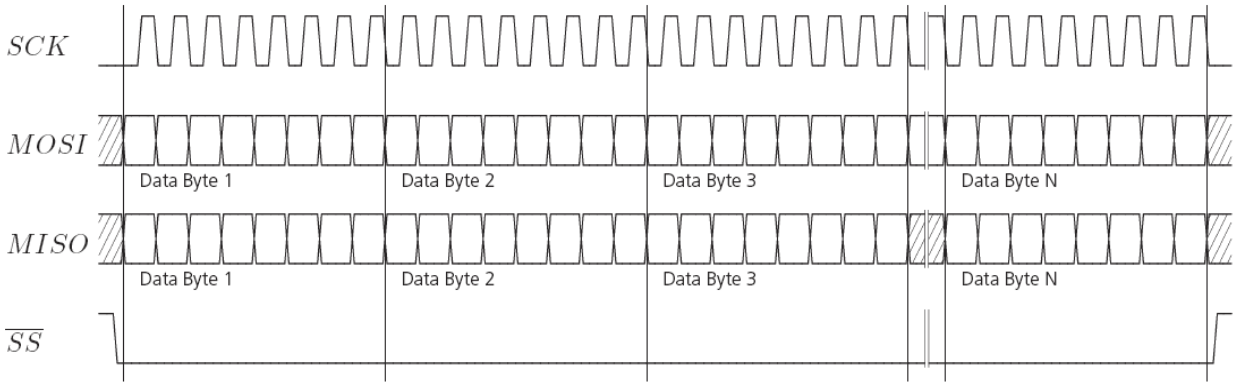
To prevent the receiver from being busy parsing incoming data, the parsing process is stopped after 50 subsequent bytes containing 0xFF. The parsing process is re-enabled with the first byte not equal to 0xFF. The number of bytes to wait for deactivation (50 by default) can be adjusted using the field **mode.ffCnt** in **CFG-PRT for SPI**, which is only necessary when messages shall be sent containing a large number of subsequent 0xFF bytes.

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.

7.6.3 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 simultaneous 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.

SPI Back-To-Back Read/Write Access



7.7 How to change between protocols

Reconfiguring a port from one protocol to another is a two-step process:

- Step 1: the preferred protocol(s) needs to be enabled on a port using `CFG-PRT`. One port can handle several protocols at the same time (e.g. NMEA and UBX). By default, all ports are configured for UBX and NMEA protocol so in most cases, it's not necessary to change the port settings at all. Port settings can be viewed and changed using the `CFG-PRT` messages.
- Step 2: activate certain messages on each port using `CFG-MSG`.

8 Receiver Configuration

8.1 Configuration Concept

u-blox positioning technology is fully configurable with UBX protocol configuration messages (message class UBX-CFG). The configuration used by the GNSS receiver during normal operation is termed "Current Configuration". The Current Configuration can be changed during normal operation by sending any UBX-CFG-XXX message to the receiver over an I/O port. The receiver will change its Current Configuration immediately after receiving the configuration message. The GNSS receiver always uses only the Current Configuration.

Unless the Current Configuration is made permanent by using `UBX-CFG-CFG` as described below, the Current Configuration will be lost in case of:

- a power cycle
- a hardware reset
- a (complete) controlled software reset

See the section on resetting a receiver for details.

The Current Configuration can be made permanent (stored in a non-volatile memory) by saving it to the "Permanent Configuration". This is done by sending a `UBX-CFG-CFG` message with an appropriate **saveMask** (UBX-CFG-CFG/save).

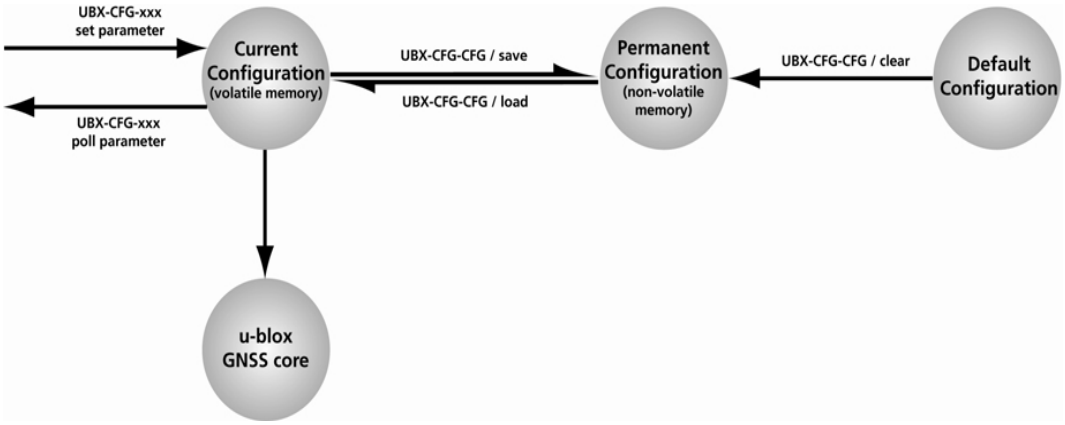
The Permanent Configuration is copied to the Current Configuration after start-up or when a UBX-CFG-CFG message with an appropriate **loadMask** (UBX-CFG-CFG/load) is sent to the receiver.

The Permanent Configuration can be restored to the receiver's Default Configuration by sending a UBX-CFG-CFG message with an appropriate **clearMask** (UBX-CFG-CFG/clear) to the receiver.

This only replaces the Permanent Configuration, not the Current Configuration. To make the receiver operate with the Default Configuration which was restored to the Permanent Configuration, a UBX-CFG-CFG/load command must be sent or the receiver must be reset.

The mentioned masks (saveMask, loadMask, clearMask) are 4-byte bitfields. Every bit represents one configuration sub-section. These sub-sections are defined in section "Organization of the Configuration Sections". All three masks are part of every UBX-CFG-CFG message. Save, load and clear commands can be combined in the same message. Order of execution is: clear, save, load.

The following diagram illustrates the process:



8.2 Organization of the Configuration Sections

The configuration is divided into several sub-sections. Each of these sub-sections corresponds to one or several UBX-CFG-XXX messages. The sub-section numbers in the following tables correspond to the bit position in the masks mentioned above. All values not listed are reserved

Configuration sub-sections

Number	*Name*	*CFG messages*	*Description*
0	PRT	UBX-CFG-PRT UBX-CFG-USB	Port and USB settings
1	MSG	UBX-CFG-MSG	Message settings (enable/disable, update rate)
2	INF	UBX-CFG-INF	Information output settings (Errors, Warnings, Notice, Test etc.)
3	NAV	UBX-CFG-NAV5 UBX-CFG-NAVX5 UBX-CFG-DAT UBX-CFG-RATE UBX-CFG-SBAS UBX-CFG-NMEA	Navigation Parameter, Receiver Datum, Measurement and Navigation Rate setting, SBAS settings, NMEA protocol settings
4	RXM	UBX-CFG-GNSS UBX-CFG-TP5 UBX-CFG-RXM UBX-CFG-PM2 UBX-CFG-ITFM	GNSS Settings, Power Mode Settings, Time Pulse Settings, Jamming/Interference Monitor Settings
9	RINV	UBX-CFG-RINV	Remote Inventory configuration

Configuration sub-sections continued

Number	*Name*	*CFG messages*	*Description*
10	ANT	UBX-CFG-ANT	Antenna configuration
11	LOG	UBX-CFG-LOGFILTER	Logging configuration

8.3 Permanent Configuration Storage Media

The Current Configuration is stored in the receiver's volatile RAM. Hence, any changes made to the Current Configuration without saving will be lost if any of the reset events listed in the section above occur. By using UBX-CFG-CFG/save, the selected configuration sub-sections are saved to all non-volatile memories available:

- On-chip BBR (battery backed RAM). In order for the BBR to work, a backup battery must be applied to the receiver.
- External flash memory, where available.

8.4 Receiver Default Configuration

The Permanent Configuration can be reset to Default Configuration through a `UBX-CFG-CFG`/clear message. The receiver's Default Configuration is normally determined when the receiver is manufactured. Refer to specific product data sheet for further details.

9 Forcing a Receiver Reset

Typically, in GNSS receivers, one distinguishes between Cold, Warm, and Hot starts, depending on the type of valid information the receiver has at the time of the restart.

- **Cold start** In this 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. Please note that some competitors 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 exist. See the section on Aiding and Acquisition.
- **Hot start** In Hot start, the receiver was powered down only for a short time (4 hours or less), so that its ephemeris is still valid. Since the receiver doesn't need to download ephemeris again, this is the fastest startup method.

In the `UBX-CFG-RST` message, one 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 CFG-RST message offers the `navBbrMask` field, where Hot, Warm and Cold starts can be initiated, and also other combinations thereof.

Data stored in flash memory is not cleared by any of the options provided by UBX-CFG-RST. So, for example, if valid AlmanacPlus data stored in the flash it is likely to have an impact on a "Cold start".

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. This is equivalent to pulling the Reset signal on the receiver.

- **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.
- **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.

10 Remote Inventory

10.1 Description

The *Remote Inventory* enables storing user-defined data in the non-volatile memory of the receiver. The data can be either binary or a string of ASCII characters. In the second case, it is possible to dump the data at startup.

10.2 Usage

- The contents of the *Remote Inventory* can be set and polled with the message `UBX-CFG-RINV`. Refer to the message specification for a detailed description.
- If the contents of the *Remote Inventory* are polled without having been set before, the default configuration (see table below) is output.

Default configuration

Parameter	*Value*
flags	0x00
data	"Notice: no data saved!"

As with all configuration changes, these must be saved in order to be made permanent. Make sure to save the section RINV before resetting or switching off the receiver. More information about saving a configuration section can be found in chapter Configuration Concept.

11 Power Management

u-blox receivers support different power modes. These modes represent strategies of how to control the acquisition and tracking engines in order to achieve either the best possible performance or good performance with reduced power consumption.

Power modes are selected using the message `CFG-RXM` and configured using `UBX-CFG-PM2`.

11.1 Continuous Mode

During a Cold start, a receiver in Continuous Mode continuously deploys the acquisition engine to search for all satellites. Once a position can be calculated and a sufficient number of satellites are being tracked, the acquisition engine is powered off resulting in significant power savings. The tracking engine continuously tracks acquired satellites and acquires other available or emerging satellites. Whenever the receiver can not calculate a position anymore or the number of satellites tracked is below the sufficient number, the acquisition engine is powered on again to guarantee a quick reacquisition.

Note that even if the acquisition engine is powered off, satellites continue to be acquired.

11.2 Power Save Mode

Power Save Mode (PSM) allows a reduction in system power consumption by selectively switching parts of the receiver on and off.

Note: Power Save Mode cannot be selected when the receiver is configured to process GLONASS signals.

11.2.1 Operation

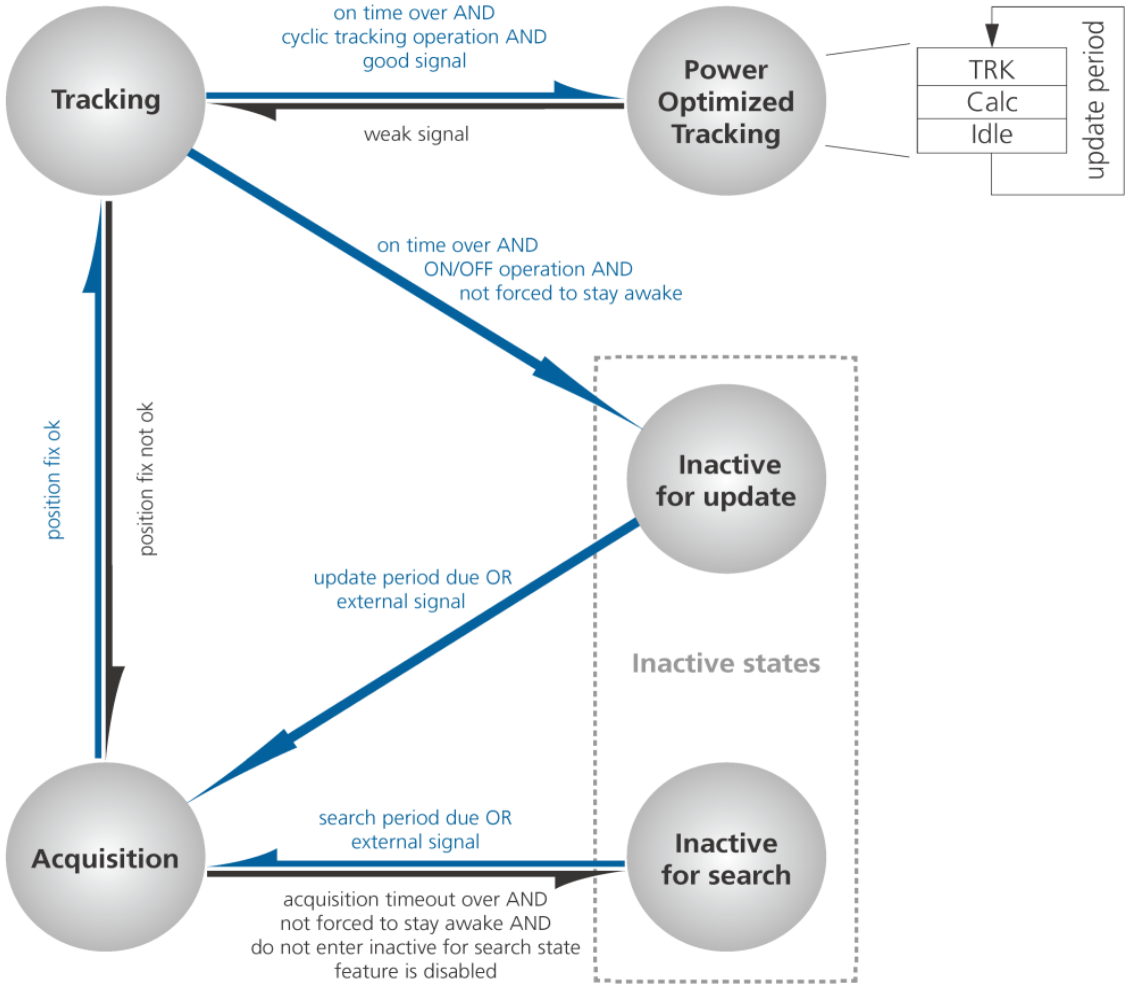
Power Save Mode has two modes of operation: cyclic tracking operation and ON/OFF operation. Cyclic tracking operation is used when position fixes are required in short periods of 1 to 10s. ON/OFF operation on the other hand is used for periods longer than 10s. Periods in ON/OFF operation can be in the order of minutes, hours or days. The mode of operation can be configured and depending on the setting, the receiver demonstrates different behavior: In ON/OFF operation the receiver switches between phases of startup/navigation and phases with low or almost no system activity. In cyclic tracking the receiver does not shut down completely between fixes, but uses low power tracking instead.

PSM is based on a state machine with five different states: *Inactive for update* and *Inactive for search* states, *Acquisition* state, *Tracking* state and *Power Optimized Tracking (POT)* state.

- *Inactive* states: Most parts of the receiver are switched off.
- *Acquisition* state: The receiver actively searches for and acquires signals. Maximum power consumption.
- *Tracking* state: The receiver continuously tracks and downloads data. Less power consumption than in *Acquisition* state.
- *POT* state: The receiver repeatedly loops through a sequence of tracking (TRK), calculating the position fix (Calc), and entering an idle period (Idle). No new signals are acquired and no data is downloaded. Much less power consumption than in *Tracking* state.

The following figure illustrates the state machine:

State machine

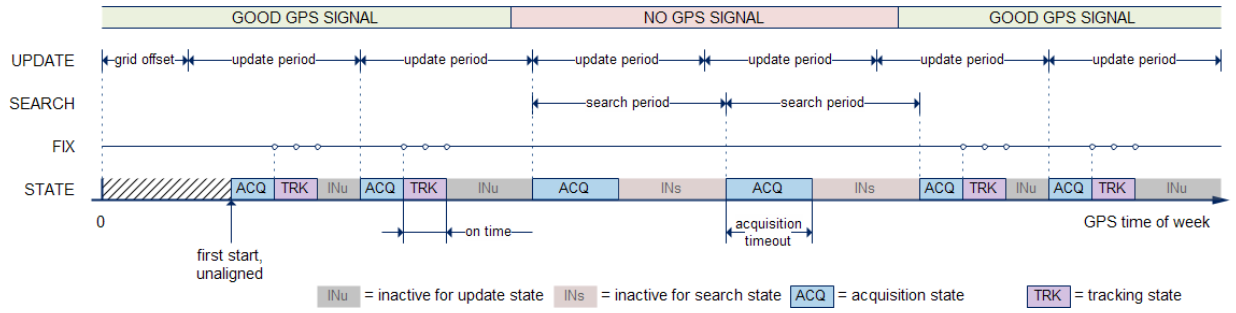


11.2.1.1 ON/OFF operation - long update period

When the receiver is switched on, it first enters *Acquisition* state. If it is able to obtain a valid position fix within the time given by the acquisition timeout, it switches to *Tracking* state. Otherwise it enters *Inactive for search* state and re-starts after the configured search period (minus a startup margin). As soon as the receiver gets a valid position fix (one passing the navigation output filters), it enters *Tracking* state. Upon entering *Tracking* state, the on time is started. Once the on time is over *Inactive for update* state is entered and the receiver re-starts according to the configured update grid (see chapter Grid offset for an explanation). If the signal is lost while in *Tracking* state, *Acquisition* state is entered. If the signal is not found within the acquisition timeout, the receiver enters *Inactive for search* state. Otherwise the receiver will re-enter *Tracking* state and stay there until the newly started on time is over.

The diagram below illustrates how ON/OFF operation works:

Diagram of ON/OFF operation

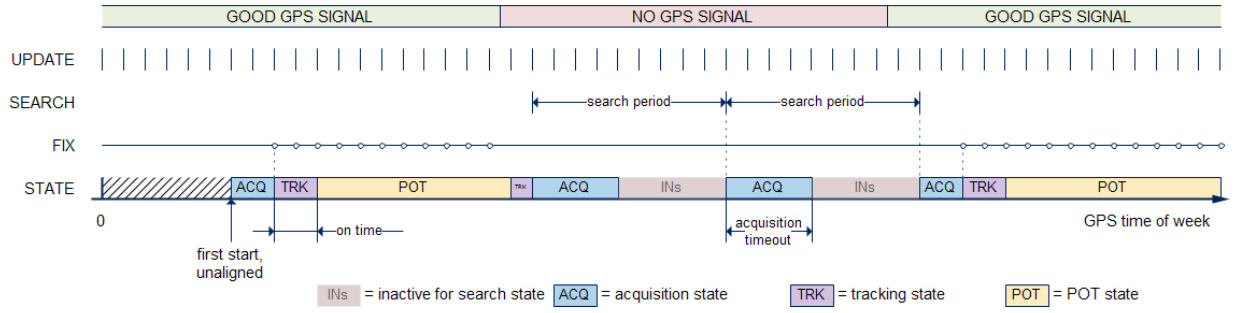


11.2.1.2 Cyclic tracking operation - short update period

When the receiver is switched on, it first enters *Acquisition* state. If it is able to obtain a position fix within the time given by the acquisition timeout, it switches to *Tracking* state. Otherwise, it will enter *Inactive for search* state and re-start within the configured search grid. After a valid position fix, *Tracking* state is entered and the on time is started. In other words the on time is started with the first valid position fix. Once the on time is over, *POT* state is entered. In *POT* state the receiver continues to output position fixes according to the update period. To have maximum power savings, set the on time to zero. This causes the receiver to enter *POT* state as soon as possible. If the signal becomes weak or is lost during *POT* state, *Tracking* state is entered. Once the signal is good again and the newly started on time is over, the receiver will re-enter *POT* state. If the receiver can't get a position fix in the *Tracking* state, it enters *Acquisition* state. Should the acquisition fail as well, *Inactive for search* state is entered.

The diagram below illustrates how cyclic tracking operation works:

Diagram of cyclic tracking operation



11.2.1.3 User controlled operation - update and search period of zero

Setting the update period to zero causes the receiver to wait in the *Inactive for update* state until woken up by the user. Setting the search period to zero causes the receiver to wait in the *Inactive for search* state indefinitely after an unsuccessful start-up. Any wake-up event will re-start the receiver. See chapter Wake-up for more information on wake-up events.

External wake-up is required when setting update or search period to zero!

11.2.1.4 Satellite data download

The receiver is not able to download satellite data (e.g. the ephemeris) while it is working in ON/OFF or cyclic tracking operation. Therefore it has to temporarily switch to continuous operation for the time the satellites transmit the desired data. To save power the receiver schedules the downloads according to an internal timetable and only switches to continuous operation while data of interest is being transmitted by the SVs.

Each SV transmits its own ephemeris data. Ephemeris data download is feasible when the corresponding SV

has been tracked with a minimal C/No over a certain period of time. The download is scheduled in a 30 minute grid or immediately when fewer than a certain number of visible SVs have valid ephemeris data.

Almanac, ionosphere, UTC correction and SV health data are transmitted by all SVs simultaneously. Therefore these parameters can be downloaded when a single SV is tracked with a high enough C/No.

11.2.2 Configuration

Power Save Mode is enabled and disabled with the UBX-CFG-RXM message and configured with the UBX-CFG-PM2 message.

When enabling Power Save Mode, SBAS support can be disabled (UBX-CFG-SBAS) since the receiver will be unable to download any SBAS data in this mode.

A number of parameters can be used to customize PSM to your specific needs. These parameters are listed in the following table:

Power Save Mode configuration options

Parameter	*Description*
Mode of operation	Receiver mode of operation
Update period	Time between two position fix attempts
Search period	Time between two acquisition attempts if the receiver is unable to get a position fix
Acquisition timeout	Time after which the receiver stops acquisition and enters *Inactive for search* state
On-time	Time the receiver remains in *Tracking* state and produces position fixes
Wait for timefix	Wait for time fix before entering *Tracking* state
Do not enter *Inactive for search* state	Receiver does not enter *Inactive for search* state if it can't get a position fix but keeps trying instead
Update RTC	Enables periodic Real Time Clock (RTC) update
Update Ephemeris	Enables periodic ephemeris update
EXTINT selection	Selects EXTINT pin used with pin control feature
EXTINT 'high' keeps awake	Enables force-ON pin control feature
EXTINT 'low' forces sleep	Enables force-OFF pin control feature
Grid offset	Time offset of update grid with respect to GPS start of week

11.2.2.1 Mode of operation

The mode of operation to use mainly depends on the update period: For short update periods (in the range of a few seconds), cyclic tracking should be configured. On the other hand, for long update periods (in the range of minutes or longer) only work with ON/OFF operation.

See chapter ON/OFF operation - long update period and Cyclic tracking operation - short update period for more information on the two modes of operation.

11.2.2.2 Update and search period

The update period specifies the time between successive position fixes. If no position fix can be obtained within the acquisition timeout, the receiver will retry after the time specified by the search period. Update and search period are fixed with respect to an absolute time grid based on GPS time. They do not refer to the time of the last valid position fix or last position fix attempt.

New settings are ignored if the update period or the search period exceeds the maximum number of milliseconds in a week. In that case the previously stored values remain effective.

11.2.2.3 Acquisition timeout

The receiver tries to obtain a position fix within the time given in the acquisition timeout. This setting is treated as a minimum value. If the receiver determines that it needs more time for the given starting conditions, it will automatically prolong this time. If set to zero, the acquisition timeout is exclusively determined by the receiver. In case of a very weak or no GPS signal, the timeout determined by the receiver may be shortened in order to save power. However, the acquisition timeout will never be shorter than the configured value.

11.2.2.4 On time and wait for timefix

The on time specifies how long the receiver stays in *Tracking* state before switching to *POT* and *Inactive for update* state respectively. The quality of the position fixes can be configured by setting the masks in the message `UBX-CFG-NAV5`. If the *wait for timefix* option is enabled the transition from *Acquisition* to *Tracking* state is made only if the GPS time is known and within the configured limits, and the receiver is continuously producing position fixes for more than two seconds. Thus enabling the wait for timefix option usually delays the transition from *Acquisition* to *Tracking* state by a few seconds. Keep in mind that setting harder limits in `UBX-CFG-NAVX5` will prolong start-up time so you might want to increase the acquisition timeout.

11.2.2.5 Do not enter 'inactive for search' state when no fix

If this option is enabled, the receiver acts differently in case it can't get a fix: instead of entering *Inactive for search* state, it keeps trying to acquire a fix. In other words, the receiver will never be in *Inactive for search* state and therefore the search period and the acquisition timeout are obsolete.

11.2.2.6 Update RTC and Ephemeris

To maintain the ability of a fast start-up, the receiver needs to calibrate its RTC and update its ephemeris data on a regular basis. This can be ensured by activating the update RTC and update Ephemeris option. The RTC is calibrated every 5 minutes and the ephemeris data is updated approximately every 30 minutes. See chapter Satellite data download for more information.

11.2.2.7 EXTINT pin control

The pin control feature allows overriding the automatic active/inactive cycle of Power Save Mode. The state of the receiver can be controlled through either the EXTINT0 or the EXTINT1 pin.

If the Force-ON feature is enabled, the receiver will not enter the *Inactive* states as long as the configured EXTINT pin (either EXTINT0 or EXTINT1) is at a 'high' level. The receiver will therefore always be in *Acquisition/Tracking* states (ON/OFF operation) and *Acquisition/Tracking/POT* states (cyclic tracking operation) respectively. When the pin level changes to 'low' the receiver continues with its configured behavior. UBX-CFG-PM2 is used to select and configure the pin that will control the behavior as described above.

If the Force-OFF feature is enabled, the receiver will enter *Inactive* state and remain there until the next wake-up event. Any wake-up event can wake up the receiver, even while the EXTINT pin is set to Force-OFF. However, the receiver will only wake up for the time period needed to read the configuration pin settings, i.e. Force-OFF, and will then enter *Inactive* state again.

11.2.2.8 Grid offset

Once the receiver has a valid time, the update grid is aligned to the start of the GPS week (Sunday at 00:00 o'clock). Before having a valid time, the update grid is unaligned. A grid offset now shifts the update grid with respect to the start of the GPS week. An example of usage can be found in chapter Use grid offset.

The grid offset is not used in cyclic tracking operation.

11.2.3 Features

11.2.3.1 Communication

When PSM is enabled, communication with the receiver (e.g. UBX message to disable PSM) requires particular attention. This is because the receiver may be in *Inactive* state and therefore unable to receive any message through its interfaces. To ensure that the configuration messages are processed by the receiver, even while in *Inactive* state, the following steps need to be taken:

- Send a dummy sequence of 0xFF (one byte is sufficient) to the receiver's UART interface. This will wake the receiver up in case it is in *Inactive* state. If the receiver is not in *Inactive* state, the sequence will be ignored.
- Send the configuration message about half a second after the dummy sequence. If the interval between the dummy sequence and the configuration message is too short, the receiver may not yet be ready. On the other hand, if the interval is too long, the receiver may return to *Inactive* state before the configuration message was received. It is therefore important to check for a `UBX-ACK-ACK` reply from the receiver to confirm that the configuration message was received.
- Send the configuration save message immediately after the configuration message.

11.2.3.2 Wake-up

The receiver can be woken up by generating an edge on one of the following pins:

- rising or falling edge on one of the EXTINT pins
- rising or falling edge on the RXD1 pin
- rising edge on NRESET pin

All wake-up signals are interpreted as a position request, where the receiver wakes up and tries to obtain a position fix. Wake-up signals have no effect if the receiver is already in *Acquisition*, *Tracking* or *POT* state.

11.2.3.3 Behavior while USB host connected

As long as the receiver is connected to a USB host, it will not enter the lowest possible power state. This is because it must retain a small level of CPU activity to avoid breaching requirements of the USB specification. The drawback, however, is that power consumption is higher.

Wake-up by pin/UART is possible even if the receiver is connected to a USB host. The state of the pin must be changed for at least one millisecond.

11.2.3.4 Cooperation with the AssistNow Autonomous feature

If both PSM and AssistNow Autonomous features are enabled, the receiver won't enter *Inactive for update* state as long as *AssistNow Autonomous* carries out calculations. This prevents losing data from unfinished calculations and, in the end, reduces the total extra power needed for *AssistNow Autonomous*. The delay before entering *Inactive for update* state, if any, will be in the range of several seconds, rarely more than 20 seconds.

Only entering *Inactive for update* state is affected by *AssistNow Autonomous*. In other words: in cyclic tracking operation, *AssistNow Autonomous* will not interfere with the PSM (apart from the increased power consumption).

Enabling the AssistNow Autonomous feature will lead to increased power consumption while prediction is calculated. The main goal of PSM is to reduce the overall power consumption. Therefore for each application special care must be taken to judge whether AssistNow Autonomous is beneficial to the overall power consumption or not.

11.2.4 Examples

11.2.4.1 Use Grid Offset

Scenario: Get a position fix once a day at a fixed time. If the position fix cannot be obtained try again every two hours.

Solution: First set the update period to 24*3600s and the search period to 2*3600s. Now a position fix is obtained every 24 hours and if the position fix fails retrials are scheduled in two hour intervals. As the update grid is aligned to midnight Saturday/Sunday, the position fixes happen at midnight. By setting the grid offset to 12*3600s the position fixes are shifted to once a day at noon. If the position fix at noon fails, retrials take place every two hours, the first at 14:00. Upon successfully acquiring a position fix the next fix attempt is scheduled for noon the following day.

11.2.4.2 Use update periods of zero

Scenario: Get a position fix on request.

Solution: Set update and search period to zero. This way the receiver stays inactive until it is woken up.

11.3 Peak current settings

The peak current during acquisition can be reduced by activating the corresponding option in CFG-PM2. A peak current reduction will result in longer start-up times of the receiver.

This setting is independent of the activated mode (Continuous or Power Save Mode).

11.4 Power On/Off command

With message RXM-PMREQ the receiver can be forced to enter *Inactive* state (in Continuous and Power Save Mode). It will stay in *Inactive* state for the time specified in the message or until it is woken up by an EXTINT or activity on the RXD1 line.

Sending the message RXM-PMREQ while the receiver is in Power Save Mode will overrule PSM and force the receiver to enter Inactive state. It will stay in Inactive state until woken up. After wake-up the receiver continues working in Power Save Mode as configured.

11.5 EXTINT pin control when Power Save Mode is not active

The receiver can be forced OFF also when Power Save Mode is not active. This works the same way as EXTINT pin control in Power Save Mode. Just as in Power Save Mode, this feature has to be enabled and configured using CFG-PM2.

11.6 Measurement and navigation rate with Power Save Mode

In Continuous Mode, measurement and navigation rate is configered using UBX-CFG-RATE. In Power Save Mode however, measurement and navigation rate can differ from the configured rates as follows:

- **Cyclic Operation**: When in state *Power Optimized Tracking*, the measurement and navigation rate is determined by the *updatePeriod* configured in CFG-PM2. The receiver can however switch to *Tracking* state (e.g. to download data). When in *Tracking* state, the measurement and navigation rate is as configured with UBX-CFG-RATE. Note: When the receiver is not able to produce position fixes anymore, it can switch from Cyclic Operation to ON/OFF Operation (if this is not disabled with the *doNotEnterOff* switch in CFG-PM2). In that case the remarks below are relevant.
- **ON/OFF Operation**: When in state *Acquisition*, the measurement and navigation rate is **fixed to 2Hz**. All NMEA (an possibly UBX) messages that are output upon a navigation fix are also output with a rate of 2Hz.

This must be considered when choosing the baud rate of a receiver that uses Power Save Mode! Note that a receiver might stay in *Acquisition* state for quite some time (can be tens of seconds under weak signal conditions). When the receiver eventually switches to *Tracking* state, the measurement and navigation rate will be as configured with UBX-CFG-RATE.

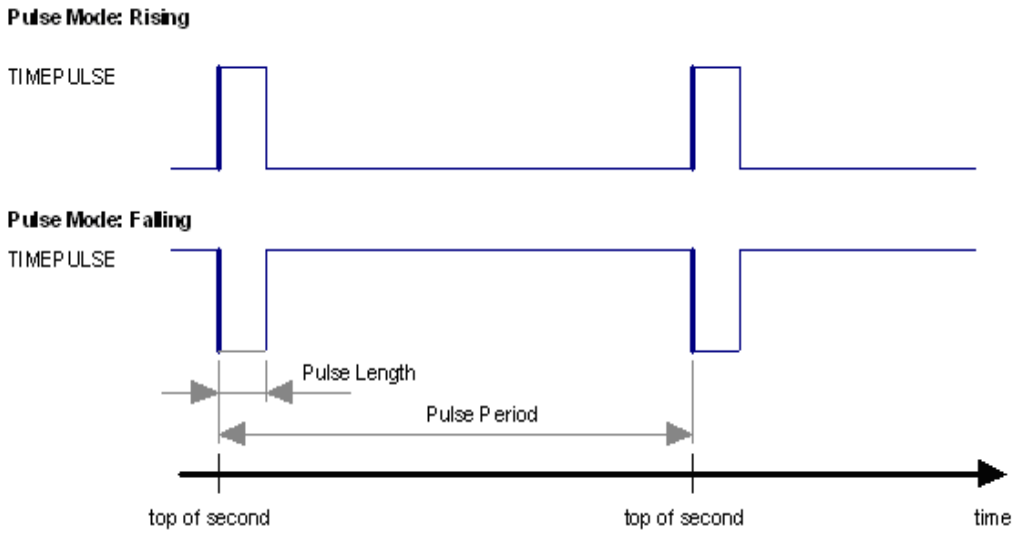
When using Power Save Mode, the baudrate of the receiver must be choosen such that it can handle the amount of data that is output when measurement and navigation rate is 2Hz.

12 Time pulse

There is only limited support for the generation of time pulses when running in GLONASS mode. In particular the accuracy of the time pulse in GLONASS mode has not been calibrated.

12.1 Introduction

u-blox GNSS receivers include a time pulse function providing clock pulses with configurable duration and frequency. The time pulse function can be configured using the CFG-TP5 message. The TIM-TP message provides time information for the next pulse, time source and the quantization error of the output pin.



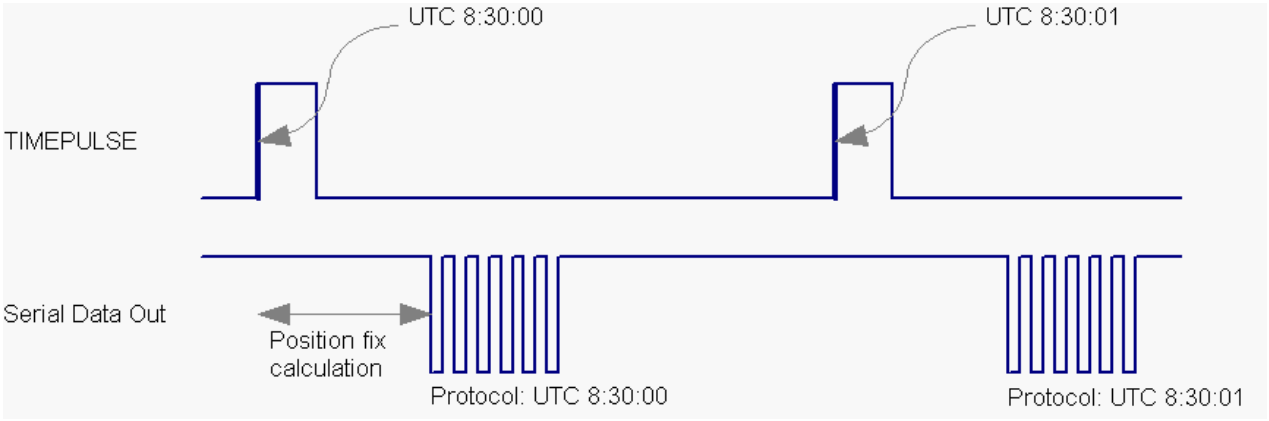
12.2 Recommendations

- For best time pulse performance it is recommended to disable the SBAS subsystem.
- When using time pulse for precision timing applications it is recommended to calibrate the RF signal delay against a reference-timing source.
- Care needs to be given to the cable delay settings in the receiver configuration.
- In order 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 by accordingly setting cable delay and user delay.
- The recommended configuration when using the TIM-TP message is to set both the measurement rate (CFG-RATE) and the time pulse frequency (CFG-TP5) to 1Hz.

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

TIM-TP message for each time pulse.

The sequential order of the signal present at the TIMEPULSE pin and the respective output message for the simple case of 1 pulse per second (1PPS) and a one second navigation update rate is shown in the following figure.



12.3 Time pulse configuration

u-blox GNSS receivers provide one or two TIMEPULSE pins (dependant on product variant) delivering a time pulse (TP) signal with a configurable pulse period, pulse length and polarity (rising or falling edge). Check the product data sheet for detailed specification of configurable values.

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 GPS time. Time pulse signals can be configured using the UBX proprietary message CFG-TP5.

12.4 Configuring time pulse with UBX-CFG-TP5

The UBX message CFG-TP5 can be used to change the time pulse settings, and includes the following parameters defining the pulse:

- **time pulse index** - Index of time pulse.
- **antenna cable delay** - Signal delay due to the cable between antenna and receiver.
- **RF group delay** - Signal delay in the RF module of the receiver (read-only).
- **pulse frequency/period** - Frequency or period time of the pulse.
- **pulse frequency/period lock** - Frequency or period time of the pulse, as soon as receiver has calculated a valid time from a received signal. Only used if the according flag 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 receiver has calculated a valid time from a received signal. Only used if the according flag is set to use another setting in locked mode.
- **user delay** - The cable delay from the receiver to the user device plus signal delay of any user application.
- **active** - time pulse will be active if this bit is set.
- **lock to gps freq** - Use frequency gained from GPS signal information rather than local oscillator's frequency if flag is set.
- **locked other setting** - If this bit is set, as soon as the receiver can calculate a valid time, the alternative setting is used. This mode can be used for example to disable time pulse if time is not locked, or indicate lock with different duty cycles.

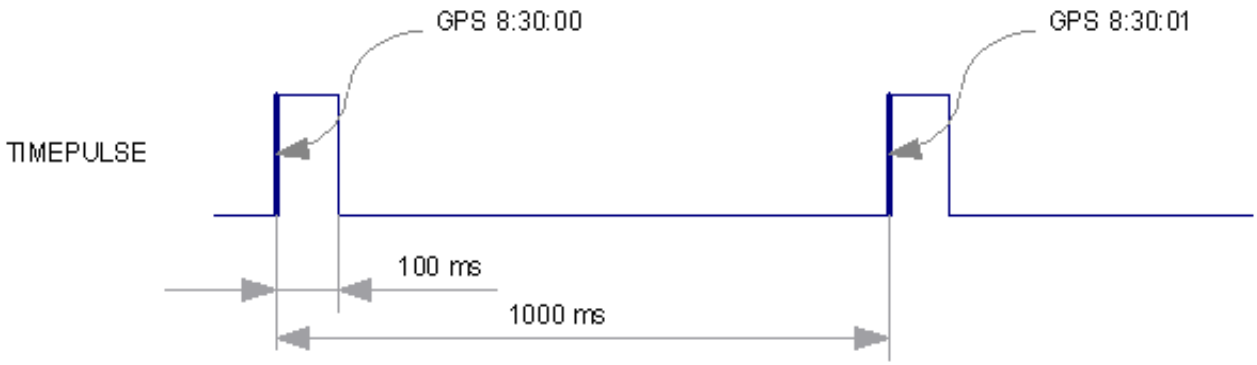
- **is frequency** - Interpret the 'Frequency/Period' field as frequency rather than period if flag is set.
- **is length** - Interpret the 'Length/Ratio' field as length rather than ratio if flag is set.
- **align to TOW** - If this bit is set, pulses are aligned to the top of a second.
- **polarity** - If set, the first edge of the pulse is a rising edge (Pulse Mode: Rising).
- **grid UTC/GPS** - Selection between UTC (0) or GPS (1) timegrid. Also effects the time output by `TIM-TP` message.

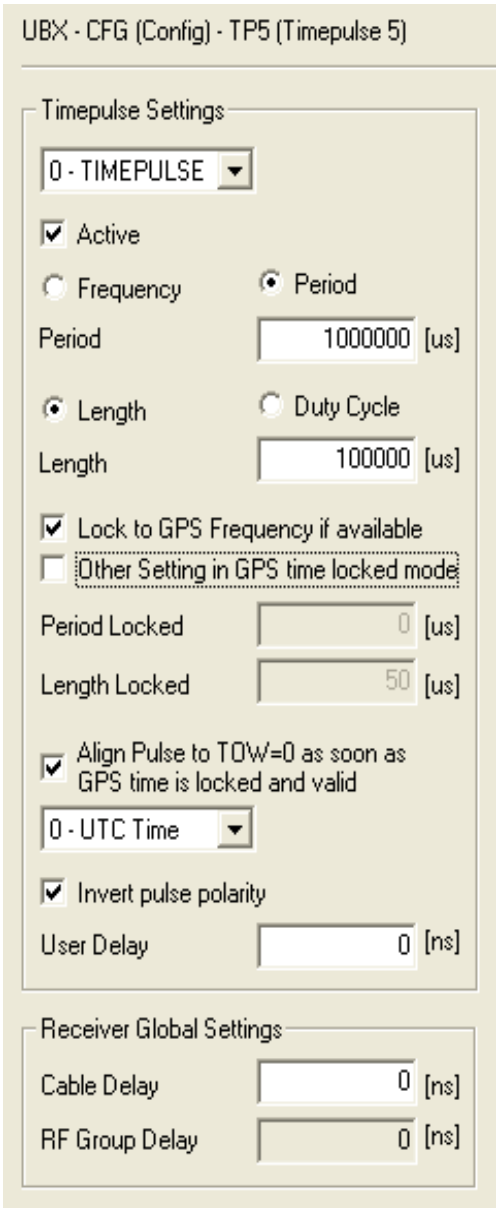
The maximum pulse length can't exceed the pulse period.

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.

12.4.1 Example 1:

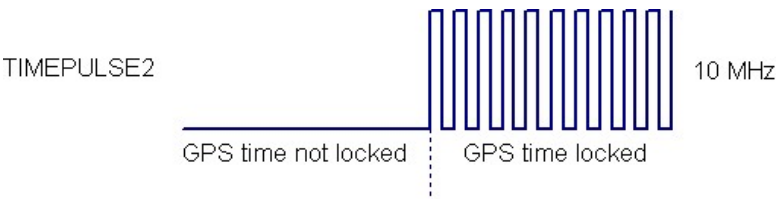
The example below shows the 1PPS TP signal generated on the time pulse output according to the specific parameters of the `CFG-TP5` message. The 1 Hz output is maintained whether or not the receiver is locked to GPS time. The alignment to TOW can only be maintained when GPS time is locked.

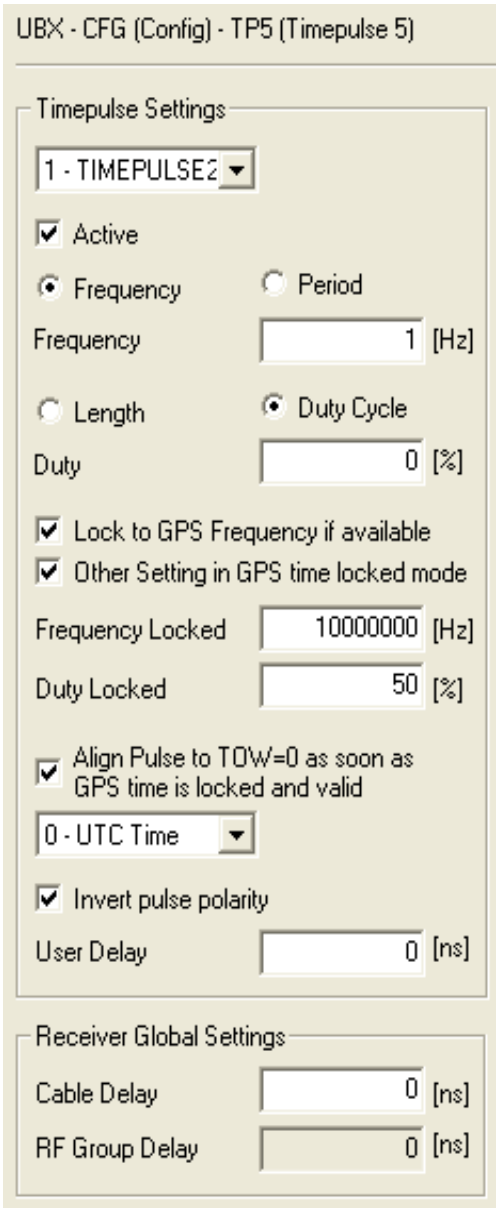




12.4.2 Example 2:

The following example shows a 10 MHz TP signal generated on the TIMEPULSE2 output when the receiver is locked to GPS time. Without the lock to GPS time no frequency is output.





13 Receiver Status Monitoring

Messages in the UBX class MON are used to report the status of the parts of the embedded computer system that are not GNSS-specific.

The main purposes are

- Hardware and Software Versions, using MON-VER
- Status of the Communications Input/Output system
- Status of various Hardware Sections with MON-HW

13.1 Input/Output system

The I/O system is a GNSS-internal layer where all data input- and output capabilities (such as UART, DDC, SPI, USB) of the GNSS receiver are combined. Each communications task has buffers assigned, where data is queued. For data originating at the receiver, to be communicated over one or multiple communications queues, the message MON-TXBUF can be used. This message shows the current and maximum buffer usage,

as well as error conditions.

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. For details see section Serial Communication Ports Description

Inbound data to the GNSS receiver is placed in buffers. Usage of these buffers is shown with the message `MON-RXBUF`. Further, as data is then decoded within the receiver (e.g. to separate UBX and NMEA data), the `MON-MSGPP` can be used. This message shows (for each port and protocol) how many messages were successfully received. It also shows (for each port) how many bytes were discarded because they were not in any of the supported protocol framings.

The following table shows the port numbers used. Note that any numbers not listed are reserved for future use.

Port Number assignment

Port #	*Electrical Interface*
0	DDC (I²C compatible)
1	UART 1
3	USB
4	SPI

Protocol numbers range from 0-7. All numbers not listed are reserved.

Protocol Number assignment

Protocol #	*Protocol Name*
0	UBX Protocol
1	NMEA Protocol

13.2 Jamming/Interference Indicator

The field `jamInd` of the `UBX-MON-HW` message can be used as an indicator for continuous wave (narrowband) jammers/interference only. The interpretation of the value depends on the application. It is necessary to run the receiver in the application and then calibrate the 'not jammed' case. If the value rises significantly above this threshold, this indicates that a continuous wave jammer is present.

This indicator is always enabled.

13.3 Jamming/Interference Monitor (ITFM)

The field `jammingState` of the `MON-HW` 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:

Jamming/Interference monitor reported states

Value	*Reported state*	*Description*
0	Unknown	Jamming/interference monitor not enabled, uninitialized or antenna disconnected
1	OK	no interference detected

Jamming/Interference monitor reported states continued

Value	*Reported state*	*Description*
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

The monitor is disabled by default. The monitor is enabled by sending an appropriate UBX-CFG-ITFM message with the **enable** bit set. 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 passive antenna.

The monitor 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.

Jamming/Interference monitor is not supported in Power Save Mode (PSM) ON/OFF mode.

14 Timemark

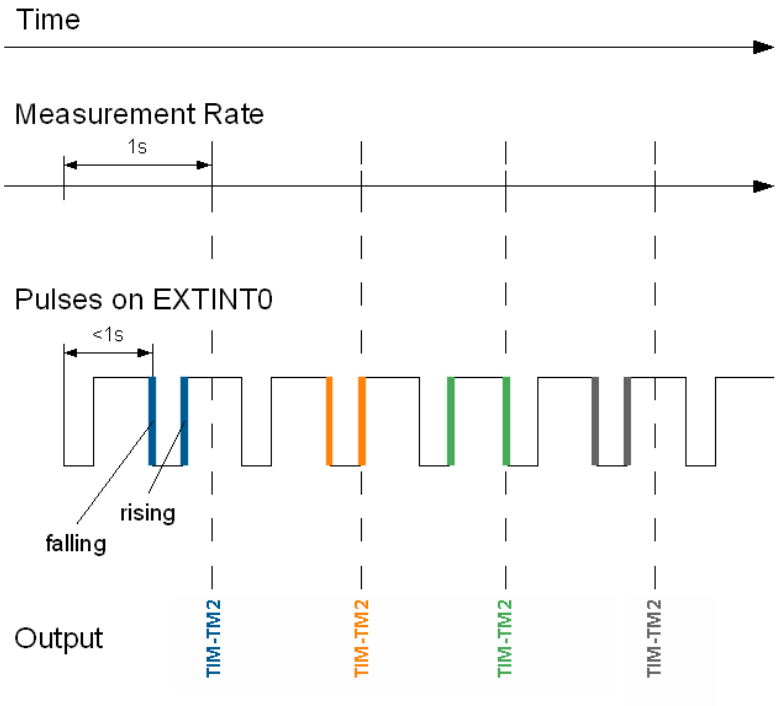
The receiver can be used to provide an accurate measurement of the time at which a pulse was detected on the external interrupt pin. The reference time can be chosen by setting the time source parameter to GPS, UTC or local time in the UBX-CFG-TP5 configuration message (using flags LockGpsFreq and gridUtcGps). The delay figures defined with UBX-CFG-TP5 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
- a rising or falling edge was triggered since last epoch on one of the EXTINT channels

The UBX-TIM-TM2 messages include time of the last timemark, new rising/falling edge indicator, time source, validity, number of marks and a quantization error. The timemark is triggered continuously.

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 UBX-CFG-RATE (see Figure below).



15 Aiding and Acquisition

15.1 Introduction

The UBX-AID message class provides the means for providing assistance data to u-blox GNSS receivers, including AssistNow Online and AssistNow Offline.

There is currently limited support for aiding of any system other than GPS. Consequently most of this section only applies to GPS operation.

15.2 Startup Strategies

- **Cold start**: In this startup mode, the receiver has no information about last position, time, velocity, frequency etc. Therefore, the receiver has to search the full time- and frequency space, and also all possible satellite numbers. If a satellite signal is found, it is being tracked to decode ephemeris (18-36 seconds under strong signal conditions), whereas the other channels continue to search satellites. Once there are sufficient number of satellites with valid ephemeris, the receiver can calculate position- and velocity data. Note that some competitors call this startup mode Factory Startup.
- **Warm start**: In Warm start mode, the receiver has approximate information of time, position, and coarse data on Satellite positions (Almanac). In this mode, after power-up, the receiver basically needs to download ephemeris until it can calculate position- and velocity data. As the ephemeris data usually is outdated after 4 hours, the receiver will typically start with a warmstart if it was powered down for more than that amount of time. For this scenario, several augmentations exist. See the sections on AssistNOW online and offline below.
- **Hot start**: In Hot start, the receiver was powered down only for a short time (4 hours or less), so that its ephemeris is still valid. Since the receiver doesn't need to download ephemeris again, this is the fastest startup method. In the `UBX-CFG-RST` message, one can force the receiver to reset and clear data, in order to see the effects of maintaining/losing such data between restarts. For that, the `UBX-CFG-RST` message

offers the navBbrMaskfield, where Hot, Warm and Cold starts can be initiated, and also other combinations thereof.

15.3 Aiding / Assisted GPS (A-GPS)

The Challenge of Stand-alone GPS

Users expect instant position information. With standard GPS this is not always possible because at least four satellites must transmit their precise orbital position data, called ephemeris, to the GPS receiver. Under adverse signal conditions, data downloads from the satellites to the receiver can take minutes, hours or even fail altogether.

Assisted GPS (A-GPS) boosts acquisition performance by providing data such as ephemeris, almanac, accurate time and satellite status to the GPS receiver via mobile networks or the Internet. The aiding data enables the receiver to compute a position within seconds, even under poor signal conditions.

15.4 Aiding Data

The following aiding data can be submitted to the receiver:

- **Position:** Position information can be submitted to the receiver using the UBX-AID-INI message. Both, ECEF X/Y/Z and latitude/longitude/height formats are supported.
- **Time:** The time can either be supplied as an inexact value via the standard communication interfaces, suffering from latency depending on the baud rate, or using hardware time synchronization where an accurate time pulse is connected to an external interrupt. Both methods are supported in the UBX-AID-INI message.
- **Frequency:** It is possible to supply hardware frequency aiding by connecting a periodic rectangular signal with a frequency up to 500 kHz and arbitrary duty cycle (low/high phase duration must not be shorter than 50 ns) to an external interrupt, and providing the applied frequency value using the UBX-AID-INI message.
- **Orbit data:** Orbit data can be submitted using UBX-AID-ALM and UBX-AID-EPH.
- **Additional information:** UBX-AID-HUI can be used to supply health information, UTC parameters and ionospheric data to the receiver.

15.5 Aiding Sequence

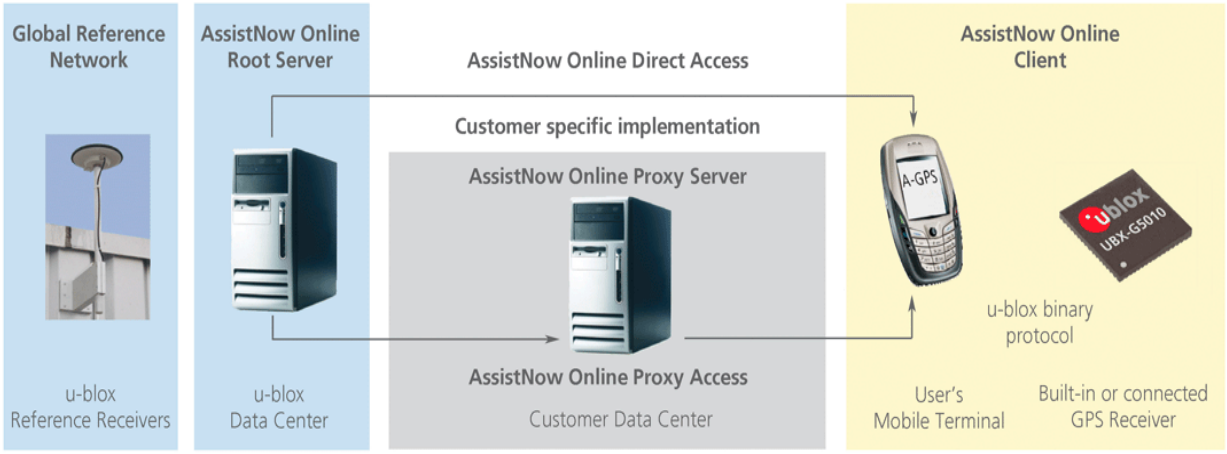
A typical aiding sequence comprises the following steps:

- Power-up the GNSS receiver
- Send UBX-AID-INI (time, clock and position) message.
- Send UBX-AID-EPH (ephemeris) message.
- Apply optional hardware time synchronization pulse within 0.5 s after (or before, depending on the configuration in UBX-AID-INI) sending the UBX-AID-INI message if hardware time synchronization is required. When sending the message before applying the pulse, make sure to allow the GNSS receiver to parse and process the aiding message. The time for parsing depends on the baud rate. The processing time is 100 ms maximum.
- Send optional UBX-AID-HUI (health, UTC and ionosphere parameters) message.
- Send optional UBX-AID-ALM (almanac) message.

15.6 AssistNow Online

AssistNow Online is u-blox' end-to-end Assisted GPS (A-GPS) solution that boosts GPS acquisition performance, bringing Time To First Fix (TTFF) down to seconds. The system works by accessing assistance data such as ephemeris, almanac and accurate time from our Global Reference Network of GNSS receivers placed around the globe. With A-GPS, the receiver can acquire satellites and provide accurate position data instantly on demand, even under poor signal conditions.

AssistNow Online makes use of User Plane communication and open standards such as TCP/IP. Therefore, it works on all standard mobile communication networks that support Internet access, including GPRS, UMTS and Wireless LAN. No special arrangements need to be made with mobile network operators to enable AssistNow Online.

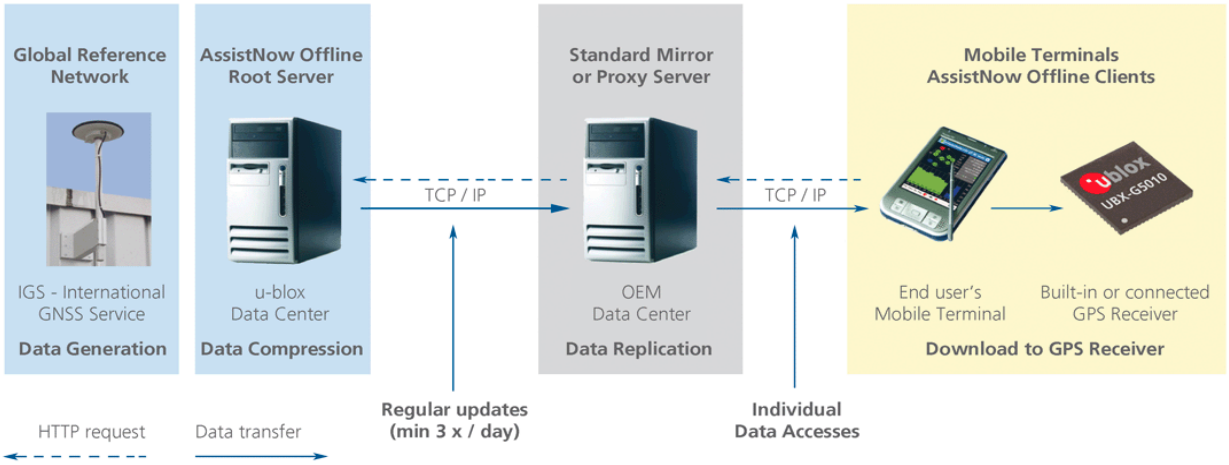


In terms of the messages AssistNow Online consists of Aiding data which deliver Position and Time `UBX-AID-INI`, Ephemerides `UBX-AID-EPH`, Almanac `UBX-AID-ALM` and Health/UTC/Iono information `UBX-AID-HUI`

AssistNow Online is the only form of aiding that currently supports GLONASS operation. Even so, GLONASS orbit data (ephemeris or almanac) it not currently supported.

15.7 AssistNow Offline

AssistNow Offline is an A-GPS service that boosts GPS acquisition performance, bringing Time To First Fix (TTFF) down to seconds. Unlike AssistNow Online, this solution enables instant positioning without the need for connectivity at start-up. The system works by using AlmanacPlus (ALP) differential almanac correction data to speed up acquisition, enabling a position fix within seconds. Users access the data by means of occasional Internet downloads, at the user's convenience.



u-blox provides AlmanacPlus (ALP) data files in different sizes, which contain differential almanac corrections that are valid for a period of between 1 and 14 days thereafter. Users can download correction data anytime they have an Internet connection. The GNSS receiver stores the downloaded data in the non-volatile memory. As an alternative, a host CPU may store the file, but deliver the data in pieces when requested.

AssistNow Offline works in locations without any wireless connectivity as the correction data files reside in the receiver or the host. This makes them immediately available upon start-up, eliminating connection set-up delays, download waiting times and call charges.

The simplest set-up is for GNSS receivers including internal non-volatile memory or an external flash memory where ALP data can be stored. In this case, the `UBX-AID-ALP` message is used.

When the receiver has neither suitable internal memory nor an external flash memory, the ALP file must be stored to the host CPU. The receiver can then request data from the host when needed. This arrangement is implemented using the `UBX-AID-ALPSRV` message.

In both cases, status reporting on ALP data currently available to the receiver can be taken from message `UBX-AID-ALP (STAT)`.

AssistNow Offline data are published at http://alp.u-blox.com/.

15.7.1 Flash-based AlmanacPlus Overview

Flash-based AlmanacPlus functionality means that AlmanacPlus data is stored in the program flash memory connected to the chip.

The task of a server is simply to download the data from an Internet server or other sources, and then deliver the full file piece by piece to the GNSS receiver. This is different to the method described in UBX-AID-ALPSRV where the file would remain within the host and the GNSS receiver would request chunks from that file when needed.

The message AID-ALP exists in several variants, combining all functionality needed to download data and report status within one Class/Message ID.

AlmanacPlus data stored in flash memory is not affected by any reset of the receiver. The only simple ways to clear it are to completely erase the whole flash memory or to overwrite it with a new set of AlmanacPlus data.

15.7.1.1 Download Procedure

The following steps are a typical sequence for downloading an ALP file to the receiver:

- The server downloads a copy of a current ALP file, and stores it locally

- It sends the first N bytes from that file, using the AID-ALP (TX) message
- The server awaits a AID-ALP (ACK) or AID-ALP (NAK) message
- If can then continue, sending the next N bytes if the message was acknowledged
- Once all data has been transferred, or a NAK has been received, the server sends an AID-ALP (STOP) message

Note that:

- N should not be larger than ~700 bytes (due to the input buffers on the RS232/USB lines). Smaller values of N might improve reliability
- N must be a multiple of 2
- There is no re-send mechanism; if a NAK message is received, the full downloading process must be restarted
- There is no explicit checksum, but an implicit one, as the ALP file already includes a checksum to verify consistency

Overview of the different versions of AID-ALP messages

Short Name	*Content*	*Direction*
AID-ALP (TX)	ALP server sends data to client	Server -> Client
AID-ALP (STOP)	ALP server terminates a transfer sequence	Server -> Client
AID-ALP (ACK)	ALP client acknowledges successful receipt of data.	Client -> Server
AID-ALP (NAK)	ALP client indicates a failed reception of data	Client -> Server
AID-ALP (STAT)	ALP client reports status of the ALP data stored in flash memory	Client -> Server

15.7.2 Host-based AlmanacPlus Overview

All three versions of AID-ALPSRV messages are used for the case where the storage of an ALP file is not within the receiver's flash memory, but on the host, and where the host needs to repeatedly deliver data to the GNSS receiver. This allows support of the AlmanacPlus functionality for GNSS receivers which do not have flash memory. For messaging details of an implementation where the data is to reside in the receiver's flash memory, see Flash-based AlmanacPlus Overview

In the following, the GNSS receiver is called the **client**, as it primarily requests data, and the host CPU where the ALP file is located in its entirety is called the **server**.

The operation is such that the client sends periodic data requests (the ALP client requests ALPSRV-REQ) to the host, and the host should answer them accordingly, as described below at ALPSRV-SRV

For this mechanism to work, the AID-ALPSRV message needs to be activated using the normal CFG-MSG commands. If it is not activated, no requests are sent out.

The client may attempt to modify the data which is stored on the server, using the ALPSRV-CLI message. The server can safely ignore such a request, in case the ALP file cannot be modified. However, for improved performance for consecutive receiver restarts, it is recommended to modify the data.

Overview of the three versions of AID-ALPSRV messages

Short Name	*Content*	*Direction*
ALPSRV-REQ	ALP client requests AlmanacPlus data from server	Client -> Server
ALPSRV-SRV	ALP server sends AlmanacPlus data to client	Server -> Client
ALPSRV-CLI	ALP client sends AlmanacPlus data to server.	Client -> Server

15.7.3 Message specifics

The three variants of this message always have a header and variable-size data appended within the same message. The first field, `idSize` gives the number of bytes where the header within the UBX payload ends and data starts.

In case of the ALP client request, the server must assemble a new message according to the `AID-ALPSRV-SRV` variant. The header needs to be duplicated for as many as `idSize` bytes. Additionally, the server needs to fill in the `fileId` and `dataSize` fields. Appended to the `idSize`-sized header, data must be added as requested by the client (from offset `ofs`, for `size` number of values).

15.7.3.1 Range checks

The server needs to perform an out-of-bounds check on the `ofs` (offsets) and `size` fields, as the client may request data beyond the actually available data. If the client request is within the bounds of available data, the `dataSize` field needs to be filled in with 2 x the content of the size field (the `size` field is in units of 16 bits, whereas the `dataSize` field expects number of bytes). If the client request would request data beyond the limits of the buffer, the data should be reduced accordingly, and this actual number of bytes sent shall be indicated in the dataSize field.

15.7.3.2 Changing ALP files

The server function periodically attempts to receive new ALP data from an upstream server, as the result of an HTTP request or other means of file transfer.

In case a new file becomes available, the server shall indicate this to the client. This is the function of the `fileId` field.

The server should number ALP files it serves arbitrarily. The only requirement is that the `fileId` actually is changed when a new file is being served, and that it does not change as long as the same file is being changed.

If the client, as a result of a client request, receives a `fileId` different from the one in earlier requests' replies, it will reinitialize the ALP engine and request data anew.

Further, if the client attempts to send data to the server, using the `ALPSRV-CLI` method, it indicates, which `fileId` needs to be written. The server shall ignore that request in case the `fileId` numbers do not match.

15.7.3.3 Sample Code

u-blox makes available sample code, written in C language, showing a server implementation, serving ALP data from its file system to a client. Please contact your nearest u-blox Field Application Engineer to receive a copy.

15.8 AssistNow Autonomous

15.8.1 Introduction

The assistance scenarios covered by *AssistNow Online* and *AssistNow Offline* require an online connection and a host that can use this connection to download aiding data and provide this to the receiver when required.

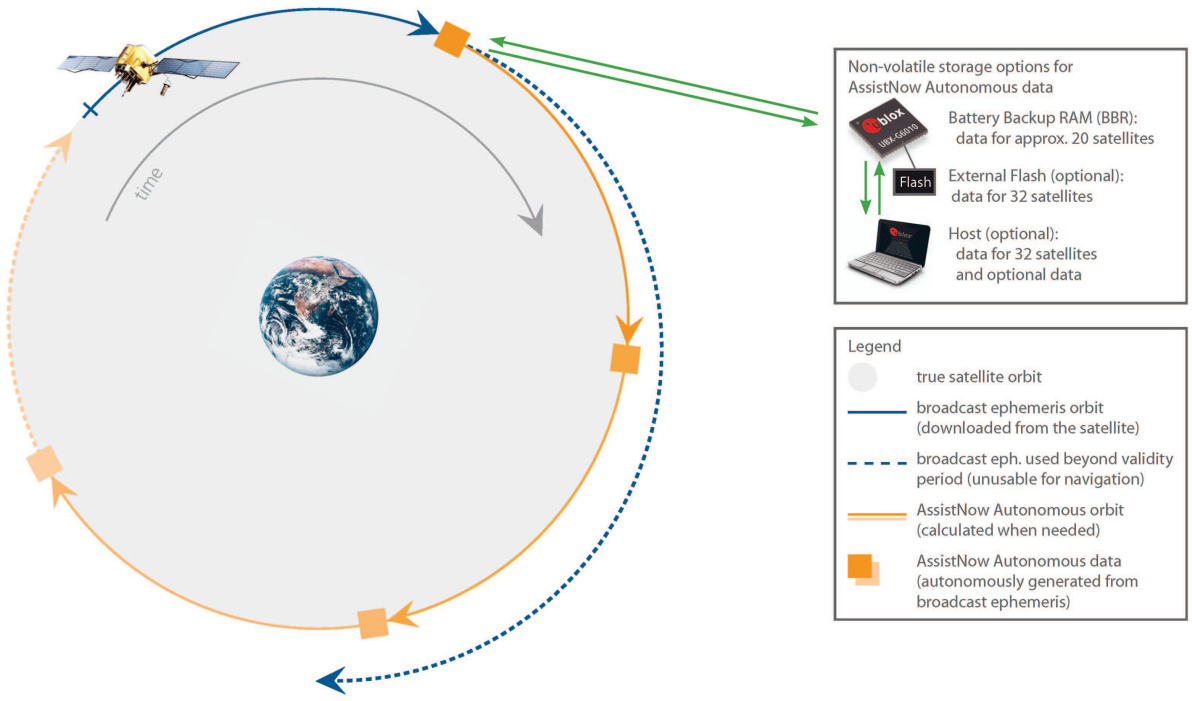
The *AssistNow Autonomous* feature provides a functionality similar to *AssistNow Offline* without the need for a host and a connection. Based on a broadcast ephemeris downloaded from the satellite (or obtained by *AssistNow Online*) the receiver can autonomously (i.e. without any host interaction or online connection) generate an accurate satellite orbit representation («AssistNow Autonomous data») that is usable for navigation much longer than the underlying broadcast ephemeris was intended for. This makes downloading new ephemeris or aiding data for the first fix unnecessary for subsequent start-ups of the receiver.

The AssistNow Autonomous feature is disabled by default. It can be enabled using the `UBX-CFG-NAVX5` message.

15.8.2 Concept

The figure below illustrates the *AssistNow Autonomous* concept in a graphical way. Note that the figure is a qualitative illustration and is not to scale.

- A broadcast ephemeris downloaded from the satellite is a precise representation of a part (nominally four hours) of the satellite's true orbit (trajectory). It is not usable for positioning beyond this validity period because it diverges dramatically from the true orbit afterwards.
- The *AssistNow Autonomous orbit* is an extension of a broadcast ephemeris. It provides a long-term orbit for the satellite for several revolutions. Although this orbit is not perfectly precise it is a sufficiently accurate representation of the true orbit to be used for navigation.
- The *AssistNow Autonomous data* is automatically and autonomously generated from downloaded (or assisted) ephemerides. The data is stored automatically in the on-chip battery-backed memory. Optionally, the data can be backed-up in external flash memory or on the host. The number of satellites for which data can be stored depends on the receiver configuration and may change during operation.
- If no broadcast ephemeris is available for navigation *AssistNow Autonomous* automatically generates the required parts of the orbits suitable for navigation from the stored data. The data is also automatically kept current in order to minimize the calculation time once the navigation engine needs orbits.
- The operation of the *AssistNow Autonomous* feature is transparent to the user and the operation of the receiver. All calculations are done in background and do not affect the normal operation of the receiver.
- The *AssistNow Autonomous* subsystem automatically invalidates data that has become too old and that would introduce unacceptable positioning errors. This threshold is configurable (see below).
- The *AssistNow Autonomous* can automatically improve the prediction quality if the receiver can download a broadcast ephemeris of a previously seen satellite 24 hours later.



15.8.3 Interface

Several UBX protocol messages provide interfaces to the *AssistNow Autonomous* feature. They are:

- The UBX-CFG-NAVX5 message is used to enable or disable the *AssistNow Autonomous* feature. It is disabled by default. Once enabled, the receiver will automatically produce *AssistNow Autonomous* data for newly received broadcast ephemerides and, if that data is available, automatically provide the navigation subsystem with orbits when necessary and adequate. The message also allows for a configuration of the maximum acceptable orbit error. See the next section for an explanation of this feature. It is recommended to use the firmware default value that corresponds to an orbit data validity of approximately three days.
- The UBX-NAV-AOPSTATUS message provides information on the current state of the *AssistNow Autonomous* subsystem as well as on the availability of *AssistNow Autonomous* data for individual GPS satellites. The status indicates whether the *AssistNow Autonomous* subsystem is currently idle (or not enabled) or busy generating data or orbits. Hosts should monitor this information and only power-off the receiver when the subsystem is idle (that is, when the `status` field shows a steady zero).
- The UBX-NAV-SVINFO message indicates the use of *AssistNow Autonomous* orbits for individual satellites.

Two means to preserve *AssistNow Autonomous* data in power-off mode where no battery backup is available are provided:

- Saving all data (including configuration, orbits, etc.) to flash where available.
- Polling all required data and configuration from the receiver and saving it on the host and store it back to the receiver on startup.. This can be achieved using the UBX-AID-AOP (required *AssistNow Autonomous* data), UBX-AID-ALM (almanac, recommended for best performance), and UBX-AID-HUI (required UTC time information) messages. Note that the UBX-AID-AOP message can contain additional (optional) data that is not stored in the battery backup RAM due to space limitations. This additional data helps the receiver to carry out some calculations faster than without it. It does not, however, affect the orbit quality. Hence, the optional data may be stripped from the message payload if, for example, host storage capacity is limited. Sending (a) valid `UBX-AID-AOP` message(s), to the receiver will automatically enable the *AssistNow Autonomous* feature. Furthermore, it is recommended to use high baud rates on serial interfaces when polling and sending this message due to its relatively large size.

Note that the receiver requires the absolute time (i.e. full Date and Time) to calculate *AssistNow Autonomous* orbits. For best performance it is, therefore, recommended to supply this information to the receiver using the UBX-AID-INI message in a scenario without a running RTC (i.e. without backup battery).

15.8.4 Benefits and Drawbacks

AssistNow Autonomous can provide quicker start-up times (lower the TTFF) provided that data is available for enough visible satellites. This is particularly true under weak signal conditions where it might not be possible to download broadcast ephemerides at all, and, therefore, no fix at all would be possible without *AssistNow Autonomous* (or A-GPS). It is, however, required that the receiver roughly know the absolute time, either from an RTC or from time-aiding using the UBX-AID-INI message, and that it knows which satellites are visible, either from the almanac or from tracking the respective signals.

The *AssistNow Autonomous* orbit (satellite position) accuracy depends on various factors, such as the particular type of satellite, the accuracy of the underlying broadcast ephemeris, or the orbital phase of the satellite and Earth, and the age of the data (errors add up over time).

AssistNow Autonomous will typically extend a broadcast ephemeris for up to three days. The UBX-CFG-NAVX5 (see above) message allows to change this threshold by setting the «maximum acceptable modelled orbit error» (in meters). Note that this number does not reflect the true orbit error introduced by extending the ephemeris. It is a statistical value that represents a certain expected upper limit based on a number of parameters. A rough approximation that relates the maximum extension time to this setting is: *maxError [m]* =

*maxAge [d] * f*, where the factor *f* is 30 for data derived from satellites seen once and and 17 for data derived for satellites seen more than once.

There is no direct relation between (true and statistical) orbit accuracy and positioning accuracy. The positioning accuracy depends on various factors, such as the satellite position accuracy, the number of visible satellites, and the geometry (DOP) of the visible satellits. Position fixes that include *AssistNow Autonomous* orbit information may be significantly worse than fixes using only broadcast ephemerides. It might be necessary to adjust the limits of the Navigation Output Filters.

A fundamental deficiency of any system to predict satellite orbits precisely is unknown future events. Hence, the receiver will not be able to know about satellites that will have become unhealthy, have undergone a clock swap, or have had a manoeuvre. This means that the navigation engine might rarely mistake a wrong satellite position as the true satellite position. However, provided that there are enough other good satellites, the navigation algorithms will eventually eliminate a defective orbit from the navigation solution.

The repeatability of the GPS satellite constellation is a potential pitfall for the use of the *AssistNow Autonomous* feature. For a given location on Earth the constellation (geometry of visible satellites) repeats every 24 hours. Hence, when the receiver «learned» about a number of satellites at some point in time the same satellites will in most places *not* be visible 12 hours later, and the available *AssistNow Autonomous* data will not be of any help. Again 12 hours later, however, usable data would be available because it had been generated 24 hours ago.

The longer a receiver observes the sky the more satellites it will have seen. At the equator, and with full sky view, approximately ten satellites will show up in a one hour window. After four hours of observation approx. 16 satellites (i.e. half the constellation), after 10 hours approx. 24 satellites (2/3rd of the constellation), and after approx. 16 hours the full constellation will have been observed (and *AssistNow Autonomous* data generated for). Lower sky visibility reduces these figures. Further away from the equator the numbers improve because the satellites can be seen twice a day. E.g. at 47 degrees north the full constellation can be observed in approx. 12 hours with full sky view.

The calculations required for *AssistNow Autonomous* are carried out on the receiver. This requires energy and users may therefore occasionally see increased power consumption during short periods (several seconds, rarely more than 60 seconds) when such calculations are running. Ongoing calculations will automatically prevent the power save mode from entering the power-off state. The power-down will be delayed until all calculations are done.

The AssistNow Offline and AssistNow Autonomous features are exclusive and must not be used at the same time.

16 Precise Point Positioning

This feature is only available with the PPP product variant

16.1 Introduction

Precise Point Positioning (PPP) is a product variant which offers enhanced positioning accuracy by utilizing the carrier phase measurements to smooth the pseudoranges measured to the satellites. The algorithm needs continuous carrier phase measurements to be able to smooth the pseudorange measurements effectively. Additionally ionospheric corrections like those received from SBAS or from GPS are required. A positioning improvement can only be expected in an environment with unobstructed sky view during a period on the order of minutes.

The PPP algorithm works for GPS satellites only and SBAS corrections are required to provide enhanced positioning accuracy.

16.2 Configuration

In order to use the *Precise Point Positioning* algorithm, PPP must be enabled/disabled by setting the appropriate flag in UBX-CFG-NAVX5.

PPP can only be activated on Precise Point Positioning product variants, where it is activated by default.

While valid RTCM corrections are provided to the receiver, the *Precise Point Positioning* algorithm will not operate. The Precise Point Positioning algorithm will restart after the last valid RTCM correction has expired.

16.3 Monitoring

The message UBX-NAV-SVINFO indicates for each satellite in use whether or not the pseudorange has been smoothed by the PPP algorithm.

17 Logging

17.1 Introduction

The logging feature allows position fixes and arbitrary byte strings from the host to be logged in flash memory attached to the receiver. Logging of position fixes happens independently of the host system, and can continue while the host is powered down.

The following tables list all the logging related messages:

Logging control and configuration messages

Message	*Description*
UBX-LOG-CREATE	Creates a log file and activates the logging subsystem
UBX-LOG-ERASE	Erases a log file and deactivates the logging subsystem
UBX-CFG-LOGFILTER	Used to start/stop recording and set/get the logging configuration
UBX-LOG-INFO	Provides information about the logging system
UBX-LOG-STRING	Enables a host process to write a string of bytes to the log file

Logging retrieval messages

Message	*Description*
UBX-LOG-RETRIEVE	Starts the log retrieval process
UBX-LOG-RETRIEVEPOS	A position log entry returned by the reciever
UBX-LOG-RETRIEVESTRING	A byte string log entry returned by the reciever
UBX-LOG-FINDTIME	Finds the index of the first entry <= given time

17.2 Setting the logging system up

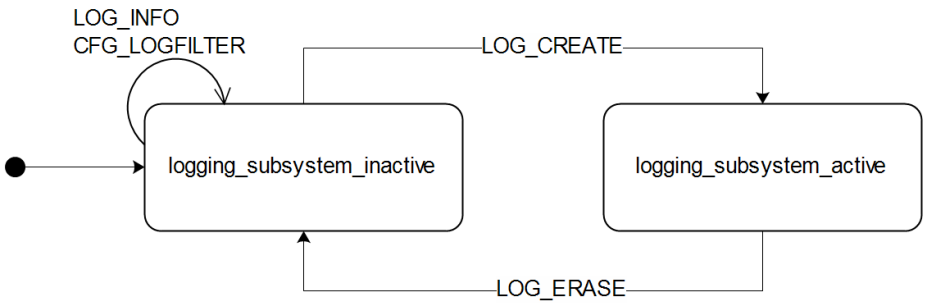
An empty log can be created using the UBX-LOG-CREATE message and a log can be deleted with the UBX-LOG-ERASE message. The logging system will only be running if a log is in existence, so most logging messages will be rejected with an UBX-ACK-NAK message if there is no log present. Only one log can be created at any one time so an UBX-ACK-NAK message will be returned if a log already exists. The message specifies the maximum size of the log in bytes (with some pre-set values provided). Both the logging subsystem and the receiver filestore have implementation overheads, so total space available for log entries will be somewhat smaller than the size specified.

UBX-LOG-CREATE also allows the log to be specified as a circular log. If the log is circular, then when it fills up, a set of older log entries will be deleted and the space freed up used for new log entries. By contrast, if a non-circular log becomes full then new entries which don't fit will be rejected. UBX-LOG-CREATE also causes the logging system to start up so that further logging messages can be processed. The logging system will start

up automatically on power-up if there is a log in existence. The log will remain in the receiver until specifically erased using the UBX-LOG-ERASE message.

UBX-CFG-LOGFILTER controls whether logging of entries is currently enabled and selects position fix messages for logging. These configuration settings will be saved if the configuration is saved to flash. If this is done, then entry logging will continue on power-up in the same manner that it did before power-down.

The top level active/inactive states of the logging subsystem.



17.3 Information about the log

The receiver can be polled for a UBX-LOG-INFO message which will give information about the log. This will include the maximum size that the log can grow to (which, due to overheads, will be smaller than that requested in UBX-LOG-CREATE) and the amount of log space currently occupied. It will also report the number of entries currently in the log together with the time and date of the newest and oldest messages which have a valid timestamp.

Log entries are compressed and have housekeeping information associated with them, so the actual space occupied by log messages may be difficult to predict. The minimum size for a position fix entry is 9 bytes and the maximum 24 bytes, the typical size is 10 or 11 bytes.

Each log also has a fixed overhead which is dependent on the log type. The approximate size of this overhead is shown in the following table.

Log overhead size

Log type	*Overhead*
circular	Up to 40 kB
non-circular	Up to 8 kB

The number of entries that can be logged in any given flash size can be estimated as follows:

```
Approx. number of entries = (flash size available for logging - log
overhead)/typical entry size
```

For example, if 1500 kB of flash is available for logging (after other flash usage such as the firmware image is taken into account) a non-circular log would be able to contain approximately 139000 entries ((1500*1024)-(8*1024))/11 = 138891.

17.4 Recording

The UBX-CFG-LOGFILTER message specifies the conditions under which entries are recorded. Nothing will be recorded if recording is disabled, otherwise position fix and UBX-LOG-STRING entries can be recorded. When recording is enabled an entry will also be created from each UBX-LOG-STRING message. These will be timestamped if the receiver has current knowledge of time.

The UBX-CFG-LOGFILTER message has several values which can be used to select position fix entries for logging. If all of these values are zero, then all position fixes will be logged (subject to a maximum rate of 1Hz). A position is logged if any of the thresholds are exceeded. If a threshold is set to zero it is ignored. In addition

the position difference and current speed thresholds also have a minimum time threshold.

Position fixes are only recorded if a valid fix is obtained - failed and invalid fixes are not recorded.

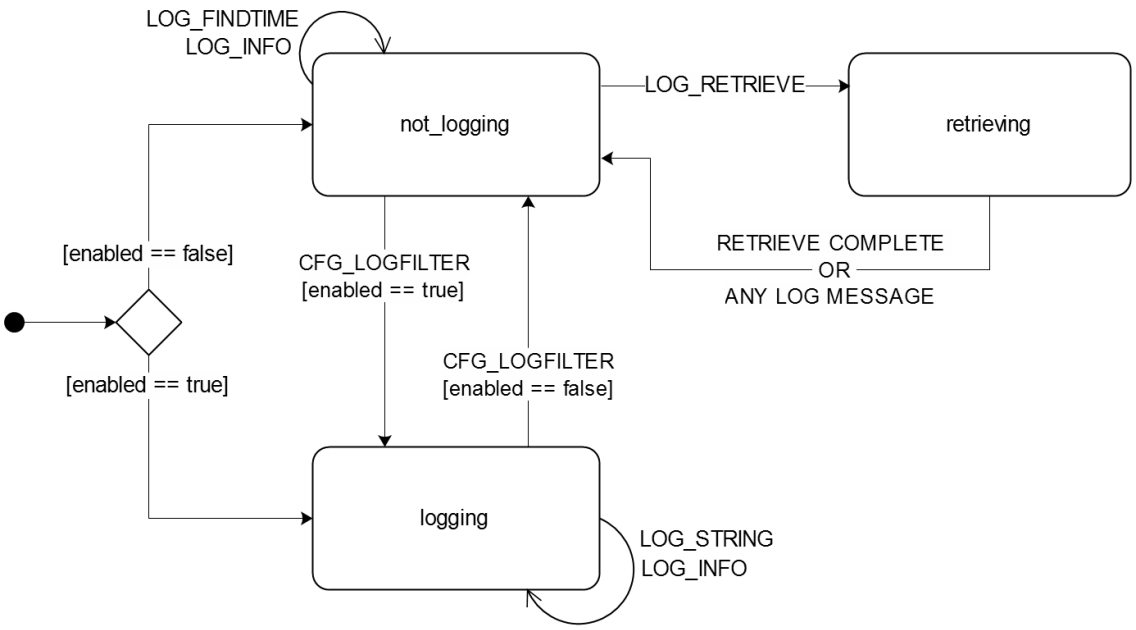
Position fixes are compressed to economise on the amount of flash space used. In order to improve the compression, the fix values are rounded to improve their compression. This means that the values returned by the logging system may differ slightly from any which are gathered in real time.

In On/Off Power Save Mode it is possible to configure the logging system so that only one fix is recorded for each on period. This will be recorded immediately before the receiver powers off and will be the best fix seen during the on period (in this case, "best" is defined as being the fix with the lowest horizontal accuracy figure).

The recorded data for a fix comprises :

- The time and date of the fix recorded to a precision of one second
- Latitude and longitude to a precision of one millionth of a degree. Depending on position on Earth this is a precision in the order of 0.1m
- Altitude (height above mean sea level) to a precision of 1m
- Ground speed to a precision of 1cm/s
- The fix type (only successful fix types, since these are the only ones recorded)
- The number of satellites used in the fix is recorded, but no value greater than 19 is logged; a value of 19 means 19 or more satellites
- A horizontal accuracy estimate is recorded to give an indication of fix quality
- Heading to a precision of one degree

The states of the active logging subsystem



17.5 Retrieval

UBX-LOG-RETRIEVE starts the process which allows the receiver to output log entries. Log recording must be stopped using UBX-CFG-LOGFILTER before this can be done. UBX-LOG-INFO may be helpful to a host system in order to understand the current log status before retrieval is started.

Once retrieval has started, one message will be output from the receiver for each log entry requested. Sending any logging message to the receiver during retrieval will cause the retrieval to stop before the message is processed.

To maximise the speed of transfer it is recommended that a high communications data rate is used and GNSS processing is stopped during the transfer (see UBX-CFG-RST)

UBX-LOG-RETRIEVE can specify a start-entry index and entry-count. The maximum number of entries that can be returned in response to a single UBX-LOG-RETRIEVE message is 256. If more entries than this are required the message will need to be sent multiple times with different startEntry indicies.

The receiver will send a UBX-LOG-RETRIEVEPOS message for each position fix log entry and a UBX-LOG-RETRIEVESTRING message for each string log entry. Messages will be sent in the order in which they were logged, so UBX-LOG-RETRIEVEPOS and UBX-LOG-RETRIEVESTRING messages may be interspersed in the message stream.

The UBX-LOG-FINDTIME message can be used to search a log for the index of the first entry less than or equal to the given time. This index can then be used with the UBX-LOG-RETRIEVE message to provide time-based retrieval of log entries.

17.6 Command message acknowledgement

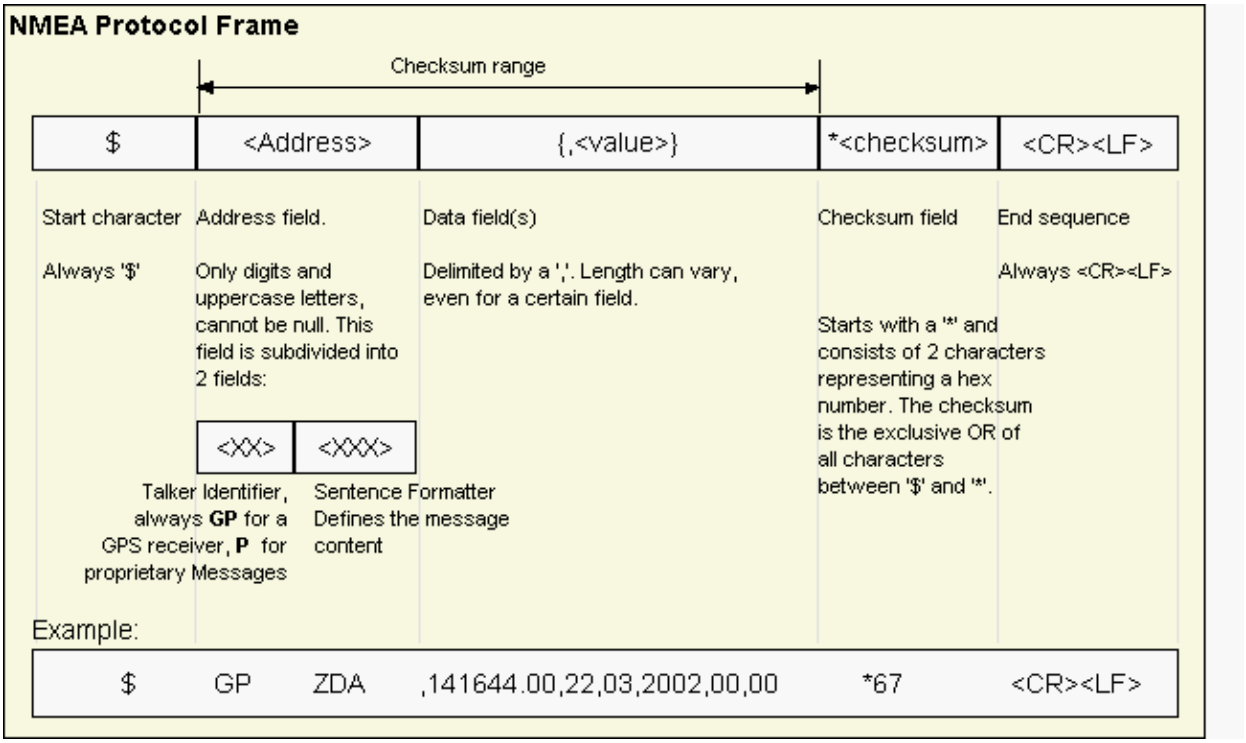
Some log operations make take a long time to execute because of the time taken to write to flash memory. The time for some operations may be unpredictable since the number and timing of flash operations may vary. In order to allow host software to synchronise to these delays logging messages will always produce a response. This will be UBX-ACK-NAK in case of error, otherwise UBX-ACK-ACK unless there is some other defined response to the message.

It is possible to send a small number of logging commands without waiting for acknowledgement, since there is a command queue, but this risks confusion between the acknowledgements for the commands. Also a command queue overflow would result in commands being lost.

NMEA Protocol

18 Protocol Overview

NMEA messages sent by the GNSS receiver are based on NMEA 0183 Version 2.3. The following picture shows the structure of a NMEA protocol message.



For further information on the NMEA Standard please refer to *NMEA 0183 Standard For Interfacing Marine Electronic Devices*, Version 2.30, March 1, 1998. See http://www.nmea.org/ for ordering instructions.

The NMEA standard allows for proprietary, manufacturer-specific messages to be added. These shall be marked with a manufacturer mnemonic. The mnemonic assigned to u-blox is UBX and is used for all non-standard messages. These proprietary NMEA messages therefore have the address field set to PUBX. The first data field in a PUBX message identifies the message number with two digits.

19 NMEA Protocol Configuration

The `NMEA protocol` on u-blox receivers can be configured to the need of customer applications using `CFG-NMEA`.

There are two NMEA standards supported. The default NMEA version is 2.3. Alternatively version 2.1 can be enabled (for details on how this affects the output refer to section `Position Fix Flags in NMEA Mode`).

The NMEA standard differentiates between GPS, GLONASS, and combined GNSS receivers using a two-letter message identifier, the 'Talker ID'. Depending upon device model and system configuration, the u-blox receiver could output messages using any one of these Talker IDs.

By default, receivers configured to support GPS, SBAS and QZSS use the 'GP' Talker ID, receivers configured to support GLONASS use the 'GL' Talker Id, and receivers configured for any other GNSS or any other combinations of GNSS use the 'GN' Talker ID

NMEA defines a satellite numbering system for GPS, SBAS, and GLONASS. Satellite numbers for other GNSS can be configured using `CFG-NMEA`. Unknown satellite numbers are always reported as a null NMEA field (i.e.

an empty string)

The NMEA specification indicates that the GGA message is GPS specific. However, u-blox recievers support the output of a GGA message for each of the Talker IDs.

NMEA filtering flags

Parameter	*Description*
Position filtering	Enable to permit positions from failed or invalid fixes to be reported (with the "V" status flag to indicate that the data is not valid).
Valid position filtering	Enable to permit positions from invalid fixes to be reported (with the "V" status flag to indicate that the data is not valid).
Time filtering	Enable to permit the receiver's best knowledge of time to be output, even though it might be wrong.
Date filtering	Enable to permit the receiver's best knowledge of date to be output, even though it might be wrong.
GPS-only filtering	Enable to restrict output to only report GPS satellites.
Track filtering	Enable to permit course over ground (COG) to be reported even when it would otherwise be frozen.

NMEA flags

Parameter	*Description*
Compatibility Mode	Some older NMEA applications expect the NMEA output to be formatted in a specific way, for example, they will only work if the latitude and longitude have exactly four digits behind the decimal point. u-blox receivers offer a compatibility mode to support these legacy applications.
Consideration Mode	u-blox receivers use a sophisticated signal quality detection scheme, in order to produce the best possible position output. This algorithm considers all SV measurements, and may eventually decide to only use a subset thereof, if it improves the overall position accuracy. If Consideration mode is enabled, all satellites, which were considered for navigation, are communicated as being used for the position determination. If Consideration Mode is disabled, only those satellites which after the consideration step remained in the position output are marked as being used.

Extended configuration

Option	*Description*
GNSS to filter	Filters satellites based on their GNSS
Satellite numbering	This field configures the display of satellites that do not have an NMEA-defined value. Note: this does not apply to satellites with an unknown ID.
Main Talker ID	By default the main Talker ID (i.e. the Talker ID used for all messages other than GSV) is determined by the GNSS assignment of the receiver's channels (see `UBX-CFG-GNSS`). This field enables the main Talker ID to be overridden.
GSV Talker ID	By default the Talker ID for GSV messages is GNSS specific (as defined by NMEA). This field enables the GSV Talker ID to be overridden.

20 Latitude and Longitude Format

According to the NMEA Standard, Latitude and Longitude are output in the format Degrees, Minutes and (Decimal) Fractions of Minutes. To convert to Degrees and Fractions of Degrees, or Degrees, Minutes, Seconds and Fractions of seconds, the 'Minutes' and 'Fractional Minutes' parts need to be converted. In other words: If the GPS Receiver reports a Latitude of 4717.112671 North and Longitude of 00833.914843 East, this is

Latitude 47 Degrees, 17.112671 Minutes

Longitude 8 Degrees, 33.914843 Minutes

or

Latitude 47 Degrees, 17 Minutes, 6.76026 Seconds

Longitude 8 Degrees, 33 Minutes, 54.89058 Seconds

or

Latitude 47.28521118 Degrees

Longitude 8.56524738 Degrees

21 Position Fix Flags in NMEA

This section shows how u-blox implements the NMEA protocol and the conditions determining how flags are set.

Flags in NMEA 2.3 and above

NMEA Message: Field	*No position fix (at power-up, after losing satellite lock)*	*GNSS fix, but user limits exceeded*	*Dead reckoning fix, but user limits exceeded*	*Dead reckoning fix (ADR with external sensors, linear extrapolation, or map matching)*	*2D GNSS fix*	*3D GNSS fix*	*Combined GNSS/dead reckoning fix (ADR with external sensors)*
GLL, RMC: status	V	V	V	A	A	A	A
	V=Data Invalid, A=Data Valid						
GGA: quality	0	0	6	6	1 / 2	1 / 2	1 / 2
	0=No Fix, 1=Autonomous GNSS Fix, 2=Differential GNSS Fix, 6=Estimated/Dead Reckoning Fix						
GSA: navMode	1	1	2	2	2	3	3
	1=No Fix, 2=2D Fix, 3=3D Fix						
GLL, RMC, VTG, GNS: posMode	N	N	E	E	A / D	A / D	A / D
	N=No Fix, E=Estimated/Dead Reckoning Fix, A=Autonomous GNSS Fix, D=Differential GNSS Fix						

Flags in NMEA 2.1 and below

The flags in NMEA 2.1 and below are the same as NMEA 2.3 and above but with the following differences:

- The posMode field is not output for GLL, RMC and VTG messages (each message has one field less).
- The GGA quality field is set to 1 (instead of 6) For both types of dead reckoning fix.

22 Ouput of invalid/unknown data

By default the receiver will not output invalid data. In such cases, it will output empty fields.

A valid position fix is reported as follows:

```
$GPGLL,4717.11634,N,00833.91297,E,124923.00,A,A*6E
```

An invalid position fix (but time valid) is reported as follows:

```
$GPGLL,,,,,124924.00,V,N*42
```

If Time is unknown (e.g. during a cold-start):

```
$GPGLL,,,,,,V,N*64
```

Please note:

An exception from the above default are dead reckoning fixes, which are also output when invalid (user limits exceeded).

Output of invalid data marked with the 'Invalid/Valid' Flags can be enabled using the UBX protocol message CFG-NMEA.

Differing from the NMEA standard, u-blox reports valid dead reckoning fixes with user limits met (not exceeded) as valid (A) instead of invalid (V).

23 NMEA Messages Overview

When configuring NMEA messages using the UBX protocol message CFG-MSG, the Class/Ids shown in the table shall be used.

Page	*Mnemonic*	*Cls/ID*	*Description*
NMEA Standard Messages			**Standard Messages**
52	**DTM**	0xF0 0x0A	Datum Reference
53	**GBS**	0xF0 0x09	GNSS Satellite Fault Detection
54	**GGA**	0xF0 0x00	Global positioning system fix data
55	**GLL**	0xF0 0x01	Latitude and longitude, with time of position fix and status
56	**GLQ**	0xF0 0x43	Poll a standard message (if the current Talker ID is GL)
56	**GNQ**	0xF0 0x42	Poll a standard message (if the current Talker ID is GN)
57	**GNS**	0xF0 0x0D	GNSS fix data
58	**GPQ**	0xF0 0x40	Poll a standard message (if the current Talker ID is GP)
58	**GRS**	0xF0 0x06	GNSS Range Residuals
59	**GSA**	0xF0 0x02	GNSS DOP and Active Satellites
60	**GST**	0xF0 0x07	GNSS Pseudo Range Error Statistics
61	**GSV**	0xF0 0x03	GNSS Satellites in View
62	**RMC**	0xF0 0x04	Recommended Minimum data
63	**TXT**	0xF0 0x41	Text Transmission
64	**VTG**	0xF0 0x05	Course over ground and Ground speed
65	**ZDA**	0xF0 0x08	Time and Date
NMEA PUBX Messages			**Proprietary Messages**
66	**CONFIG**	0xF1 0x41	Set Protocols and Baudrate
67	**POSITION**	0xF1 0x00	Poll a PUBX,00 message
67	**POSITION**	0xF1 0x00	Lat/Long Position Data
69	**RATE**	0xF1 0x40	Set NMEA message output rate
70	**SVSTATUS**	0xF1 0x03	Poll a PUBX,03 message
70	**SVSTATUS**	0xF1 0x03	Satellite Status
71	**TIME**	0xF1 0x04	Poll a PUBX,04 message
72	**TIME**	0xF1 0x04	Time of Day and Clock Information

24 Standard Messages

Standard Messages: i.e. Messages as defined in the NMEA Standard.

24.1 DTM

24.1.1 Datum Reference

Message	**DTM**		
Description	**Datum Reference**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Output Message		
Comment	This message gives the difference between the current datum and the reference datum. The current datum defaults to WGS84 The reference datum cannot be changed and is always set to WGS84.		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF0 0x0A	11	

Message Structure:

```
$xxDTM,datum,subDatum,lat,NS,lon,EW,alt,refDatum*cs<CR><LF>
```

Example:

```
$GPDTM,W84,,0.0,N,0.0,E,0.0,W84*6F
$GPDTM,999,,0.08,N,0.07,E,-47.7,W84*1C
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	`xxDTM`	-	string	$GPDTM	DTM Message ID (xx = current Talker ID)
1	`datum`	-	string	W84	Local datum code: W84 = WGS84, 999 = user defined
2	`subDatum`	-	string	-	A null field
3	`lat`	min	numeric	0.08	Offset in Latitude
4	`NS`	-	character	S	North/South indicator
5	`lon`	min	numeric	0.07	Offset in Longitude
6	`EW`	-	character	E	East/West indicator
7	`alt`	m	numeric	-2.8	Offset in altitude
8	`refDatum`	-	string	W84	Reference datum code (always W84 = WGS 84)
9	`cs`	-	hexadecimal	*67	Checksum
10	`<CR><LF>`	-	character	-	Carriage return and line feed

24.2 GBS

24.2.1 GNSS Satellite Fault Detection

Message	**GBS**		
Description	**GNSS Satellite Fault Detection**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Output Message		
Comment	This message outputs the results of the Receiver Autonomous Integrity Monitoring Algorithm (RAIM). • The fields **errLat**, **errLon** and **errAlt** output the standard deviation of the position calculation, using all satellites which pass the RAIM test successfully. • The fields **errLat**, **errLon** and **errAlt** are only output if the RAIM process passed successfully (i.e. no or successful edits happened). These fields are never output if 4 or fewer satellites are used for the navigation calculation (because, in such cases, integrity can not be determined by the receiver autonomously). • The fields **prob**, **bias** and **stdev** are only output if at least one satellite failed in the RAIM test. If more than one satellites fail the RAIM test, only the information for the worst satellite is output in this message.		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF0 0x09	11	

Message Structure:

```
$xxGBS,time,errLat,errLon,errAlt,svid,prob,bias,stddev*cs<CR><LF>
```

Example:

```
$GPGBS,235503.00,1.6,1.4,3.2,,,,*40
$GPGBS,235458.00,1.4,1.3,3.1,03,,-21.4,3.8*5B
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	`xxGBS`	-	string	$GPGBS	GBS Message ID (xx = current Talker ID)
1	`time`	-	hhmmss.ss	235503.00	UTC time to which this RAIM sentence belongs, see note on UTC representation
2	`errLat`	m	numeric	1.6	Expected error in latitude
3	`errLon`	m	numeric	1.4	Expected error in longitude
4	`errAlt`	m	numeric	3.2	Expected error in altitude
5	`svid`	-	numeric	03	Satellite ID of most likely failed satellite
6	`prob`	-	numeric	-	Probability of missed detection, not supported (empty)
7	`bias`	m	numeric	-21.4	Estimate on most likely failed satellite (a priori residual)
8	`stddev`	m	numeric	3.8	Standard deviation of estimated bias
9	`cs`	-	hexadecimal	*5B	Checksum
10	`<CR><LF>`	-	character	-	Carriage return and line feed

24.3 GGA

24.3.1 Global positioning system fix data

Message	**GGA**		
Description	**Global positioning system fix data**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Output Message		
Comment	**The output of this message is dependent on the currently selected datum (default: WGS84)** Time and position, together with GPS fixing related data (number of satellites in use, and the resulting HDOP, age of differential data if in use, etc.).		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF0 0x00	17	

Message Structure:

```
$xxGGA,time,lat,NS,long,EW,quality,numSV,HDOP,alt,M,sep,M,diffAge,diffStation*cs<CR><LF>
```

Example:

```
$GPGGA,092725.00,4717.11399,N,00833.91590,E,1,08,1.01,499.6,M,48.0,M,,*5B
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	`xxGGA`	-	string	$GPGGA	GGA Message ID (xx = current Talker ID)
1	`time`	-	hhmmss.ss	092725.00	UTC time, see note on UTC representation
2	`lat`	-	ddmm.mmmmm	4717.11399	Latitude (degrees & minutes), see format description
3	`NS`	-	character	N	North/South indicator
4	`long`	-	dddmm.mmmmm	00833.91590	Longitude (degrees & minutes), see format description
5	`EW`	-	character	E	East/West indicator
6	`quality`	-	digit	1	Quality indicator for position fix, see table below and position fix flags description
7	`numSV`	-	numeric	08	Number of satellites used (range: 0-12)
8	`HDOP`	-	numeric	1.01	Horizontal Dilution of Precision
9	`alt`	m	numeric	499.6	Altitude above mean sea level
10	`uAlt`	-	character	M	Altitude units: meters (fixed field)
11	`sep`	m	numeric	48.0	Geoid separation: difference between geoid and mean sea level
12	`uSep`	-	character	M	Separation units: meters (fixed field)
13	`diffAge`	s	numeric	-	Age of differential corrections (blank when DGPS is not used)
14	`diffStation`	-	numeric	-	ID of station providing differential corrections (blank when DGPS is not used)
15	`cs`	-	hexadecimal	*5B	Checksum
16	`<CR><LF>`	-	character	-	Carriage return and line feed

Table Quality Indicator

Quality Indicator	*Description, see also position fix flags description*
0	No Fix / Invalid
1	Standard GPS (2D/3D)
2	Differential GPS
6	Estimated (DR) Fix

24.4 GLL

24.4.1 Latitude and longitude, with time of position fix and status

Message	**GLL**		
Description	**Latitude and longitude, with time of position fix and status**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Output Message		
Comment	**The output of this message is dependent on the currently selected datum (default: WGS84)** -		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF0 0x01	(9) or (10)	

Message Structure:

```
$xxGLL,lat,NS,long,EW,time,status,posMode*cs<CR><LF>
```

Example:

```
$GPGLL,4717.11364,N,00833.91565,E,092321.00,A,A*60
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	`xxGLL`	-	string	$GPGLL	GLL Message ID (xx = current Talker ID)
1	`lat`	-	ddmm.mmmmm	4717.11364	Latitude (degrees & minutes), see format description
2	`NS`	-	character	N	North/South indicator
3	`long`	-	dddmm.mmmmm	00833.91565	Longitude (degrees & minutes), see format description
4	`EW`	-	character	E	East/West indicator
5	`time`	-	hhmmss.ss	092321.00	UTC time, see note on UTC representation
6	`status`	-	character	A	V = Data invalid or receiver warning, A = Data valid. See position fix flags description.
Start of optional block					
7	`posMode`	-	character	A	Positioning mode, see position fix flags description
End of optional block					
7	`cs`	-	hexadecimal	*60	Checksum
8	`<CR><LF>`	-	character	-	Carriage return and line feed

24.5 GLQ

24.5.1 Poll a standard message (if the current Talker ID is GL)

Message	**GLQ**		
Description	**Poll a standard message (if the current Talker ID is GL)**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Input Message		
Comment	Polls a standard NMEA message if the current Talker ID is GL		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF0 0x43	4	

Message Structure:

```
$xxGLQ,msgId*cs<CR><LF>
```

Example:

```
$EIGLQ,RMC*3A
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	xxGLQ	-	string	$EIGLQ	GLQ Message ID (xx = Talker ID of the device requesting the poll)
1	msgId	-	string	RMC	Message ID of the message to be polled
2	cs	-	hexadecimal	*3A	Checksum
3	<CR><LF>	-	character	-	Carriage return and line feed

24.6 GNQ

24.6.1 Poll a standard message (if the current Talker ID is GN)

Message	**GNQ**		
Description	**Poll a standard message (if the current Talker ID is GN)**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Input Message		
Comment	Polls a standard NMEA message if the current Talker ID is GN		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF0 0x42	4	

Message Structure:

```
$xxGNQ,msgId*cs<CR><LF>
```

Example:

```
$EIGNQ,RMC*3A
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	xxGNQ	-	string	$EIGNQ	GNQ Message ID (xx = Talker ID of the device requesting the poll)
1	msgId	-	string	RMC	Message ID of the message to be polled
2	cs	-	hexadecimal	*3A	Checksum
3	<CR><LF>	-	character	-	Carriage return and line feed

24.7 GNS

24.7.1 GNSS fix data

Message	**GNS**		
Description	**GNSS fix data**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Output Message		
Comment	**The output of this message is dependent on the currently selected datum (default: WGS84)** Time and position, together with GNSS fixing related data (number of satellites in use, and the resulting HDOP, age of differential data if in use, etc.).		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF0 0x0D	15	

Message Structure:

```
$xxGNS,time,lat,NS,long,EW,posMode,numSV,HDOP,alt,altRef,diffAge,diffStation*cs<CR><LF>
```

Example:

```
$GPGNS,091547.00,5114.50897,N,00012.28663,W,AA,10,0.83,111.1,45.6,,*71
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	`xxGNS`	-	string	$GPGNS	GNS Message ID (xx = current Talker ID)
1	`time`	-	hhmmss.ss	091547.00	UTC time, see note on UTC representation
2	`lat`	-	ddmm.mmmmm	5114.50897	Latitude (degrees & minutes), see format description
3	`NS`	-	character	N	North/South indicator
4	`long`	-	dddmm.mmmmm	00012.28663	Longitude (degrees & minutes), see format description
5	`EW`	-	character	E	East/West indicator
6	`posMode`	-	character	AA	Positioning mode, see position fix flags description. First character for GPS, second character for GLONASS
7	`numSV`	-	numeric	10	Number of satellites used (range: 0-99)
8	`HDOP`	-	numeric	0.83	Horizontal Dilution of Precision
9	`alt`	m	numeric	111.1	Altitude above mean sea level
10	`sep`	m	numeric	45.6	Geoid separation: difference between geoid and mean sea level
11	`diffAge`	s	numeric	-	Age of differential corrections (blank when DGPS is not used)
12	`diffStation`	-	numeric	-	ID of station providing differential corrections (blank when DGPS is not used)
13	`cs`	-	hexadecimal	*71	Checksum
14	`<CR><LF>`	-	character	-	Carriage return and line feed

24.8 GPQ

24.8.1 Poll a standard message (if the current Talker ID is GP)

Message	**GPQ**		
Description	**Poll a standard message (if the current Talker ID is GP)**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Input Message		
Comment	Polls a standard NMEA message if the current Talker ID is GP		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF0 0x40	4	

Message Structure:

```
$xxGPQ,msgId*cs<CR><LF>
```

Example:

```
$EIGPQ,RMC*3A
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	xxGPQ	-	string	$EIGPQ	GPQ Message ID (xx = Talker ID of the device requesting the poll)
1	msgId	-	string	RMC	Message ID of the message to be polled
2	cs	-	hexadecimal	*3A	Checksum
3	<CR><LF>	-	character	-	Carriage return and line feed

24.9 GRS

24.9.1 GNSS Range Residuals

Message	**GRS**		
Description	**GNSS Range Residuals**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Output Message		
Comment	**This messages relates to associated GGA and GSA messages.** If less than 12 SVs are available, the remaining fields are output empty. If more than 12 SVs are used, only the residuals of the first 12 SVs are output, in order to remain consistent with the NMEA standard.		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF0 0x06	17	

Message Structure:

```
$xxGRS,time, mode {,residual}*cs<CR><LF>
```

Example:

```
$GPGRS,082632.00,1,0.54,0.83,1.00,1.02,-2.12,2.64,-0.71,-1.18,0.25,,,*70
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	xxGRS	-	string	$GPGRS	GRS Message ID (xx = current Talker ID)
1	time	-	hhmmss.ss	082632.00	UTC time of associated position fix, see note on UTC representation

GRS continued

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
2	mode	-	digit	1	Mode (see table below), u-blox receivers will always output Mode 1 residuals
Start of repeated block (12 times)					
3 + 1*N	residual	m	numeric	0.54	Range residuals for SVs used in navigation. The SV order matches the order from the GSA sentence.
End of repeated block					
15	cs	-	hexadecimal	*70	Checksum
16	<CR><LF>	-	character	-	Carriage return and line feed

Table Mode

Mode	*Description*
0	Residuals were used to calculate the position given in the matching GGA sentence.
1	Residuals were recomputed after the GGA position was computed.

24.10 GSA

24.10.1 GNSS DOP and Active Satellites

Message	**GSA**		
Description	**GNSS DOP and Active Satellites**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Output Message		
Comment	The GPS receiver operating mode, satellites used for navigation, and DOP values. • If less than 12 SVs are used for navigation, the remaining fields are left empty. If more than 12 SVs are used for navigation, only the IDs of the first 12 are output. • The SV numbers (fields 'sv') are in the range of 1 to 32 for GPS satellites, and 33 to 64 for SBAS satellites (33 = SBAS PRN 120, 34 = SBAS PRN 121, and so on)		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF0 0x02	20	

Message Structure:

```
$xxGSA,opMode,navMode{,sv},PDOP,HDOP,VDOP*cs<CR><LF>
```

Example:

```
$GPGSA,A,3,23,29,07,08,09,18,26,28,,,,,1.94,1.18,1.54*0D
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	xxGSA	-	string	$GPGSA	GSA Message ID (xx = current Talker ID)
1	opMode	-	character	A	Operation mode, see first table below
2	navMode	-	digit	3	Navigation mode, see second table below and position fix flags description
Start of repeated block (12 times)					
3 + 1*N	sv	-	numeric	29	Satellite number
End of repeated block					
15	PDOP	-	numeric	1.94	Position dilution of precision

GSA continued

Field No.	Name	Unit	Format	Example	Description
16	HDOP	-	numeric	1.18	Horizontal dilution of precision
17	VDOP	-	numeric	1.54	Vertical dilution of precision
18	cs	-	hexadecimal	*0D	Checksum
19	<CR><LF>	-	character	-	Carriage return and line feed

Table Operation Mode

Operation Mode	Description
M	Manually set to operate in 2D or 3D mode
A	Automatically switching between 2D or 3D mode

Table Navigation Mode

Navigation Mode	Description, see also position fix flags description
1	Fix not available
2	2D Fix
3	3D Fix

24.11 GST

24.11.1 GNSS Pseudo Range Error Statistics

Message	**GST**		
Description	**GNSS Pseudo Range Error Statistics**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Output Message		
Comment	This message reports statisical information on the quality of the position solution.		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF0 0x07	11	

Message Structure:

```
$xxGST,time,rangeRms,stdMajor,stdMinor,orient,stdLat,stdLong,stdAlt*cs<CR><LF>
```

Example:

```
$GPGST,082356.00,1.8,,,,1.7,1.3,2.2*7E
```

Field No.	Name	Unit	Format	Example	Description
0	xxGST	-	string	$GPGST	GST Message ID (xx = current Talker ID)
1	time	-	hhmmss.ss	082356.00	UTC time of associated position fix, see note on UTC representation
2	rangeRms	m	numeric	1.8	RMS value of the standard deviation of the ranges
3	stdMajor	m	numeric	-	Standard deviation of semi-major axis (blank - not supported)
4	stdMinor	m	numeric	-	Standard deviation of semi-minor axis (blank - not supported)
5	orient	deg	numeric	-	Orientation of semi-major axis (blank - not supported)
6	stdLat	m	numeric	1.7	Standard deviation of latitude error
7	stdLong	m	numeric	1.3	Standard deviation of longitude error

GST continued

Field No.	Name	Unit	Format	Example	Description
8	stdAlt	m	numeric	2.2	Standard deviation of altitude error
9	cs	-	hexadecimal	*7E	Checksum
10	<CR><LF>	-	character	-	Carriage return and line feed

24.12 GSV

24.12.1 GNSS Satellites in View

Message	**GSV**		
Description	**GNSS Satellites in View**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Output Message		
Comment	The number of satellites in view, together with each SV ID, elevation azimuth, and signal strength (C/No) value. Only four satellite details are transmitted in one message.		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF0 0x03	7..16	

Message Structure:

```
$xxGSV,numMsg,msgNum,numSV,{,sv,elv,az,cno}*cs<CR><LF>
```

Example:

```
$GPGSV,3,1,10,23,38,230,44,29,71,156,47,07,29,116,41,08,09,081,36*7F

$GPGSV,3,2,10,10,07,189,,05,05,220,,09,34,274,42,18,25,309,44*72

$GPGSV,3,3,10,26,82,187,47,28,43,056,46*77
```

Field No.	Name	Unit	Format	Example	Description
0	xxGSV	-	string	$GPGSV	GSV Message ID (xx = GSV Talker ID)
1	numMsg	-	digit	3	Number of messages, total number of GSV messages being output
2	msgNum	-	digit	1	Number of this message
3	numSV	-	numeric	10	Number of satellites in view
Start of repeated block (1..4 times)					
4 + 4*N	sv	-	numeric	23	Satellite ID
5 + 4*N	elv	deg	numeric	38	Elevation (range 0-90)
6 + 4*N	az	deg	numeric	230	Azimuth, (range 0-359)
7 + 4*N	cno	dBHz	numeric	44	Signal strength (C/N0, range 0-99), blank when not tracking
End of repeated block					
5..16	cs	-	hexadecimal	*7F	Checksum
6..16	<CR><LF>	-	character	-	Carriage return and line feed

24.13 RMC

24.13.1 Recommended Minimum data

Message	**RMC**		
Description	**Recommended Minimum data**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Output Message		
Comment	**The output of this message is dependent on the currently selected datum (default: WGS84)** The recommended minimum sentence defined by NMEA for GNSS system data.		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF0 0x04	15	

Message Structure:

```
$xxRMC,time,status,lat,NS,long,EW,spd,cog,date,mv,mvEW,posMode*cs<CR><LF>
```

Example:

```
$GPRMC,083559.00,A,4717.11437,N,00833.91522,E,0.004,77.52,091202,,,A*57
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	`xxRMC`	-	string	$GPRMC	RMC Message ID (xx = current Talker ID)
1	`time`	-	hhmmss.ss	083559.00	UTC time, see note on UTC representation
2	`status`	-	character	A	Status, V = Navigation receiver warning, A = Data valid, see position fix flags description
3	`lat`	-	ddmm.mmmmm	4717.11437	Latitude (degrees & minutes), see format description
4	`NS`	-	character	N	North/South indicator
5	`long`	-	dddmm.mmmmm	00833.91522	Longitude (degrees & minutes), see format description
6	`EW`	-	character	E	East/West indicator
7	`spd`	knots	numeric	0.004	Speed over ground
8	`cog`	degrees	numeric	77.52	Course over ground
9	`date`	-	ddmmyy	091202	Date in day, month, year format, see note on UTC representation
10	`mv`	degrees	numeric	-	Magnetic variation value (blank - not supported)
11	`mvEW`	-	character	-	Magnetic variation E/W indicator (blank - not supported)
12	`posMode`	-	character	-	Mode Indicator, see position fix flags description
13	`cs`	-	hexadecimal	*57	Checksum
14	`<CR><LF>`	-	character	-	Carriage return and line feed

24.14 TXT

24.14.1 Text Transmission

Message	**TXT**		
Description	**Text Transmission**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Output Message		
Comment	**This message is not configured through UBX-CFG-MSG, but instead through UBX-CFG-INF.** This message outputs various information on the receiver, such as power-up screen, software version etc. This message can be configured using UBX Protocol message UBX-CFG-INF.		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF0 0x41	7	

Message Structure:

```
$xxTXT,numMsg,msgNum,msgType,text*cs<CR><LF>
```

Example:

```
$GPTXT,01,01,02,u-blox ag - www.u-blox.com*50
$GPTXT,01,01,02,ANTARIS ATR0620 HW 00000040*67
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	xxTXT	-	string	$GPTXT	TXT Message ID (xx = current Talker ID)
1	numMsg	-	numeric	01	Total number of messages in this transmission, 01..99
2	msgNum	-	numeric	01	Message number in this transmission, range 01..xx
3	msgType	-	numeric	02	Text identifier, u-blox GPS receivers specify the type of the message with this number. 00: Error 01: Warning 02: Notice 07: User
4	text	-	string	www.u-blox.com	Any ASCII text
5	cs	-	hexadecimal	*67	Checksum
6	<CR><LF>	-	character	-	Carriage return and line feed

24.15 VTG

24.15.1 Course over ground and Ground speed

Message	**VTG**		
Description	**Course over ground and Ground speed**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Output Message		
Comment	Velocity is given as Course over Ground (COG) and Speed over Ground (SOG).		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF0 0x05	12	

Message Structure:

```
$xxVTG,cogt,T,cogm,M,knots,N,kph,K,posMode*cs<CR><LF>
```

Example:

```
$GPVTG,77.52,T,,M,0.004,N,0.008,K,A*06
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	`xxVTG`	-	string	$GPVTG	VTG Message ID (xx = current Talker ID)
1	`cogt`	degrees	numeric	77.52	Course over ground (true)
2	`T`	-	character	T	Fixed field: true
3	`cogm`	degrees	numeric	-	Course over ground (magnetic), not output
4	`M`	-	character	M	Fixed field: magnetic
5	`knots`	knots	numeric	0.004	Speed over ground
6	`N`	-	character	N	Fixed field: knots
7	`kph`	km/h	numeric	0.008	Speed over ground
8	`K`	-	character	K	Fixed field: kilometers per hour
9	`posMode`	-	character	A	Mode Indicator, see position fix flags description
10	`cs`	-	hexadecimal	*06	Checksum
11	`<CR><LF>`	-	character	-	Carriage return and line feed

24.16 ZDA

24.16.1 Time and Date

Message	**ZDA**		
Description	**Time and Date**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Output Message		
Comment	-		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF0 0x08	9	

Message Structure:

```
$xxZDA,hhmmss.ss,day,month,year,ltzh,ltzn*cs<CR><LF>
```

Example:

```
$GPZDA,082710.00,16,09,2002,00,00*64
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	`xxZDA`	-	string	$GPZDA	ZDA Message ID (xx = current Talker ID)
1	`time`	-	hhmmss.ss	082710.00	UTC Time, see note on UTC representation
2	`day`	day	dd	16	UTC day (range: 1-31)
3	`month`	month	mm	09	UTC month (range: 1-12)
4	`year`	year	yyyy	2002	UTC year
5	`ltzh`	-	-xx	00	Local time zone hours (fixed to 00)
6	`ltzn`	-	zz	00	Local time zone minutes (fixed to 00)
7	`cs`	-	hexadecimal	*64	Checksum
8	`<CR><LF>`	-	character	-	Carriage return and line feed

25 PUBX Messages

Proprietary Messages: i.e. Messages defined by u-blox.

25.1 CONFIG (PUBX,41)

25.1.1 Set Protocols and Baudrate

Message	**CONFIG**		
Description	**Set Protocols and Baudrate**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Set Message		
Comment	-		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF1 0x41	9	

Message Structure:

```
$PUBX,41,portId,inProto,outProto,baudrate,autobauding*cs<CR><LF>
```

Example:

```
$PUBX,41,1,0007,0003,19200,0*25
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	`$PUBX`	-	string	$PUBX	Message ID, UBX protocol header, proprietary sentence
1	`msgId`	-	numeric	41	Proprietary message identifier
2	`portId`	-	numeric	1	ID of communication port. For a list of port IDs see Serial Communication Ports Description.
3	`inProto`	-	hexadecimal	0007	Input protocol mask. Bitmask, specifying which protocols(s) are allowed for input. For details see corresponding field in `UBX-CFG-PRT`.
4	`outProto`	-	hexadecimal	0003	Output protocol mask. Bitmask, specifying which protocols(s) are allowed for input. For details see corresponding field in `UBX-CFG-PRT`.
5	`baudrate`	bits/s	numeric	19200	Baudrate
6	`autobauding`	-	numeric	0	Autobauding: 1=enable, 0=disable (not supported on u-blox 5, set to 0)
7	`cs`	-	hexadecimal	*25	Checksum
8	`<CR><LF>`	-	character	-	Carriage return and line feed

25.2 POSITION (PUBX,00)

25.2.1 Poll a PUBX,00 message

Message	**POSITION**		
Description	**Poll a PUBX,00 message**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Input Message		
Comment	A PUBX,00 message is polled by sending the PUBX,00 message without any data fields.		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF1 0x00	4	

Message Structure:

```
$PUBX,00*33<CR><LF>
```

Example:

```
$PUBX,00*33
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	`$PUBX`	-	string	$PUBX	Message ID, UBX protocol header, proprietary sentence
1	`msgId`	-	numeric	00	Set to 00 to poll a PUBX,00 message
2	`cs`	-	hexadecimal	*33	Checksum
3	`<CR><LF>`	-	character	-	Carriage return and line feed

25.2.2 Lat/Long Position Data

Message	**POSITION**		
Description	**Lat/Long Position Data**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Output Message		
Comment	**The output of this message is dependent on the currently selected datum (default: WGS84)** This message contains position solution data. The datum selection may be changed using the message `UBX-CFG-DAT`.		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF1 0x00	23	

Message Structure:

```
$PUBX,00,time,lat,NS,long,EW,altRef,navStat,hAcc,vAcc,SOG,COG,vVel,diffAge,HDOP,VDOP,TDOP,numSvs,reserved,DR,*cs<CR><LF>
```

Example:

```
$PUBX,00,081350.00,4717.113210,N,00833.915187,E,546.589,G3,2.1,2.0,0.007,77.52,0.007,,0.92,1.19,0.77,9,0,0*5F
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	`$PUBX`	-	string	$PUBX	Message ID, UBX protocol header, proprietary sentence

POSITION continued

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
1	`msgId`	-	numeric	00	Proprietary message identifier: 00
2	`time`	-	hhmmss.ss	081350.00	UTC time, see note on UTC representation
3	`lat`	-	ddmm.mmmmm	4717.113210	Latitude (degrees & minutes), see format description
4	`NS`	-	character	N	North/South Indicator
5	`long`	-	dddmm.mmmmm	00833.915187	Longitude (degrees & minutes), see format description
6	`EW`	-	character	E	East/West indicator
7	`altRef`	m	numeric	546.589	Altitude above user datum ellipsoid.
8	`navStat`	-	string	G3	Navigation Status, See Table below
9	`hAcc`	m	numeric	2.1	Horizontal accuracy estimate.
10	`vAcc`	m	numeric	2.0	Vertical accuracy estimate.
11	`SOG`	km/h	numeric	0.007	Speed over ground
12	`COG`	deg	numeric	77.52	Course over ground
13	`vVel`	m/s	numeric	0.007	Vertical velocity (positive downwards)
14	`diffAge`	s	numeric	-	Age of differential corrections (blank when DGPS is not used)
15	`HDOP`	-	numeric	0.92	HDOP, Horizontal Dilution of Precision
16	`VDOP`	-	numeric	1.19	VDOP, Vertical Dilution of Precision
17	`TDOP`	-	numeric	0.77	TDOP, Time Dilution of Precision
18	`numSvs`	-	numeric	9	Number of satellites used in the navigation solution
19	`reserved`	-	numeric	0	Reserved, always set to 0
20	`DR`	-	numeric	0	DR used
21	`cs`	-	hexadecimal	*5B	Checksum
22	`<CR><LF>`	-	character	-	Carriage return and line feed

Table Navigation Status

Navigation Status	*Description*
NF	No Fix
DR	Dead reckoning only solution
G2	Stand alone 2D solution
G3	Stand alone 3D solution
D2	Differential 2D solution
D3	Differential 3D solution
RK	Combined GPS + dead reckoning solution
TT	Time only solution

25.3 RATE (PUBX,40)

25.3.1 Set NMEA message output rate

Message	**RATE**		
Description	**Set NMEA message output rate**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Set Message		
Comment	Set/Get message rate configuration (s) to/from the receiver. • Send rate is relative to the event a message is registered on. For example, if the rate of a navigation message is set to 2, the message is sent every second navigation solution.		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF1 0x40	11	

Message Structure:

```
$PUBX,40,msgId,rddc,rus1,rus2,rusb,rspi,reserved*cs<CR><LF>
```

Example:

```
$PUBX,40,GLL,1,0,0,0,0,0*5D
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	`$PUBX`	-	string	$PUBX	Message ID, UBX protocol header, proprietary sentence
1	`ID`	-	numeric	40	Proprietary message identifier
2	`msgId`	-	string	GLL	NMEA message identifier
3	`rddc`	cycles	numeric	1	output rate on DDC 0 disables that message from being output on this port 1 means that this message is output every epoch
4	`rus1`	cycles	numeric	1	output rate on USART 1 0 disables that message from being output on this port 1 means that this message is output every epoch
5	`rus2`	cycles	numeric	1	output rate on USART 2 0 disables that message from being output on this port 1 means that this message is output every epoch
6	`rusb`	cycles	numeric	1	output rate on USB 0 disables that message from being output on this port 1 means that this message is output every epoch
7	`rspi`	cycles	numeric	1	output rate on SPI 0 disables that message from being output on this port 1 means that this message is output every epoch
8	`reserved`	-	numeric	0	Reserved: always fill with 0
9	`cs`	-	hexadecimal	*5D	Checksum
10	`<CR><LF>`	-	character	-	Carriage return and line feed

25.4 SVSTATUS (PUBX,03)

25.4.1 Poll a PUBX,03 message

Message	**SVSTATUS**		
Description	**Poll a PUBX,03 message**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Input Message		
Comment	A PUBX,03 message is polled by sending the PUBX,03 message without any data fields.		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF1 0x03	4	

Message Structure:

```
$PUBX,03*30<CR><LF>
```

Example:

```
$PUBX,03*30
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	`$PUBX`	-	string	$PUBX	Message ID, UBX protocol header, proprietary sentence
1	`msgId`	-	numeric	03	Set to 03 to poll a PUBX,03 message
2	`cs`	-	hexadecimal	*30	Checksum
3	`<CR><LF>`	-	character	-	Carriage return and line feed

25.4.2 Satellite Status

Message	**SVSTATUS**		
Description	**Satellite Status**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Output Message		
Comment	The PUBX,03 message contains satellite status information.		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF1 0x03	5 + 6*n	

Message Structure:

```
$PUBX,03,GT{,sv,s,az,el,cno,lck},*cs<CR><LF>
```

Example:

```
$PUBX,03,11,23,-,,,45,010,29,-,,,46,013,07,-,,,42,015,08,U,067,31,42,025,10,U,195,33,46,026,18,U,32
6,08,39,026,17,-,,,32,015,26,U,306,66,48,025,27,U,073,10,36,026,28,U,089,61,46,024,15,-,,,39,014*0D
```

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
0	`$PUBX`	-	string	$PUBX	Message ID, UBX protocol header, proprietary sentence
1	`msgId`	-	numeric	03	Proprietary message identifier: 03
2	`n`	-	numeric	11	Number of GPS satellites tracked
Start of repeated block (n times)					

SVSTATUS continued

Field No.	Name	Unit	Format	Example	Description
3 + 6*N	sv	-	numeric	23	Satellite ID
4 + 6*N	s	-	character	-	Satellite status, see table below
5 + 6*N	az	deg	numeric	-	Satellite azimuth (range: 0-359)
6 + 6*N	el	deg	numeric	-	Satellite elevation (range: 0-90)
7 + 6*N	cno	dBHz	numeric	45	Signal strength (C/N0, range 0-99), blank when not tracking
8 + 6*N	lck	s	numeric	010	Satellite carrier lock time (range: 0-64) 0: code lock only 64: lock for 64 seconds or more
End of repeated block					
3 + 6*n	cs	-	hexadecimal	*0D	Checksum
4 + 6*n	<CR><LF>	-	character	-	Carriage return and line feed

Table Satellite Status

Satellite Status	Description
-	Not used
U	Used in solution
e	Ephemeris available, but not used for navigation

25.5 TIME (PUBX,04)

25.5.1 Poll a PUBX,04 message

Message	**TIME**		
Description	**Poll a PUBX,04 message**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Input Message		
Comment	A PUBX,04 message is polled by sending the PUBX,04 message without any data fields.		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF1 0x04	4	

Message Structure:

```
$PUBX,04*37<CR><LF>
```

Example:

```
$PUBX,04*37
```

Field No.	Name	Unit	Format	Example	Description
0	$PUBX	-	string	$PUBX	Message ID, UBX protocol header, proprietary sentence
1	msgId	-	numeric	04	Set to 04 to poll a PUBX,04 message

TIME continued

Field No.	Name	Unit	Format	Example	Description
2	cs	-	hexadecimal	*37	Checksum
3	<CR><LF>	-	character	-	Carriage Return and Line Feed

25.5.2 Time of Day and Clock Information

Message	**TIME**		
Description	**Time of Day and Clock Information**		
Firmware	Supported on: • u-blox 7 firmware version 1.00		
Type	Output Message		
Comment	-		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF1 0x04	12	

Message Structure:

```
$PUBX,04,time,date,utcTow,utcWk,leapSec,clkBias,clkDrift,tpGran,*cs<CR><LF>
```

Example:

```
$PUBX,04,073731.00,091202,113851.00,1196,15D,1930035,-2660.664,43,*3C
```

Field No.	Name	Unit	Format	Example	Description
0	$PUBX	-	string	$PUBX	Message ID, UBX protocol header, proprietary sentence
1	msgId	-	numeric	04	Proprietary message identifier: 04
2	time	-	hhmmss.ss	073731.00	UTC time, see note on UTC representation
3	date	-	ddmmyy	091202	UTC date, day, month, year format, see note on UTC representation
4	utcTow	s	numeric	113851.00	UTC Time of Week
5	utcWk	-	numeric	1196	UTC week number, continues beyond 1023
6	leapSec	s	numeric/text	15D	Leap seconds The number is marked with a 'D' if the value is the firmware default value. If the value is not marked it has been received from a satellite.
7	clkBias	ns	numeric	1930035	Receiver clock bias
8	clkDrift	ns/s	numeric	-2660.664	Receiver clock drift
9	tpGran	ns	numeric	43	Time Pulse Granularity, The quantization error of the TIMEPULSE pin
10	cs	-	hexadecimal	*3C	Checksum
11	<CR><LF>	-	character	-	Carriage Return and Line Feed

UBX Protocol

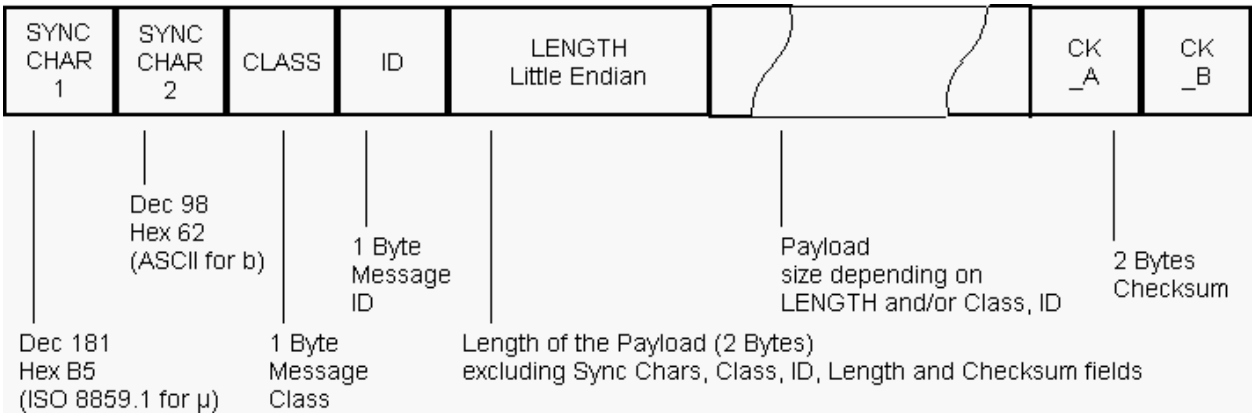
26 UBX Protocol Key Features

u-blox GNSS receivers use a u-blox proprietary protocol to transmit GNSS data to a host computer. This protocol has the following key features:

- Compact - uses 8 Bit Binary Data.
- Checksum Protected - uses a low-overhead checksum algorithm
- Modular - uses a 2-stage message identifier (Class- and Message ID)

27 UBX Packet Structure

A basic UBX Packet looks as follows:



- Every Message starts with 2 Bytes: 0xB5 0x62
- A 1 Byte Class Field follows. The Class defines the basic subset of the message
- A 1 Byte ID Field defines the message that is to follow
- A 2 Byte Length Field is following. Length is defined as being the length of the payload, only. It does not include Sync Chars, Length Field, Class, ID or CRC fields. The number format of the length field is an unsigned 16-Bit integer in Little Endian Format.
- The Payload is a variable length field.
- CK_A and CK_B is a 16 Bit checksum whose calculation is defined below.

28 UBX Payload Definition Rules

28.1 Structure Packing

Values are placed in an order that structure packing is not a problem. This means that 2 byte values shall start on offsets which are a multiple of 2, 4 byte values shall start at a multiple of 4, and so on.

28.2 Message Naming

Referring to messages is done by adding the class name and a dash in front of the message name. For example, the ECEF-Message is referred to as NAV-POSECEF. Referring to values is done by adding a dash and the name, e.g. NAV-POSECEF-X

28.3 Number Formats

All multi-byte values are ordered in Little Endian format, unless otherwise indicated.

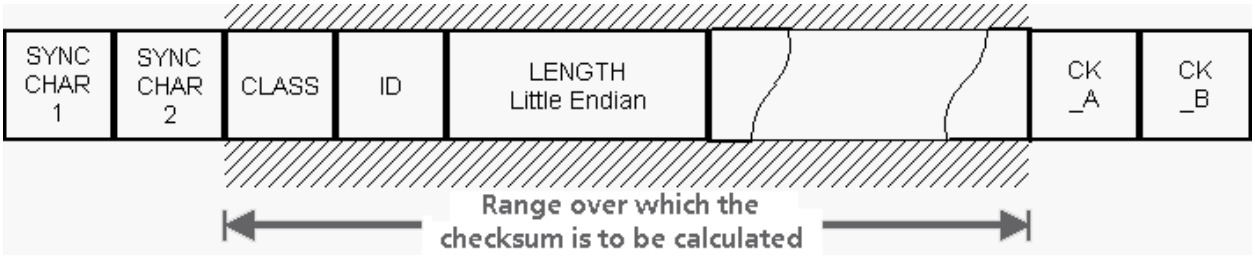
All floating point values are transmitted in IEEE754 single or double precision. A technical description of the IEEE754 format can be found in the AnswerBook from the ADS1.x toolkit.

Variable Type Definitions

Short	*Type*	*Size (Bytes)*	*Comment*	*Min/Max*	*Resolution*
U1	Unsigned Char	1		0..255	1
I1	Signed Char	1	2's complement	-128..127	1
X1	Bitfield	1		n/a	n/a
U2	Unsigned Short	2		0..65535	1
I2	Signed Short	2	2's complement	-32768..32767	1
X2	Bitfield	2		n/a	n/a
U4	Unsigned Long	4		0..4 '294'967'295	1
I4	Signed Long	4	2's complement	-2'147'483'648 .. 2'147'483'647	1
X4	Bitfield	4		n/a	n/a
R4	IEEE 754 Single Precision	4		-1*2^+127 .. 2^+127	~ Value * 2^-24
R8	IEEE 754 Double Precision	8		-1*2^+1023 .. 2^+1023	~ Value * 2^-53
CH	ASCII / ISO 8859.1 Encoding	1			

29 UBX Checksum

The checksum is calculated over the packet, starting and including the CLASS field, up until, but excluding, the Checksum Field:



The checksum algorithm used is the 8-Bit Fletcher Algorithm, which is used in the TCP standard (RFC 1145). This algorithm works as follows:

Buffer[N] contains the data over which the checksum is to be calculated.

The two CK_ values are 8-Bit unsigned integers, only! If implementing with larger-sized integer values, make sure to mask both CK_A and CK_B with 0xFF after both operations in the loop.

```
CK_A = 0, CK_B = 0
For(I=0;I<N;I++)
{
    CK_A = CK_A + Buffer[I]
    CK_B = CK_B + CK_A
}
```

After the loop, the two U1 values contain the checksum, transmitted at the end of the packet.

30 UBX Message Flow

There are certain features associated with the messages being sent back and forth:

30.1 Acknowledgement

When messages from the class CFG are sent to the receiver, the receiver will send an "acknowledge" (ACK-ACK) or a "not acknowledge" (ACK-NAK) message back to the sender, depending on whether or not the message was processed correctly.

Some messages from other classes (e.g. LOG) also use the same acknowledgement mechanism.

30.2 Polling Mechanism

All messages that are output by the receiver in a periodic manner (i.e. messages in classes MON, NAV and RXM) can also be polled.

There is not a single specific message which polls any other message. The UBX protocol was designed such, that when sending a message with no payload (or just a single parameter which identifies the poll request) the message is polled.

31 UBX Class IDs

A class is a grouping of messages which are related to each other. The following table lists all the current message classes.

Name	*Class*	*Description*
NAV	0x01	Navigation Results: Position, Speed, Time, Acc, Heading, DOP, SVs used
RXM	0x02	Receiver Manager Messages: Satellite Status, RTC Status
INF	0x04	Information Messages: Printf-Style Messages, with IDs such as Error, Warning, Notice
ACK	0x05	Ack/Nack Messages: as replies to CFG Input Messages
CFG	0x06	Configuration Input Messages: Set Dynamic Model, Set DOP Mask, Set Baud Rate, etc.
MON	0x0A	Monitoring Messages: Comunication Status, CPU Load, Stack Usage, Task Status
AID	0x0B	AssistNow Aiding Messages: Ephemeris, Almanac, other A-GPS data input
TIM	0x0D	Timing Messages: Time Pulse Output, Timemark Results
LOG	0x21	Logging Messages: Log creation, deletion, info and retrieval

All remaining class IDs are reserved.

32 UBX Messages Overview

Page	Mnemonic	Cls/ID	Length	Type	Description
UBX Class ACK				**Ack/Nack Messages**	
80	**ACK-ACK**	0x05 0x01	2	Output	Message Acknowledged
80	**ACK-NAK**	0x05 0x00	2	Output	Message Not-Acknowledged
UBX Class AID				**AssistNow Aiding Messages**	
81	**AID-ALM**	0x0B 0x30	0	Poll Request	Poll GPS Aiding Almanac Data
81	**AID-ALM**	0x0B 0x30	1	Poll Request	Poll GPS Aiding Almanac Data for a SV
82	**AID-ALM**	0x0B 0x30	(8) or (40)	Input/Output	GPS Aiding Almanac Input/Output Message
82	**AID-ALPSRV**	0x0B 0x32	16	Output	ALP client requests AlmanacPlus data from server
83	**AID-ALPSRV**	0x0B 0x32	16 + 1*dataSize	Input	ALP server sends AlmanacPlus data to client
84	**AID-ALPSRV**	0x0B 0x32	8 + 2*size	Output	ALP client sends AlmanacPlus data to server.
84	**AID-ALP**	0x0B 0x50	0 + 2*N	Input	ALP file data transfer to the receiver
85	**AID-ALP**	0x0B 0x50	1	Input	Mark end of data transfer
85	**AID-ALP**	0x0B 0x50	1	Output	Acknowledges a data transfer
86	**AID-ALP**	0x0B 0x50	1	Output	Indicate problems with a data transfer
86	**AID-ALP**	0x0B 0x50	24	Periodic/Polled	Poll the AlmanacPlus status
87	**AID-AOP**	0x0B 0x33	0	Poll request	Poll AssistNow Autonomous data
87	**AID-AOP**	0x0B 0x33	1	Poll request	Poll AssistNow Autonomous data for one satellite
88	**AID-AOP**	0x0B 0x33	(60) or (204)	Input/Output	AssistNow Autonomous data
88	**AID-DATA**	0x0B 0x10	0	Poll Request	Polls all GPS Initial Aiding Data
89	**AID-EPH**	0x0B 0x31	0	Poll Request	Poll GPS Aiding Ephemeris Data
89	**AID-EPH**	0x0B 0x31	1	Poll Request	Poll GPS Aiding Ephemeris Data for a SV
89	**AID-EPH**	0x0B 0x31	(8) or (104)	Input/Output	GPS Aiding Ephemeris Input/Output Message
90	**AID-HUI**	0x0B 0x02	0	Poll Request	Poll GPS Health, UTC and ionosphere parameters
91	**AID-HUI**	0x0B 0x02	72	Input/Output	GPS Health, UTC and ionosphere parameters
92	**AID-INI**	0x0B 0x01	0	Poll Request	Poll GPS Initial Aiding Data
92	**AID-INI**	0x0B 0x01	48	Input/Output	Aiding position, time, frequency, clock drift
94	**AID-REQ**	0x0B 0x00	0	Virtual	Sends a poll (AID-DATA) for all GPS Aiding Data
UBX Class CFG				**Configuration Input Messages**	
95	**CFG-ANT**	0x06 0x13	0	Poll Request	Poll Antenna Control Settings
95	**CFG-ANT**	0x06 0x13	4	Input/Output	Antenna Control Settings
96	**CFG-CFG**	0x06 0x09	(12) or (13)	Command	Clear, Save and Load configurations
98	**CFG-DAT**	0x06 0x06	0	Poll Request	Poll Datum Setting
98	**CFG-DAT**	0x06 0x06	44	Input	Set User-defined Datum
99	**CFG-DAT**	0x06 0x06	52	Output	The currently defined Datum
100	**CFG-GNSS**	0x06 0x3E	0	Poll Request	Polls the configuration of the GNSS system configuration
100	**CFG-GNSS**	0x06 0x3E	4 + 8*numConfigBlocks	Input/Output	GNSS system configuration
101	**CFG-INF**	0x06 0x02	1	Poll Request	Poll INF message configuration for one protocol

UBX Messages Overview continued

Page	*Mnemonic*	*Cls/ID*	*Length*	*Type*	*Description*
102	**CFG-INF**	0x06 0x02	0 + 10*N	Input/Output	Information message configuration
103	**CFG-ITFM**	0x06 0x39	0	Poll Request	Polls the Jamming/Interference Monitor configuration.
103	**CFG-ITFM**	0x06 0x39	8	Command	Jamming/Interference Monitor configuration.
104	**CFG-LOGFILTER**	0x06 0x47	0	Poll Request	Poll Data Logger filter Configuration
104	**CFG-LOGFILTER**	0x06 0x47	12	Input/Output	Data Logger Configuration
106	**CFG-MSG**	0x06 0x01	2	Poll Request	Poll a message configuration
106	**CFG-MSG**	0x06 0x01	8	Input/Output	Set Message Rate(s)
107	**CFG-MSG**	0x06 0x01	3	Input/Output	Set Message Rate
107	**CFG-NAV5**	0x06 0x24	0	Poll Request	Poll Navigation Engine Settings
107	**CFG-NAV5**	0x06 0x24	36	Input/Output	Navigation Engine Settings
109	**CFG-NAVX5**	0x06 0x23	0	Poll Request	Poll Navigation Engine Expert Settings
109	**CFG-NAVX5**	0x06 0x23	40	Input/Output	Navigation Engine Expert Settings
111	**CFG-NMEA**	0x06 0x17	0	Poll Request	Poll the NMEA protocol configuration
111	**CFG-NMEA**	0x06 0x17	4	Input/Output	NMEA protocol configuration (deprecated)
113	**CFG-NMEA**	0x06 0x17	12	Input/Output	NMEA protocol configuration
115	**CFG-NVS**	0x06 0x22	13	Command	Clear, Save and Load non-volatile storage data
116	**CFG-PM2**	0x06 0x3B	0	Poll Request	Poll extended Power Management configuration
116	**CFG-PM2**	0x06 0x3B	44	Input/Output	Extended Power Management configuration
118	**CFG-PRT**	0x06 0x00	0	Poll Request	Polls the configuration of the used I/O Port
118	**CFG-PRT**	0x06 0x00	1	Poll Request	Polls the configuration for one I/O Port
119	**CFG-PRT**	0x06 0x00	20	Input/Output	Port Configuration for UART
122	**CFG-PRT**	0x06 0x00	20	Input/Output	Port Configuration for USB Port
123	**CFG-PRT**	0x06 0x00	20	Input/Output	Port Configuration for SPI Port
126	**CFG-PRT**	0x06 0x00	20	Input/Output	Port Configuration for DDC Port
128	**CFG-RATE**	0x06 0x08	0	Poll Request	Poll Navigation/Measurement Rate Settings
129	**CFG-RATE**	0x06 0x08	6	Input/Output	Navigation/Measurement Rate Settings
129	**CFG-RINV**	0x06 0x34	0	Poll Request	Poll contents of Remote Inventory
130	**CFG-RINV**	0x06 0x34	1 + 1*N	Input/Output	Contents of Remote Inventory
130	**CFG-RST**	0x06 0x04	4	Command	Reset Receiver / Clear Backup Data Structures
132	**CFG-RXM**	0x06 0x11	0	Poll Request	Poll RXM configuration
132	**CFG-RXM**	0x06 0x11	2	Input/Output	RXM configuration
133	**CFG-SBAS**	0x06 0x16	0	Poll Request	Poll contents of SBAS Configuration
133	**CFG-SBAS**	0x06 0x16	8	Input/Output	SBAS Configuration
135	**CFG-TP5**	0x06 0x31	0	Poll Request	Poll Time Pulse Parameters
135	**CFG-TP5**	0x06 0x31	1	Poll Request	Poll Time Pulse Parameters
135	**CFG-TP5**	0x06 0x31	32	Input/Output	Time Pulse Parameters
137	**CFG-USB**	0x06 0x1B	0	Poll Request	Poll a USB configuration
137	**CFG-USB**	0x06 0x1B	108	Input/Output	USB Configuration

UBX Messages Overview continued

Page	Mnemonic	Cls/ID	Length	Type	Description
UBX Class INF				**Information Messages**	
139	**INF-DEBUG**	0x04 0x04	0 + 1*N	Output	ASCII String output, indicating debug output
139	**INF-ERROR**	0x04 0x00	0 + 1*N	Output	ASCII String output, indicating an error
140	**INF-NOTICE**	0x04 0x02	0 + 1*N	Output	ASCII String output, with informational contents
140	**INF-TEST**	0x04 0x03	0 + 1*N	Output	ASCII String output, indicating test output
141	**INF-WARNING**	0x04 0x01	0 + 1*N	Output	ASCII String output, indicating a warning
UBX Class LOG				**Logging Messages**	
142	**LOG-CREATE**	0x21 0x07	8	Command	Create Log File
143	**LOG-ERASE**	0x21 0x03	0	Command	Erase Logged Data
143	**LOG-FINDTIME**	0x21 0x0E	12	Input	Finds the index of the first log entry <= given time
144	**LOG-FINDTIME**	0x21 0x0E	8	Output	This message is the response to FINDTIME request.
144	**LOG-INFO**	0x21 0x08	0	Poll Request	Poll for log information
144	**LOG-INFO**	0x21 0x08	48	Output	Log information
146	**LOG-RETRIEVEPOS**	0x21 0x0b	40	Output	Position fix log entry
147	**LOG-RETRIEVESTRING**	0x21 0x0d	16 + 1*byteCount	Output	Byte string log entry
147	**LOG-RETRIEVE**	0x21 0x09	12	Command	Request log data
148	**LOG-STRING**	0x21 0x04	0 + 1*N	Command	Store arbitrary string in on-board Flash memory
UBX Class MON				**Monitoring Messages**	
149	**MON-HW2**	0x0A 0x0B	28	Periodic/Polled	Extended Hardware Status
150	**MON-HW**	0x0A 0x09	60	Periodic/Polled	Hardware Status
151	**MON-IO**	0x0A 0x02	0 + 20*N	Periodic/Polled	I/O Subsystem Status
152	**MON-MSGPP**	0x0A 0x06	120	Periodic/Polled	Message Parse and Process Status
152	**MON-RXBUF**	0x0A 0x07	24	Periodic/Polled	Receiver Buffer Status
153	**MON-RXR**	0x0A 0x21	1	Output	Receiver Status Information
153	**MON-TXBUF**	0x0A 0x08	28	Periodic/Polled	Transmitter Buffer Status
154	**MON-VER**	0x0A 0x04	0	Poll Request	Poll Receiver/Software Version
155	**MON-VER**	0x0A 0x04	40 + 30*N	Answer to Poll	Receiver/Software Version
UBX Class NAV				**Navigation Results**	
156	**NAV-AOPSTATUS**	0x01 0x60	20	Periodic/Polled	AssistNow Autonomous Status
157	**NAV-CLOCK**	0x01 0x22	20	Periodic/Polled	Clock Solution
157	**NAV-DGPS**	0x01 0x31	16 + 12*numCh	Periodic/Polled	DGPS Data Used for NAV
158	**NAV-DOP**	0x01 0x04	18	Periodic/Polled	Dilution of precision
159	**NAV-POSECEF**	0x01 0x01	20	Periodic/Polled	Position Solution in ECEF
159	**NAV-POSLLH**	0x01 0x02	28	Periodic/Polled	Geodetic Position Solution
160	**NAV-PVT**	0x01 0x07	84	Periodic/Polled	Navigation Position Velocity Time Solution
162	**NAV-SBAS**	0x01 0x32	12 + 12*cnt	Periodic/Polled	SBAS Status Data
163	**NAV-SOL**	0x01 0x06	52	Periodic/Polled	Navigation Solution Information
165	**NAV-STATUS**	0x01 0x03	16	Periodic/Polled	Receiver Navigation Status

UBX Messages Overview continued

Page	*Mnemonic*	*Cls/ID*	*Length*	*Type*	*Description*
167	**NAV-SVINFO**	0x01 0x30	8 + 12*numCh	Periodic/Polled	Space Vehicle Information
169	**NAV-TIMEGPS**	0x01 0x20	16	Periodic/Polled	GPS Time Solution
170	**NAV-TIMEUTC**	0x01 0x21	20	Periodic/Polled	UTC Time Solution
171	**NAV-VELECEF**	0x01 0x11	20	Periodic/Polled	Velocity Solution in ECEF
171	**NAV-VELNED**	0x01 0x12	36	Periodic/Polled	Velocity Solution in NED
UBX Class RXM				**Receiver Manager Messages**	
173	**RXM-ALM**	0x02 0x30	0	Poll Request	Poll GPS Constellation Almanac Data
173	**RXM-ALM**	0x02 0x30	1	Poll Request	Poll GPS Constellation Almanac Data for a SV
174	**RXM-ALM**	0x02 0x30	(8) or (40)	Poll Answer / Periodic	GPS Aiding Almanac Input/Output Message
174	**RXM-EPH**	0x02 0x31	0	Poll Request	Poll GPS Constellation Ephemeris Data
175	**RXM-EPH**	0x02 0x31	1	Poll Request	Poll GPS Constellation Ephemeris Data for a SV
175	**RXM-EPH**	0x02 0x31	(8) or (104)	Poll Answer / Periodic	GPS Aiding Ephemeris Input/Output Message
176	**RXM-PMREQ**	0x02 0x41	8	Command	Requests a Power Management task
176	**RXM-RAW**	0x02 0x10	8 + 24*numSV	Periodic/Polled	Raw Measurement Data
177	**RXM-SFRB**	0x02 0x11	42	Periodic	Subframe Buffer
178	**RXM-SVSI**	0x02 0x20	8 + 6*numSV	Periodic/Polled	SV Status Info
UBX Class TIM				**Timing Messages**	
180	**TIM-TM2**	0x0D 0x03	28	Periodic/Polled	Time mark data
181	**TIM-TP**	0x0D 0x01	16	Periodic/Polled	Time Pulse Timedata
182	**TIM-VRFY**	0x0D 0x06	20	Polled/Once	Sourced Time Verification

33 ACK (0x05)

Ack/Nack Messages: i.e. as replies to CFG Input Messages.
Messages in this class are sent as a result of a CFG message being received, decoded and processed by the receiver.

33.1 ACK-ACK (0x05 0x01)

33.1.1 Message Acknowledged

Message	**ACK-ACK**				
Description	**Message Acknowledged**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Output				
Comment	Output upon processing of an input message				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x05 0x01	2	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`clsID`	-	Class ID of the Acknowledged Message
1	U1	-	`msgID`	-	Message ID of the Acknowledged Message

33.2 ACK-NAK (0x05 0x00)

33.2.1 Message Not-Acknowledged

Message	**ACK-NAK**				
Description	**Message Not-Acknowledged**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Output				
Comment	Output upon processing of an input message				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x05 0x00	2	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`clsID`	-	Class ID of the Not-Acknowledged Message
1	U1	-	`msgID`	-	Message ID of the Not-Acknowledged Message

34 AID (0x0B)

AssistNow Aiding Messages: i.e. Ephemeris, Almanac, other A-GPS data input.
Messages in this class are used to send aiding data to the receiver.

34.1 AID-ALM (0x0B 0x30)

34.1.1 Poll GPS Aiding Almanac Data

Message	**AID-ALM**				
Description	**Poll GPS Aiding Almanac Data**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	**This message has an empty payload!** Poll GPS Aiding Data (Almanac) for all 32 SVs by sending this message to the receiver without any payload. The receiver will return 32 messages of type AID-ALM as defined below.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0B 0x30	0	*see below*	CK_A CK_B
No payload					

34.1.2 Poll GPS Aiding Almanac Data for a SV

Message	**AID-ALM**				
Description	**Poll GPS Aiding Almanac Data for a SV**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	Poll GPS Aiding Data (Almanac) for an SV by sending this message to the receiver. The receiver will return one message of type AID-ALM as defined below.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0B 0x30	1	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`svid`	-	SV ID for which the receiver shall return its Almanac Data (Valid Range: 1 .. 32 or 51, 56, 63).

34.1.3 GPS Aiding Almanac Input/Output Message

Message	**AID-ALM**				
Description	**GPS Aiding Almanac Input/Output Message**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	• If the WEEK Value is 0, DWRD0 to DWRD7 are not sent as the Almanac is not available for the given SV. This may happen even if NAV-SVINFO and RXM-SVSI are indicating almanac availability as the internal data may not represent the content of an original broadcast almanac (or only parts thereof). • DWORD0 to DWORD7 contain the 8 words following the Hand-Over Word (HOW) from the GPS navigation message, either pages 1 to 24 of sub-frame 5 or pages 2 to 10 of subframe 4. See IS-GPS-200 for a full description of the contents of the Almanac pages. • In DWORD0 to DWORD7, the parity bits have been removed, and the 24 bits of data are located in Bits 0 to 23. Bits 24 to 31 shall be ignored. • Example: Parameter e (Eccentricity) from Almanac Subframe 4/5, Word 3, Bits 69-84 within the subframe can be found in DWRD0, Bits 15-0 whereas Bit 0 is the LSB.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0B 0x30	(8) or (40)	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`svid`	-	SV ID for which this Almanac Data is (Valid Range: 1 .. 32 or 51, 56, 63).
4	U4	-	`week`	-	Issue Date of Almanac (GPS week number)
Start of optional block					
8	U4[8]	-	`dwrd`	-	Almanac Words
End of optional block					

34.2 AID-ALPSRV (0x0B 0x32)

34.2.1 ALP client requests AlmanacPlus data from server

Message	**AID-ALPSRV**				
Description	**ALP client requests AlmanacPlus data from server**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Output				
Comment	This message is sent by the ALP client to the ALP server in order to request data. The given identifier must be prepended to the requested data when submitting the data.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0B 0x32	16	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*

AID-ALPSRV continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`idSize`	bytes	Identifier size. This data, beginning at message start, must prepend the returned data.
1	U1	-	`type`	-	Requested data type. Must be different from 0xff, otherwise this is not a data request.
2	U2	-	`ofs`	-	Requested data offset [16bit words]
4	U2	-	`size`	-	Requested data size [16bit words]
6	U2	-	`fileId`	-	Unused when requesting data, filled in when sending back the data
8	U2	-	`dataSize`	bytes	Actual data size. Unused when requesting data, filled in when sending back the data.
10	U1	-	`id1`	-	Identifier data
11	U1	-	`id2`	-	Identifier data
12	U4	-	`id3`	-	Identifier data

34.2.2 ALP server sends AlmanacPlus data to client

Message	**AID-ALPSRV**				
Description	**ALP server sends AlmanacPlus data to client**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input				
Comment	This message is sent by the ALP server to the ALP client and is usually sent in response to a data request. The server copies the identifier from the request and fills in the dataSize and fileId fields.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0B 0x32	16 + 1*dataSize	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`idSize`	bytes	Identifier size
1	U1	-	`type`	-	Requested data type
2	U2	-	`ofs`	-	Requested data offset [16bit words]
4	U2	-	`size`	-	Requested data size [16bit words]
6	U2	-	`fileId`	-	Corresponding ALP file ID, must be filled in by the server!
8	U2	-	`dataSize`	bytes	Actual data contained in this message, must be filled in by the server!
10	U1	-	`id1`	-	Identifier data
11	U1	-	`id2`	-	Identifier data
12	U4	-	`id3`	-	Identifier data
Start of repeated block (dataSize times)					
16 + 1*N	U1	-	`data`	-	Data for the ALP client
End of repeated block					

34.2.3 ALP client sends AlmanacPlus data to server.

Message	**AID-ALPSRV**				
Description	**ALP client sends AlmanacPlus data to server.**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Output				
Comment	This message is sent by the ALP client to the ALP server in order to submit updated data. The server can either replace the current data at this position or ignore this new data (which will result in degraded performance).				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0B 0x32	8 + 2*size	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`idSize`	bytes	Identifier size
1	U1	-	`type`	-	Set to 0xff to mark that is *not* a data request
2	U2	-	`ofs`	-	Data offset [16bit words]
4	U2	-	`size`	-	Data size [16bit words]
6	U2	-	`fileId`	-	Corresponding ALP file id
Start of repeated block (size times)					
8 + 2*N	U2	-	`data`	-	16bit word data to be submitted to the ALP server
End of repeated block					

34.3 AID-ALP (0x0B 0x50)

34.3.1 ALP file data transfer to the receiver

Message	**AID-ALP**				
Description	**ALP file data transfer to the receiver**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input				
Comment	This message is used to transfer a chunk of data from the AlmanacPlus file to the receiver. Upon reception of this message, the receiver will write the payload data to its internal non-volatile memory, eventually also erasing that part of the memory first. Make sure that the payload size is even sized (i.e. always a multiple of 2). Do not use payloads larger than ~ 700 bytes, as this would exceed the receiver's internal buffering capabilities. The receiver will (not-) acknowledge this message using the message alternatives given below. The host shall wait for an acknowledge message before sending the next chunk.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0B 0x50	0 + 2*N	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
Start of repeated block (N times)					
N*2	U2	-	`alpData`	-	ALP file data

AID-ALP continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
End of repeated block					

34.3.2 Mark end of data transfer

Message	**AID-ALP**				
Description	**Mark end of data transfer**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input				
Comment	This message is used to indicate that all chunks have been transferred, and normal receiver operation can resume. Upon reception of this message, the receiver will verify all chunks received so far, and enable AssistNow Offline and GPS receiver operation if successful. This message could also be sent to cancel an incomplete download.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0B 0x50	1	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`dummy`	-	Value is ignored

34.3.3 Acknowledges a data transfer

Message	**AID-ALP**				
Description	**Acknowledges a data transfer**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Output				
Comment	This message from the receiver acknowledges successful processing of a previously received chunk of data with the ``Chunk Transfer`` Message. This message will also be sent once a ``Stop`` message has been received, and the integrity of all chunks received so far has been checked successfully.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0B 0x50	1	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`ack`	-	Set to 0x01

34.3.4 Indicate problems with a data transfer

Message	**AID-ALP**				
Description	**Indicate problems with a data transfer**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Output				
Comment	This message from the receiver indicates that an error has occurred while processing and storing the data received with the ``Chunk Transfer`` message. This message will also be sent once a stop command has been received, and the integrity of all chunks received failed.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0B 0x50	1	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`nak`	-	Set to 0x00

34.3.5 Poll the AlmanacPlus status

Message	**AID-ALP**				
Description	**Poll the AlmanacPlus status**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0B 0x50	24	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`predTow`	s	Prediction start time of week
4	U4	-	`predDur`	s	Prediction duration from start of first data set to end of last data set
8	I4	-	`age`	s	Current age of ALP data
12	U2	-	`predWno`	-	Prediction start week number
14	U2	-	`almWno`	-	Truncated week number of reference almanac
16	U4	-	`reserved1`	-	Reserved
20	U1	-	`svs`	-	Number of satellite data sets contained in the ALP data
21	U1	-	`reserved2`	-	Reserved
22	U2	-	`reserved3`	-	Reserved

34.4 AID-AOP (0x0B 0x33)

34.4.1 Poll AssistNow Autonomous data

Message	**AID-AOP**				
Description	**Poll AssistNow Autonomous data**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll request				
Comment	**This message has an empty payload.** Poll *AssistNow Autonomous* aiding data for all satellits by sending this empty message. The receiver will return an AID-AOP message (see definition below) for each satellite for which data is available. For satellites for which no data is available it will return a corresponding AID-AOP poll request message (see below).				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0B 0x33	0	*see below*	CK_A CK_B
No payload					

34.4.2 Poll AssistNow Autonomous data for one satellite

Message	**AID-AOP**				
Description	**Poll AssistNow Autonomous data for one satellite**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll request				
Comment	Poll the *AssistNow Autonomous* data for the specified satellite. The receiver will return a AID-AOP message (see definition below) if data is available for the requested satellite. If no data is available it will return corresponding AID-AOP poll request message (i.e. this message).				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0B 0x33	1	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`svid`	-	GPS SV id for which the data is requested (valid range: 1..32).

34.4.3 AssistNow Autonomous data

Message	**AID-AOP**				
Description	**AssistNow Autonomous data**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	If enabled, this message is output at irregular intervals. It is output whenever *AssistNow Autonomous* has produced new data for a satellite. Depending on the availability of the optional data the receiver will output either version of the message. If this message is polled using one of the two poll requests described above the receiver will send this message if AOP data is available or the corresponding poll request message if no AOP data is available for each satellite (i.e. svid 1..32). At the user's choice the optional data may be chopped from the payload of a previously polled message when sending the message back to the receiver. Sending a valid AID-AOP message to the receiver will automatically enable the *AssistNow Autonomous* feature on the receiver. See the section AssistNow Autonomous in the receiver description for details on this feature.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0B 0x33	(60) or (204)	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`svid`	-	GPS SV id
1	U1[59]	-	`data`	-	*AssistNow Autonomous* data
Start of optional block					
60	U1[48]	-	`optional0`	-	Optional data chunk 1/3
108	U1[48]	-	`optional1`	-	Optional data chunk 2/3
156	U1[48]	-	`optional2`	-	Optional data chunk 3/3
End of optional block					

34.5 AID-DATA (0x0B 0x10)

34.5.1 Polls all GPS Initial Aiding Data

Message	**AID-DATA**				
Description	**Polls all GPS Initial Aiding Data**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	If this poll is received, the messages AID-INI, AID-HUI, AID-EPH and AID-ALM are sent.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0B 0x10	0	*see below*	CK_A CK_B
No payload					

34.6 AID-EPH (0x0B 0x31)

34.6.1 Poll GPS Aiding Ephemeris Data

Message	**AID-EPH**				
Description	**Poll GPS Aiding Ephemeris Data**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	**This message has an empty payload!** Poll GPS Aiding Data (Ephemeris) for all 32 SVs by sending this message to the receiver without any payload. The receiver will return 32 messages of type AID-EPH as defined below.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0B 0x31	0	*see below*	CK_A CK_B
No payload					

34.6.2 Poll GPS Aiding Ephemeris Data for a SV

Message	**AID-EPH**				
Description	**Poll GPS Aiding Ephemeris Data for a SV**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	Poll GPS Constellation Data (Ephemeris) for an SV by sending this message to the receiver. The receiver will return one message of type AID-EPH as defined below.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0B 0x31	1	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`svid`	-	SV ID for which the receiver shall return its Ephemeris Data (Valid Range: 1 .. 32).

34.6.3 GPS Aiding Ephemeris Input/Output Message

Message	**AID-EPH**
Description	**GPS Aiding Ephemeris Input/Output Message**
Firmware	Supported on: • u-blox 7 firmware version 1.00
Type	Input/Output
Comment	• SF1D0 to SF3D7 is only sent if ephemeris is available for this SV. If not, the payload may be reduced to 8 Bytes, or all bytes are set to zero, indicating that this SV Number does not have valid ephemeris for the moment. This may happen even if NAV-SVINFO and RXM-SVSI are indicating ephemeris availability as the internal data may not represent the content of an original broadcast ephemeris (or only parts thereof). • SF1D0 to SF3D7 contain the 24 words following the Hand-Over Word (HOW) from the GPS navigation message, subframes 1 to 3. The Truncated TOW Count is not valid and cannot be used. See IS-GPS-200 for a full description of the contents of the Subframes.

- In SF1D0 to SF3D7, the parity bits have been removed, and the 24 bits of data are located in Bits 0 to 23. Bits 24 to 31 shall be ignored.
- When polled, the data contained in this message does not represent the full original ephemeris broadcast. Some fields that are irrelevant to u-blox receivers may be missing. The week number in Subframe 1 has already been modified to match the Time Of Ephemeris (TOE).

	Header	ID	Length (Bytes)	Payload	Checksum
Message Structure	0xB5 0x62	0x0B 0x31	(8) or (104)	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	Number Format	Scaling	Name	Unit	Description
0	U4	-	`svid`	-	SV ID for which this ephemeris data is (Valid Range: 1 .. 32).
4	U4	-	`how`	-	Hand-Over Word of first Subframe. This is required if data is sent to the receiver. 0 indicates that no Ephemeris Data is following.
Start of optional block					
8	U4[8]	-	`sf1d`	-	Subframe 1 Words 3..10 (SF1D0..SF1D7)
40	U4[8]	-	`sf2d`	-	Subframe 2 Words 3..10 (SF2D0..SF2D7)
72	U4[8]	-	`sf3d`	-	Subframe 3 Words 3..10 (SF3D0..SF3D7)
End of optional block					

34.7 AID-HUI (0x0B 0x02)

34.7.1 Poll GPS Health, UTC and ionosphere parameters

Message	**AID-HUI**
Description	**Poll GPS Health, UTC and ionosphere parameters**
Firmware	Supported on: • u-blox 7 firmware version 1.00
Type	Poll Request
Comment	**This message has an empty payload!** -

	Header	ID	Length (Bytes)	Payload	Checksum
Message Structure	0xB5 0x62	0x0B 0x02	0	*see below*	CK_A CK_B
No payload					

34.7.2 GPS Health, UTC and ionosphere parameters

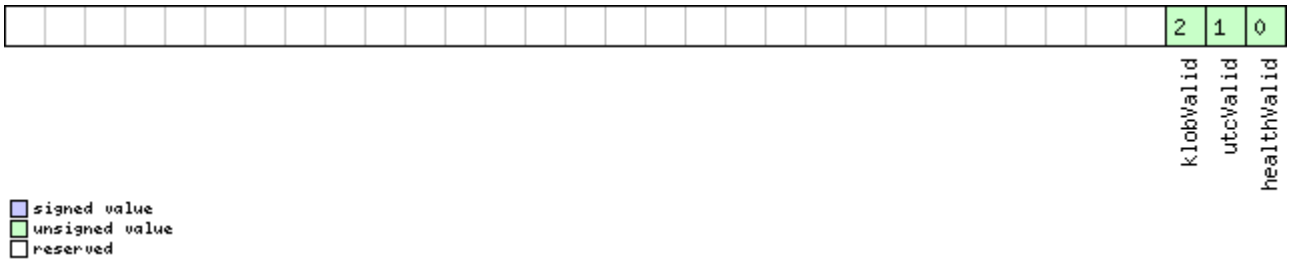
Message	**AID-HUI**				
Description	**GPS Health, UTC and ionosphere parameters**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	This message contains a health bit mask, UTC time and Klobuchar parameters. For more information on these parameters, please see the ICD-GPS-200 documentation.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0B 0x02	72	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	X4	-	`health`	-	Bitmask, every bit represenst a GPS SV (1-32). If the bit is set the SV is healthy.
4	R8	-	`utcA0`	-	UTC - parameter A0
12	R8	-	`utcA1`	-	UTC - parameter A1
20	I4	-	`utcTOW`	-	UTC - reference time of week
24	I2	-	`utcWNT`	-	UTC - reference week number
26	I2	-	`utcLS`	-	UTC - time difference due to leap seconds before event
28	I2	-	`utcWNF`	-	UTC - week number when next leap second event occurs
30	I2	-	`utcDN`	-	UTC - day of week when next leap second event occurs
32	I2	-	`utcLSF`	-	UTC - time difference due to leap seconds after event
34	I2	-	`utcSpare`	-	UTC - Spare to ensure structure is a multiple of 4 bytes
36	R4	-	`klobA0`	s	Klobuchar - alpha 0
40	R4	-	`klobA1`	s/semicircle	Klobuchar - alpha 1
44	R4	-	`klobA2`	s/semicircle^2	Klobuchar - alpha 2
48	R4	-	`klobA3`	s/semicircle^3	Klobuchar - alpha 3
52	R4	-	`klobB0`	s	Klobuchar - beta 0
56	R4	-	`klobB1`	s/semicircle	Klobuchar - beta 1
60	R4	-	`klobB2`	s/semicircle^2	Klobuchar - beta 2
64	R4	-	`klobB3`	s/semicircle^3	Klobuchar - beta 3
68	X4	-	`flags`	-	flags (see graphic below)

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`healthValid`	Healthmask field in this message is valid
`utcValid`	UTC parameter fields in this message are valid
`klobValid`	Klobuchar parameter fields in this message are valid

34.8 AID-INI (0x0B 0x01)

34.8.1 Poll GPS Initial Aiding Data

Message	**AID-INI**				
Description	**Poll GPS Initial Aiding Data**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	**This message has an empty payload!** -				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0B 0x01	0	*see below*	CK_A CK_B
No payload					

34.8.2 Aiding position, time, frequency, clock drift

Message	**AID-INI**				
Description	**Aiding position, time, frequency, clock drift**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	This message contains position, time and clock drift information. The position can be input in either the ECEF X/Y/Z coordinate system or as lat/lon/height. The time can either be input as inexact value via the standard communication interface, suffering from latency depending on the baudrate, or using harware time synchronization where an accuracte time pulse is input on the external interrupts. It is also possible to supply hardware frequency aiding by connecting a continuous signal to an external interrupt.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0B 0x01	48	*see below*	CK_A CK_B

Payload Contents:

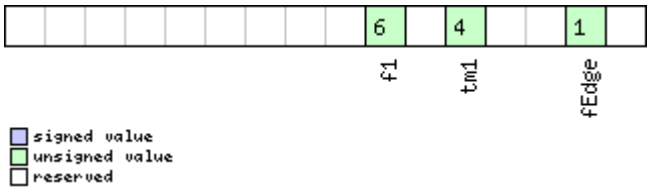
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	I4	-	`ecefXOrLat`	cm_or_deg*1e-7	WGS84 ECEF X coordinate or latitude, depending on flags below

AID-INI continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
4	I4	-	`ecefYOrLon`	cm_or_deg*1e-7	WGS84 ECEF Y coordinate or longitude, depending on flags below
8	I4	-	`ecefZOrAlt`	cm	WGS84 ECEF Z coordinate or altitude, depending on flags below
12	U4	-	`posAcc`	cm	Position accuracy (stddev)
16	X2	-	`tmCfg`	-	Time mark configuration (see graphic below)
18	U2	-	`wnoOrDate`	week_or_yearMonth	Actual week number or yearSince2000/Month (YYMM), depending on flags below
20	U4	-	`towOrTime`	ms_or_dayHourMinuteSec	Actual time of week or DayOfMonth/Hour/Minute/Second (DDHHMMSS), depending on flags below
24	I4	-	`towNs`	ns	Fractional part of time of week
28	U4	-	`tAccMs`	ms	Milliseconds part of time accuracy
32	U4	-	`tAccNs`	ns	Nanoseconds part of time accuracy
36	I4	-	`clkDOrFreq`	ns/s_or_Hz*1e-2	Clock drift or frequency, depending on flags below
40	U4	-	`clkDAccOrFreqAcc`	ns/s_or_ppb	Accuracy of clock drift or frequency, depending on flags below
44	X4	-	`flags`	-	Bitmask with the following flags (see graphic below)

Bitfield tmCfg

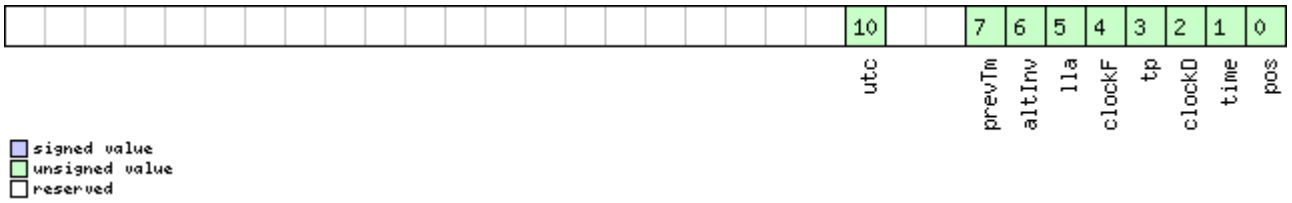
This Graphic explains the bits of `tmCfg`



Name	*Description*
`fEdge`	use falling edge (default rising)
`tm1`	time mark on extint 1 (default extint 0)
`f1`	frequency on extint 1 (default extint 0)

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`pos`	Position is valid
`time`	Time is valid
`clockD`	Clock drift data contains valid clock drift, must not be set together with clockF
`tp`	Use time pulse
`clockF`	Clock drift data contains valid frequency, must not be set together with clockD
`lla`	Position is given in lat/long/alt (default is ECEF)
`altInv`	Altitude is not valid, in case lla was set
`prevTm`	Use time mark received before AID-INI message (default uses mark received after message)
`utc`	Time is given as UTC date/time (default is GPS wno/tow)

34.9 AID-REQ (0x0B 0x00)

34.9.1 Sends a poll (AID-DATA) for all GPS Aiding Data

Message	**AID-REQ**				
Description	**Sends a poll (AID-DATA) for all GPS Aiding Data**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Virtual				
Comment	**AID-REQ is not a message but a placeholder for configuration purposes.** If the virtual AID-REQ is configured to be output (see CFG-MSG), the receiver will output a request for aiding data (AID-DATA) after a start-up if its internally stored data (position, time) don't allow it to perform a hot start. If position and time information could be retrieved from internal storage, no AID-REQ will be sent, even when the receiver is missing valid ephemeris data. Only GPS orbits are supported for GNSS.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0B 0x00	0	*see below*	CK_A CK_B
No payload					

35 CFG (0x06)

Configuration Input Messages: i.e. Set Dynamic Model, Set DOP Mask, Set Baud Rate, etc..
The CFG Class can be used to configure the receiver and read out current configuration values. Any messages in Class CFG sent to the receiver are acknowledged (with Message ACK-ACK) if processed successfully, and rejected (with Message ACK-NAK) if processing the message failed.

35.1 CFG-ANT (0x06 0x13)

35.1.1 Poll Antenna Control Settings

Message	**CFG-ANT**				
Description	**Poll Antenna Control Settings**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	Sending this (empty / no-payload) message to the receiver results in the receiver returning a message of type CFG-ANT with a payload as defined below				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x13	0	*see below*	CK_A CK_B
No payload					

35.1.2 Antenna Control Settings

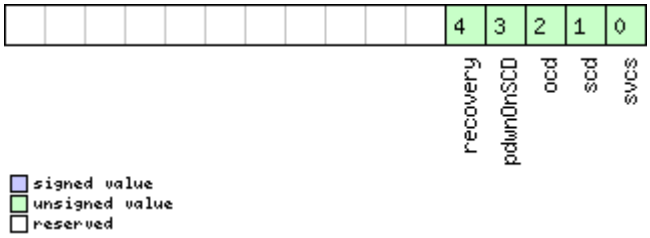
Message	**CFG-ANT**				
Description	**Antenna Control Settings**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x13	4	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	X2	-	`flags`	-	Antenna Flag Mask (see graphic below)
2	X2	-	`pins`	-	Antenna Pin Configuration (see graphic below)

Bitfield flags

This Graphic explains the bits of `flags`



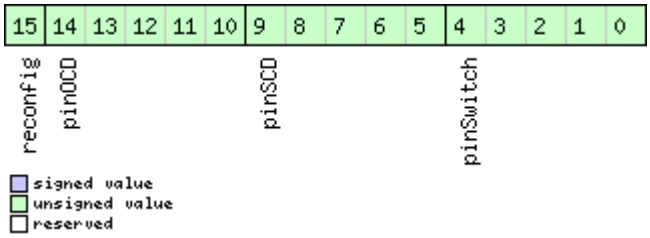
Name	*Description*
`svcs`	Enable Antenna Supply Voltage Control Signal
`scd`	Enable Short Circuit Detection

Bitfield flags Description continued

Name	*Description*
`ocd`	Enable Open Circuit Detection
`pdwnOnSCD`	Power Down Antenna supply if Short Circuit is detected. (only in combination with Bit 1)
`recovery`	Enable automatic recovery from short state

Bitfield pins

This Graphic explains the bits of `pins`



Name	*Description*
`pinSwitch`	PIO-Pin used for switching antenna supply (internal to TIM-LP/TIM-LF)
`pinSCD`	PIO-Pin used for detecting a short in the antenna supply
`pinOCD`	PIO-Pin used for detecting open/not connected antenna
`reconfig`	if set to one, and this command is sent to the receiver, the receiver will reconfigure the pins as specified.

35.2 CFG-CFG (0x06 0x09)

35.2.1 Clear, Save and Load configurations

Message	**CFG-CFG**				
Description	**Clear, Save and Load configurations**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Command				
Comment	See the Receiver Configuration chapter for a detailed description on how Receiver Configuration should be used. The three masks are made up of individual bits, each bit indicating the sub-section of all configurations on which the corresponding action shall be carried out. The reserved bits in the masks must be set to '0'. For detailed information please refer to the Organization of the Configuration Sections. Please note that commands can be combined. The sequence of execution is Clear, Save, Load				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x09	(12) or (13)	*see below*	CK_A CK_B

Payload Contents:

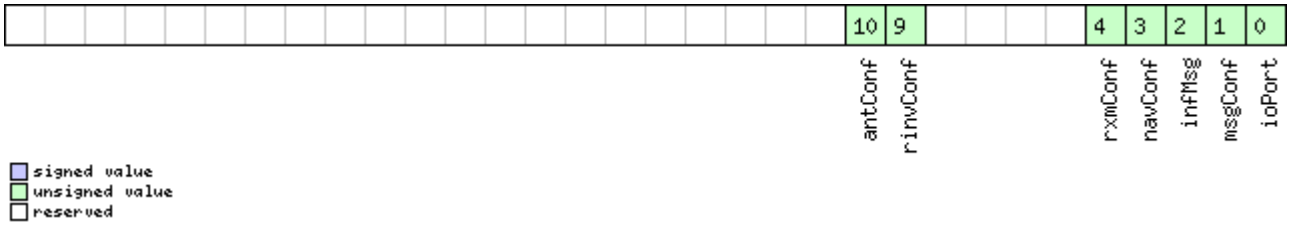
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	X4	-	`clearMask`	-	Mask with configuration sub-sections to Clear (=Load Default Configurations to Permanent Configurations in non-volatile memory) (see graphic below)
4	X4	-	`saveMask`	-	Mask with configuration sub-section to Save (=Save Current Configuration to Non-volatile Memory), see ID description of clearMask

CFG-CFG continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
8	X4	-	`loadMask`	-	Mask with configuration sub-sections to Load (=Load Permanent Configurations from Non-volatile Memory to Current Configurations), see ID description of clearMask
Start of optional block					
12	X1	-	`deviceMask`	-	Mask which selects the devices for this command. (see graphic below)
End of optional block					

Bitfield clearMask

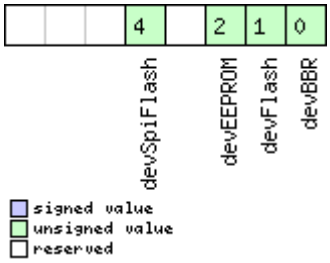
This Graphic explains the bits of `clearMask`



Name	*Description*
`ioPort`	Port Settings
`msgConf`	Message Configuration
`infMsg`	INF Message Configuration
`navConf`	Navigation Configuration
`rxmConf`	Receiver Manager Configuration
`rinvConf`	Remote Inventory Configuration
`antConf`	Antenna Configuration

Bitfield deviceMask

This Graphic explains the bits of `deviceMask`



Name	*Description*
`devBBR`	device battery backed RAM
`devFlash`	device Flash
`devEEPROM`	device EEPROM
`devSpiFlash`	device SPI Flash

35.3 CFG-DAT (0x06 0x06)

35.3.1 Poll Datum Setting

Message	**CFG-DAT**				
Description	**Poll Datum Setting**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	Upon sending of this message, the receiver returns CFG-DAT as defined below				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x06	0	*see below*	CK_A CK_B
No payload					

35.3.2 Set User-defined Datum

Message	**CFG-DAT**				
Description	**Set User-defined Datum**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x06	44	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	R8	-	`majA`	m	Semi-major Axis (accepted range = 6,300,000.0 to 6,500,000.0 metres).
8	R8	-	`flat`	-	1.0 / Flattening (accepted range is 0.0 to 500.0).
16	R4	-	`dX`	m	X Axis shift at the origin (accepted range is +/- 5000.0 metres).
20	R4	-	`dY`	m	Y Axis shift at the origin (accepted range is +/- 5000.0 metres).
24	R4	-	`dZ`	m	Z Axis shift at the origin (accepted range is +/- 5000.0 metres).
28	R4	-	`rotX`	s	Rotation about the X Axis (accepted range is +/- 20.0 milli-arc seconds).
32	R4	-	`rotY`	s	Rotation about the Y Axis (accepted range is +/- 20.0 milli-arc seconds).
36	R4	-	`rotZ`	s	Rotation about the Z Axis (accepted range is +/- 20.0 milli-arc seconds).
40	R4	-	`scale`	ppm	Scale change (accepted range is 0.0 to 50.0 parts per million).

35.3.3 The currently defined Datum

Message	**CFG-DAT**				
Description	**The currently defined Datum**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Output				
Comment	Returns the parameters of the currently defined datum. If no user-defined datum has been set, this will default to WGS84.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x06 0x06	52	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U2	-	`datumNum`	-	Datum Number: 0 = WGS84, -1 = user-defined
2	CH[6]	-	`datumName`	-	ASCII String: WGS84 or USER
8	R8	-	`majA`	m	Semi-major Axis (accepted range = 6,300,000.0 to 6,500,000.0 metres).
16	R8	-	`flat`	-	1.0 / Flattening (accepted range is 0.0 to 500.0).
24	R4	-	`dX`	m	X Axis shift at the origin (accepted range is +/- 5000.0 metres).
28	R4	-	`dY`	m	Y Axis shift at the origin (accepted range is +/- 5000.0 metres).
32	R4	-	`dZ`	m	Z Axis shift at the origin (accepted range is +/- 5000.0 metres).
36	R4	-	`rotX`	s	Rotation about the X Axis (accepted range is +/- 20.0 milli-arc seconds).
40	R4	-	`rotY`	s	Rotation about the Y Axis (accepted range is +/- 20.0 milli-arc seconds).
44	R4	-	`rotZ`	s	Rotation about the Z Axis (accepted range is +/- 20.0 milli-arc seconds).
48	R4	-	`scale`	ppm	Scale change (accepted range is 0.0 to 50.0 parts per million).

35.4 CFG-GNSS (0x06 0x3E)

35.4.1 Polls the configuration of the GNSS system configuration

Message	**CFG-GNSS**				
Description	**Polls the configuration of the GNSS system configuration**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	Polls the configuration of the GNSS system configuration				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x3E	0	*see below*	CK_A CK_B
No payload					

35.4.2 GNSS system configuration

Message	**CFG-GNSS**				
Description	**GNSS system configuration**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	Gets or sets the GNSS system channel sharing configuration. The receiver will send an UBX-ACK-ACK message if the configuration is valid, an UBX-ACK-NAK if any configuration parameter is invalid. The number of tracking channels in use must not exceed the number of tracking channels available on hardware, and the sum of all reserved tracking channels needs to be smaller or equal the number of tracking channels in use. Additionally, the maximum number of tracking channels used for the specific GNSS system must be greater or equal to the number of reserved tracking channels. See section GNSS Configuration for a discussion of the use of this message and section Satellite Numbering for a description of the GNSS IDs available. Configuration specific to the GNSS system can be done via other messages. Configuration specific to SBAS can be done with `CFG-SBAS`. Note that GLONASS operation cannot be selected when the receiver is configured to operate in Power Save Mode (using `CFG-RXM`).				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x3E	4 + 8*numConfigBlocks	*see below*	CK_A CK_B

Payload Contents:

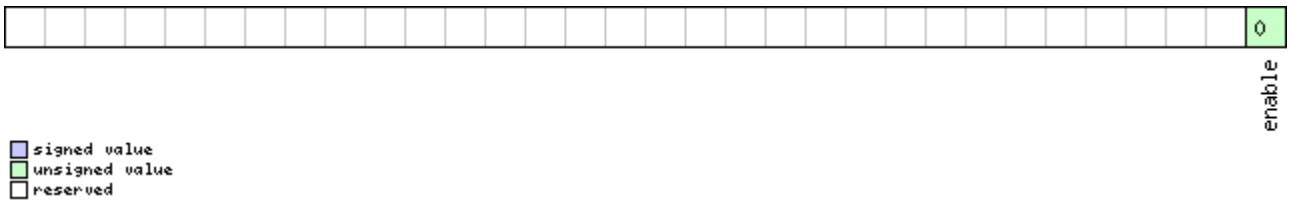
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`msgVer`	-	Message version (=0 for this version)
1	U1	-	`numTrkChHw`	-	Number of tracking channels available in hardware (read only)
2	U1	-	`numTrkChUse`	-	Number of tracking channels to use (<= numTrkChHw)
3	U1	-	`numConfigBlocks`	-	Number of configuration blocks following
Start of repeated block (numConfigBlocks times)					
4 + 8*N	U1	-	`gnssId`	-	GNSS identifier (see Satellite Numbering)

CFG-GNSS continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
5 + 8*N	U1	-	`resTrkCh`	-	Number of reserved (minimum) tracking channels for this GNSS system
6 + 8*N	U1	-	`maxTrkCh`	-	Maximum number of tracking channels used for this GNSS system (>=resTrkChn)
7 + 8*N	U1	-	`reserved1`	-	Reserved
8 + 8*N	X4	-	`flags`	-	bitfield of flags (see graphic below)
End of repeated block					

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`enable`	Enable this GNSS system

35.5 CFG-INF (0x06 0x02)

35.5.1 Poll INF message configuration for one protocol

Message	**CFG-INF**				
Description	**Poll INF message configuration for one protocol**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x02	1	*see below*	CK_A CK_B
Payload Contents:					

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`protocolID`	-	Protocol Identifier, identifying the output protocol for this Poll Request. The following are valid Protocol Identifiers: 0: UBX Protocol 1: NMEA Protocol 2-255: Reserved

35.5.2 Information message configuration

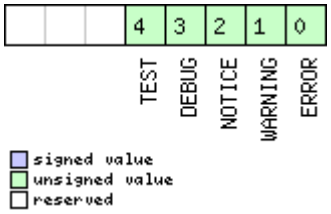
Message	**CFG-INF**				
Description	**Information message configuration**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	The value of infMsgMask[x] below are that each bit represents one of the INF class messages (Bit 0 for ERROR, Bit 1 for WARNING and so on.). For a complete list, please see the `Message Class INF`. Several configurations can be concatenated to one input message. In this case the payload length can be a multiple of the normal length. Output messages from the module contain only one configuration unit. Please note that I/O Ports 1 and 2 correspond to serial ports 1 and 2. I/O port 0 is DDC. I/O port 3 is USB. I/O port 4 is SPI. I/O port 5 is reserved for future use.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x06 0x02	0 + 10*N	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
Start of repeated block (N times)					
N*10	U1	-	`protocolID`	-	Protocol Identifier, identifying for which protocol the configuration is set/get. The following are valid Protocol Identifiers: 0: UBX Protocol 1: NMEA Protocol 2-255: Reserved
1 + 10*N	U1	-	`reserved0`	-	Reserved
2 + 10*N	U2	-	`reserved1`	-	Reserved
4 + 10*N	X1[6]	-	`infMsgMask`	-	A bit mask, saying which information messages are enabled on each I/O port (see graphic below)
End of repeated block					

Bitfield infMsgMask

This Graphic explains the bits of `infMsgMask`



35.6 CFG-ITFM (0x06 0x39)

35.6.1 Polls the Jamming/Interference Monitor configuration.

Message	**CFG-ITFM**				
Description	**Polls the Jamming/Interference Monitor configuration.**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x39	0	*see below*	CK_A CK_B
No payload					

35.6.2 Jamming/Interference Monitor configuration.

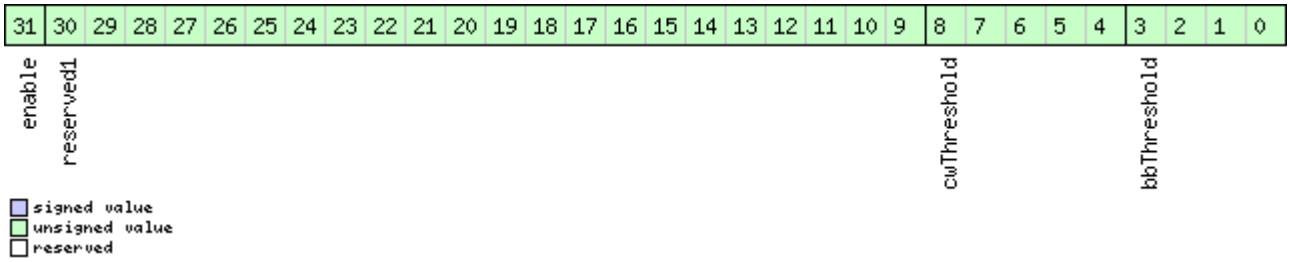
Message	**CFG-ITFM**				
Description	**Jamming/Interference Monitor configuration.**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Command				
Comment	Configuration of Jamming/Interference monitor.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x39	8	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	X4	-	`config`	-	interference config word. (see graphic below)
4	X4	-	`config2`	-	extra settings for jamming/interference monitor (see graphic below)

Bitfield config

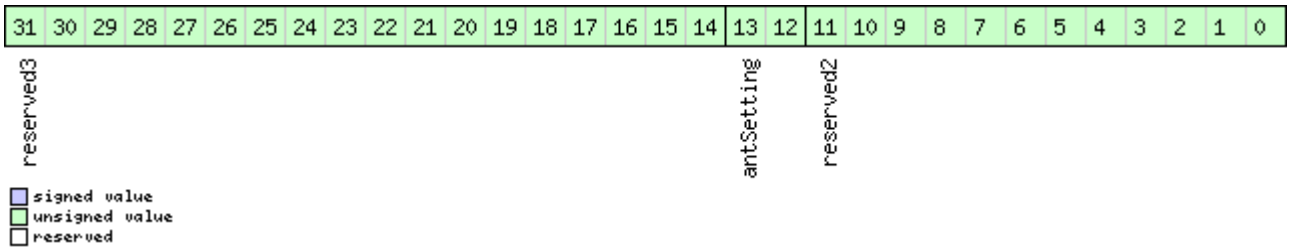
This Graphic explains the bits of `config`



Name	*Description*
`bbThreshold`	Broadband jamming detection threshold (unit = dB)
`cwThreshold`	CW jamming detection threshold (unit = dB)
`reserved1`	reserved algorithm settings - should be set to 0x16B156 in hex for correct settings
`enable`	enable interference detection

Bitfield config2

This Graphic explains the bits of `config2`



Name	Description
`reserved2`	should be set to 0x31E in hex for correct setting
`antSetting`	antennaSetting, 0=unknown, 1=passive, 2=active
`reserved3`	reserved, set to 0

35.7 CFG-LOGFILTER (0x06 0x47)

35.7.1 Poll Data Logger filter Configuration

Message	**CFG-LOGFILTER**				
Description	**Poll Data Logger filter Configuration**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	Upon sending of this message, the receiver returns CFG-LOGFILTER as defined below				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x47	0	*see below*	CK_A CK_B
No payload					

35.7.2 Data Logger Configuration

Message	**CFG-LOGFILTER**				
Description	**Data Logger Configuration**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	This message is used to enable/disable logging and to get or set the position entry filter settings. Position entries can be filtered based on time difference, position difference or current speed thresholds. Position and speed filtering also have a minimum time interval. A position is logged if any of the thresholds are exceeded. If a threshold is set to zero it is ignored. The maximum rate of position logging is 1Hz. The filter settings will only be applied if the 'applyAllFilterSettings' flag is set. This enables recording to be enabled/disabled without affecting the other settings.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x47	12	*see below*	CK_A CK_B

Payload Contents:

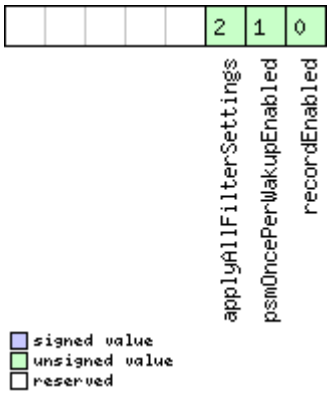
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`version`	-	The version of this message. Set to 1
1	X1	-	`flags`	-	Flags (see graphic below)

CFG-LOGFILTER continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
2	U2	-	`minInterval`	s	Minimum time interval between logged positions (0 = not set). **This is only applied in combination with the speed and/or position thresholds**
4	U2	-	`timeThreshold`	s	If the time difference is greater than the threshold then the position is logged (0 = not set).
6	U2	-	`speedThreshold`	m/s	If the current speed is greater than the threshold then the position is logged (0 = not set). minInterval also applies
8	U4	-	`positionThreshold`	m	If the 3D position difference is greater than the threshold then the position is logged (0 = not set). minInterval also applies

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`recordEnabled`	1 = enable recording, 0 = disable recording
`psmOncePerWakupEnabled`	1 = enable recording only one single position per PSM on/off mode wake up period, 0 = disable once per wake up
`applyAllFilterSettings`	1 = apply all filter settings, 0 = only apply recordEnabled

35.8 CFG-MSG (0x06 0x01)

35.8.1 Poll a message configuration

Message	**CFG-MSG**				
Description	**Poll a message configuration**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x01	2	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`msgClass`	-	Message Class
1	U1	-	`msgID`	-	Message Identifier

35.8.2 Set Message Rate(s)

Message	**CFG-MSG**				
Description	**Set Message Rate(s)**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	Set/Get message rate configuration (s) to/from the receiver. See also section How to change between protocols. • Send rate is relative to the event a message is registered on. For example, if the rate of a navigation message is set to 2, the message is sent every second navigation solution. For configuring NMEA messages, the section NMEA Messages Overview describes Class and Identifier numbers used.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x01	8	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`msgClass`	-	Message Class
1	U1	-	`msgID`	-	Message Identifier
2	U1[6]	-	`rate`	-	Send rate on I/O Port (6 Ports)

35.8.3 Set Message Rate

Message	**CFG-MSG**				
Description	**Set Message Rate**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	Set message rate configuration for the current port. See also section How to change between protocols.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x01	3	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`msgClass`	-	Message Class
1	U1	-	`msgID`	-	Message Identifier
2	U1	-	`rate`	-	Send rate on current Port

35.9 CFG-NAV5 (0x06 0x24)

35.9.1 Poll Navigation Engine Settings

Message	**CFG-NAV5**				
Description	**Poll Navigation Engine Settings**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	Sending this (empty / no-payload) message to the receiver results in the receiver returning a message of type CFG-NAV5 with a payload as defined below.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x24	0	*see below*	CK_A CK_B
No payload					

35.9.2 Navigation Engine Settings

Message	**CFG-NAV5**				
Description	**Navigation Engine Settings**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	See the Navigation Configuration Settings Description for a detailed description of how these settings affect receiver operation.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x24	36	*see below*	CK_A CK_B

Payload Contents:

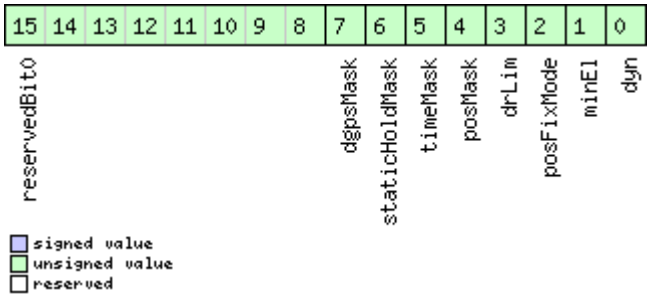
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	X2	-	`mask`	-	Parameters Bitmask. Only the masked parameters will be applied. (see graphic below)

CFG-NAV5 continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
2	U1	-	`dynModel`	-	Dynamic Platform model: 0 Portable 2 Stationary 3 Pedestrian 4 Automotive 5 Sea 6 Airborne with <1g Acceleration 7 Airborne with <2g Acceleration 8 Airborne with <4g Acceleration
3	U1	-	`fixMode`	-	Position Fixing Mode. 1: 2D only 2: 3D only 3: Auto 2D/3D
4	I4	0.01	`fixedAlt`	m	Fixed altitude (mean sea level) for 2D fix mode.
8	U4	0.0001	`fixedAltVar`	m^2	Fixed altitude variance for 2D mode.
12	I1	-	`minElev`	deg	Minimum Elevation for a GNSS satellite to be used in NAV
13	U1	-	`drLimit`	s	Reserved
14	U2	0.1	`pDop`	-	Position DOP Mask to use
16	U2	0.1	`tDop`	-	Time DOP Mask to use
18	U2	-	`pAcc`	m	Position Accuracy Mask
20	U2	-	`tAcc`	m	Time Accuracy Mask
22	U1	-	`staticHoldThresh`	cm/s	Static hold threshold
23	U1	-	`dgpsTimeOut`	s	DGPS timeout.
24	U1	-	`cnoThreshNumSVs`	-	Number of satellites required to have C/N0 above cnoThresh for a fix to be attempted
25	U1	-	`cnoThresh`	dBHz	C/N0 threshold for deciding whether to attempt a fix
26	U2	-	`reserved2`	-	Always set to zero
28	U4	-	`reserved3`	-	Always set to zero
32	U4	-	`reserved4`	-	Always set to zero

Bitfield mask

This Graphic explains the bits of `mask`



Name	*Description*
`dyn`	Apply dynamic model settings

Bitfield mask Description continued

Name	*Description*
`minEl`	Apply minimum elevation settings
`posFixMode`	Apply fix mode settings
`drLim`	Reserved
`posMask`	Apply position mask settings
`timeMask`	Apply time mask settings
`staticHoldMask`	Apply static hold settings
`dgpsMask`	Apply DGPS settings.
`reservedBit0`	reserved

35.10 CFG-NAVX5 (0x06 0x23)

35.10.1 Poll Navigation Engine Expert Settings

Message	**CFG-NAVX5**				
Description	**Poll Navigation Engine Expert Settings**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	Sending this (empty / no-payload) message to the receiver results in the receiver returning a message of type CFG-NAVX5 with a payload as defined below.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x23	0	*see below*	CK_A CK_B
No payload					

35.10.2 Navigation Engine Expert Settings

Message	**CFG-NAVX5**				
Description	**Navigation Engine Expert Settings**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x23	40	*see below*	CK_A CK_B

Payload Contents:

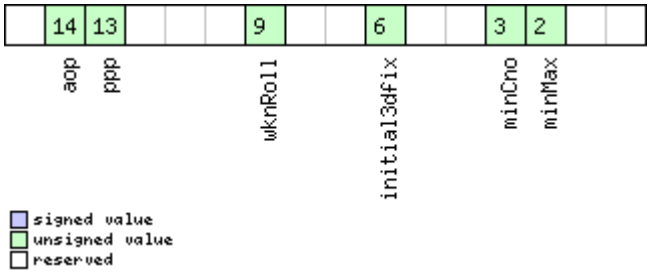
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U2	-	`version`	-	Message version (0 for this version)
2	X2	-	`mask1`	-	First Parameters Bitmask. Only the flagged parameters will be applied, unused bits must be set to 0. (see graphic below)
4	U4	-	`reserved0`	-	Always set to zero
8	U1	-	`reserved1`	-	Always set to zero
9	U1	-	`reserved2`	-	Always set to zero
10	U1	-	`minSVs`	#SVs	Minimum number of satellites for navigation
11	U1	-	`maxSVs`	#SVs	Maximum number of satellites for navigation

CFG-NAVX5 continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
12	U1	-	`minCNO`	dBHz	Minimum satellite signal level for navigation
13	U1	-	`reserved5`	-	Always set to zero
14	U1	-	`iniFix3D`	-	Initial Fix must be 3D flag (0=false/1=true)
15	U1	-	`reserved6`	-	Always set to zero
16	U1	-	`reserved7`	-	Always set to zero
17	U1	-	`reserved8`	-	Always set to zero
18	U2	-	`wknRollover`	-	GPS week rollover number; GPS week numbers will be set correctly from this week up to 1024 weeks after this week. Setting this to 0 reverts to firmware default.
20	U4	-	`reserved9`	-	Always set to zero
24	U1	-	`reserved10`	-	Always set to zero
25	U1	-	`reserved11`	-	Always set to zero
26	U1	-	`usePPP`	-	*Only supported on certain product variants* use Precise Point Positioning flag (0=false/1=true)
27	U1	-	`aopCfg`	-	*AssistNow Autonomous* configuration (see graphic below)
28	U1	-	`reserved12`	-	Always set to zero
29	U1	-	`reserved13`	-	Always set to zero
30	U2	-	`aopOrbMaxErr`	m	maximum acceptable (modelled) *AssistNow Autonomous* orbit error (valid range = 5..1000, or 0 = reset to firmware default)
32	U1	-	`reserved14`	-	Always set to zero
33	U1	-	`reserved15`	-	Always set to zero
34	U2	-	`reserved3`	-	Always set to zero
36	U4	-	`reserved4`	-	Always set to zero

Bitfield mask1

This Graphic explains the bits of `mask1`



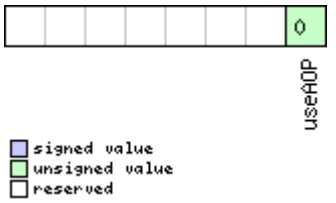
Name	*Description*
`minMax`	Apply min/max SVs settings
`minCno`	Apply minimum C/N0 setting
`initial3dfix`	Apply initial 3D fix settings
`wknRoll`	Apply GPS weeknumber rollover settings
`ppp`	*Only supported on certain product variants* Apply PPP flag

Bitfield mask1 Description continued

Name	*Description*
`aop`	Apply useAOP flag and aopOrbMaxErr setting (AssistNow Autonomous)

Bitfield aopCfg

This Graphic explains the bits of `aopCfg`



Name	*Description*
`useAOP`	AOP enabled flag

35.11 CFG-NMEA (0x06 0x17)

35.11.1 Poll the NMEA protocol configuration

Message	**CFG-NMEA**				
Description	**Poll the NMEA protocol configuration**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x17	0	*see below*	CK_A CK_B
No payload					

35.11.2 NMEA protocol configuration (deprecated)

Message	**CFG-NMEA**				
Description	**NMEA protocol configuration (deprecated)**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	**This message version is provided for backwards compatibility only. Please use the alternative `UBX-CFG-NMEA` message instead** Set/Get the NMEA protocol configuration. See section NMEA Protocol Configuration for a detailed description of the configuration effects on NMEA output.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x17	4	*see below*	CK_A CK_B

Payload Contents:

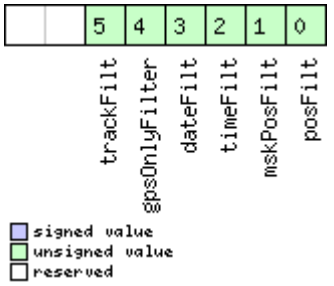
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	X1	-	`filter`	-	filter flags (see graphic below)
1	U1	-	`nmeaVersion`	-	0x23 = NMEA version 2.3 0x21 = NMEA version 2.1

CFG-NMEA continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
2	U1	-	`numSV`	-	Maximum Number of SVs to report in NMEA protocol (0 = unlimited). This does not affect the receiver's operation. It only limits the number of SVs reported in NMEA mode (this might be needed with older mapping applications which only support 8- or 12-channel receivers).
3	X1	-	`flags`	-	flags (see graphic below)

Bitfield filter

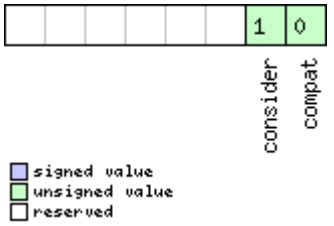
This Graphic explains the bits of `filter`



Name	*Description*
`posFilt`	Enable position output for failed or invalid fixes
`mskPosFilt`	Enable position output for invalid fixes
`timeFilt`	Enable time output for invalid times
`dateFilt`	Enable date output for invalid dates
`gpsOnlyFilter`	Restrict output to GPS satellites only
`trackFilt`	Enable COG output even if COG is frozen

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`compat`	enable compatibility mode. This might be needed for certain applications when customer's NMEA parser expects a fixed number of digits in position coordinates
`consider`	enable considering mode.

35.11.3 NMEA protocol configuration

Message	**CFG-NMEA**				
Description	**NMEA protocol configuration**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	Set/Get the NMEA protocol configuration. See section NMEA Protocol Configuration for a detailed description of the configuration effects on NMEA output.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x17	12	*see below*	CK_A CK_B

Payload Contents:

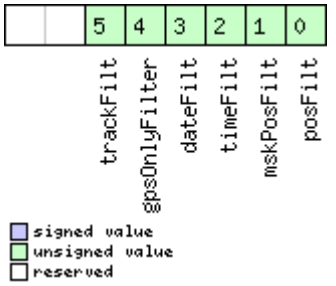
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	X1	-	`filter`	-	filter flags (see graphic below)
1	U1	-	`nmeaVersion`	-	0x23 = NMEA version 2.3 0x21 = NMEA version 2.1
2	U1	-	`numSV`	-	Maximum Number of SVs to report in NMEA protocol. This does not affect the receiver's operation. It only limits the number of SVs reported in NMEA mode (this might be needed with older mapping applications which only support 8- or 12-channel receivers).
3	X1	-	`flags`	-	flags (see graphic below)
4	X4	-	`gnssToFilter`	-	Filters out satellites based on their GNSS. If a bitfield is enabled, the corresponding satellites will be not output. (see graphic below)
8	U1	-	`svNumbering`	-	Configures the display of satellites that do not have an NMEA-defined value. Note: this does not apply to satellites with an unknown ID. 0: Strict - Satellites are not output 1: Extended - Use UBX proprietary numbering (see Satellite numbering)
9	U1	-	`mainTalkerId`	-	By default the main Talker ID (i.e. the Talker ID used for all messages other than GSV) is determined by the GNSS assignment of the receiver's channels (see `UBX-CFG-GNSS`). This field enables the main Talker ID to be overridden. 0: Main Talker ID is not overridden 1: Set main Talker ID to 'GP' 2: Set main Talker ID to 'GL' 3: Set main Talker ID to 'GN'

CFG-NMEA continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
10	U1	-	`gsvTalkerId`	-	By default the Talker ID for GSV messages is GNSS specific (as defined by NMEA). This field enables the GSV Talker ID to be overridden. 0: Use GNSS specific Talker ID (as defined by NMEA) 1: Use the main Talker ID
11	U1	-	`reserved`	-	Reserved, always set to 0

Bitfield filter

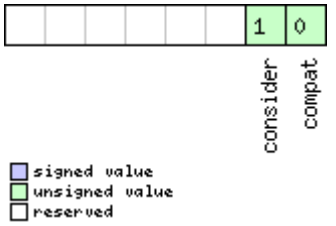
This Graphic explains the bits of `filter`



Name	*Description*
`posFilt`	Enable position output for failed or invalid fixes
`mskPosFilt`	Enable position output for invalid fixes
`timeFilt`	Enable time output for invalid times
`dateFilt`	Enable date output for invalid dates
`gpsOnlyFilter`	Restrict output to GPS satellites only
`trackFilt`	Enable COG output even if COG is frozen

Bitfield flags

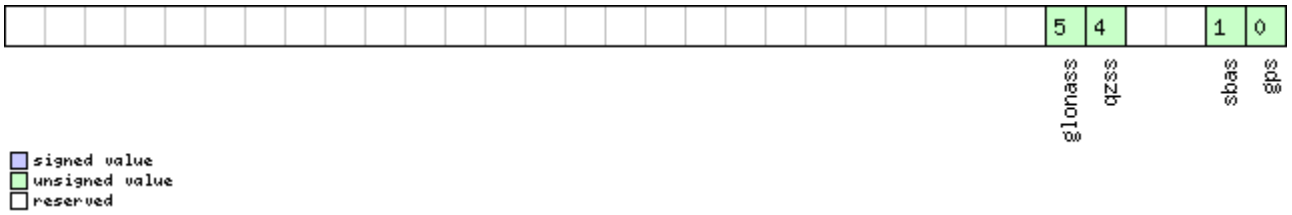
This Graphic explains the bits of `flags`



Name	*Description*
`compat`	enable compatibility mode. This might be needed for certain applications when customer's NMEA parser expects a fixed number of digits in position coordinates
`consider`	enable considering mode.

Bitfield gnssToFilter

This Graphic explains the bits of `gnssToFilter`



Name	Description
`gps`	Disable reporting of GPS satellites
`sbas`	Disable reporting of SBAS satellites
`qzss`	Disable reporting of QZSS satellites
`glonass`	Disable reporting of GLONASS satellites

35.12 CFG-NVS (0x06 0x22)

35.12.1 Clear, Save and Load non-volatile storage data

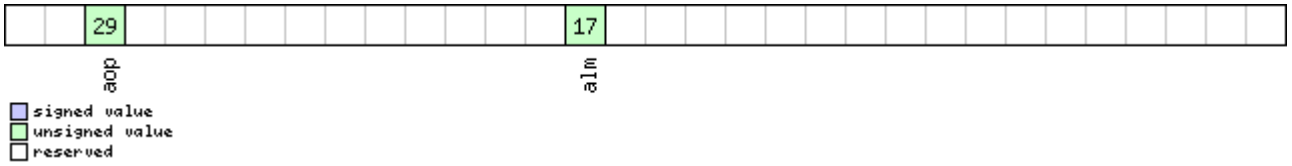
Message	**CFG-NVS**				
Description	**Clear, Save and Load non-volatile storage data**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Command				
Comment	Three masks are made up of individual bits that indicate which data is to be cleared, saved and/or loaded. The fourth mask defines on which devices the corresponding action shall be carried out. Please note that only one command should be flagged at once. Otherwise all commands are processed in the order Clear, Save, and Load. All reserved bits must be set to zero.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x22	13	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	X4	-	`clearMask`	-	Mask of data to be cleared (see graphic below)
4	X4	-	`saveMask`	-	Mask of data to be saved, uses the same bits as the clearMask
8	X4	-	`loadMask`	-	Mask of data to be loaded, uses the same bits as the clearMask
12	X1	-	`deviceMask`	-	Mask of devices to consider (default: all devices) (see graphic below)

Bitfield clearMask

This Graphic explains the bits of `clearMask`



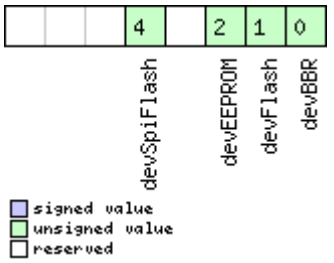
Name	Description
`alm`	GPS Almanac data

Bitfield clearMask Description continued

Name	*Description*
`aop`	AOP data

Bitfield deviceMask

This Graphic explains the bits of `deviceMask`



Name	*Description*
`devBBR`	built-in battery-backed RAM
`devFlash`	external flash memory
`devEEPROM`	external EEPROM
`devSpiFlash`	external SPI Flash

35.13 CFG-PM2 (0x06 0x3B)

35.13.1 Poll extended Power Management configuration

Message	**CFG-PM2**				
Description	**Poll extended Power Management configuration**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x3B	0	*see below*	CK_A CK_B
No payload					

35.13.2 Extended Power Management configuration

Message	**CFG-PM2**				
Description	**Extended Power Management configuration**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x3B	44	*see below*	CK_A CK_B

Payload Contents:

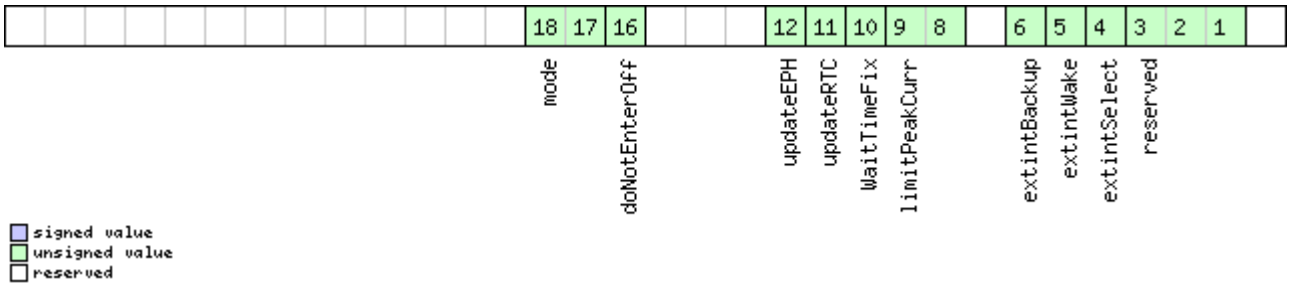
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`version`	-	Message version (1 for this version)
1	U1	-	`reserved1`	-	Reserved
2	U1	-	`reserved2`	-	Reserved

CFG-PM2 continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
3	U1	-	`reserved3`	-	Reserved
4	X4	-	`flags`	-	PSM configuration flags (see graphic below)
8	U4	-	`updatePeriod`	ms	Position update period. If set to 0, the receiver will never retry a fix
12	U4	-	`searchPeriod`	ms	Acquisition retry period. If set to 0, the receiver will never retry a startup
16	U4	-	`gridOffset`	ms	Grid offset relative to GPS start of week
20	U2	-	`onTime`	s	on time after first successful fix
22	U2	-	`minAcqTime`	s	minimal search time
24	U2	-	`reserved4`	-	Reserved
26	U2	-	`reserved5`	-	Reserved
28	U4	-	`reserved6`	-	Reserved
32	U4	-	`reserved7`	-	Reserved
36	U1	-	`reserved8`	-	Reserved
37	U1	-	`reserved9`	-	Reserved
38	U2	-	`reserved10`	-	Reserved
40	U4	-	`reserved11`	-	Reserved

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`reserved`	Reserved: **Must be set to '000'**
`extintSelect`	EXTINT Pin Select 0 EXTINT0 1 EXTINT1
`extintWake`	EXTINT Pin Control 0 disabled 1 enabled, keep receiver awake as long as selected EXTINT pin is 'high'
`extintBackup`	EXTINT Pin Control 0 disabled 1 enabled, force receiver into BACKUP mode when selected EXTINT pin is 'low'
`limitPeakCurr`	Limit Peak Current 00 disabled 01 enabled, peak current is limited 10 reserved 11 reserved

Bitfield flags Description continued

Name	*Description*
`WaitTimeFix`	Wait for Timefix 0 wait for normal Fix ok, before starting on-time 1 wait for time fix ok, before starting on-time
`updateRTC`	Update Real Time Clock 0 Do not wake-up to update RTC. RTC is updated during normal on-time. 1 Update RTC. The receiver adds extra wake-up cycles to update the RTC.
`updateEPH`	Update Ephemeris 0 Do not wake-up to update Ephemeris data 1 Update Ephemeris. The receiver adds extra wake-up cycles to update the Ephemeris data
`doNotEnterOff`	Behavior of receiver in case of no fix 0 receiver enters *inactive for search* state 1 receiver does not enter *inactive for search* state but keeps trying to acquire a fix instead
`mode`	Mode of operation 00 ON/OFF operation 01 Cyclic tracking operation 10 reserved 11 reserved

35.14 CFG-PRT (0x06 0x00)

35.14.1 Polls the configuration of the used I/O Port

Message	**CFG-PRT**				
Description	**Polls the configuration of the used I/O Port**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	Polls the configuration of the I/O Port on which this message is received				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x00	0	*see below*	CK_A CK_B
No payload					

35.14.2 Polls the configuration for one I/O Port

Message	**CFG-PRT**				
Description	**Polls the configuration for one I/O Port**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	Sending this message with a port ID as payload results in having the receiver return the configuration for the specified port.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x00	1	*see below*	CK_A CK_B
Payload Contents:					

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*

CFG-PRT continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`PortID`	-	Port Identifier Number (see the other versions of CFG-PRT for valid values)

35.14.3 Port Configuration for UART

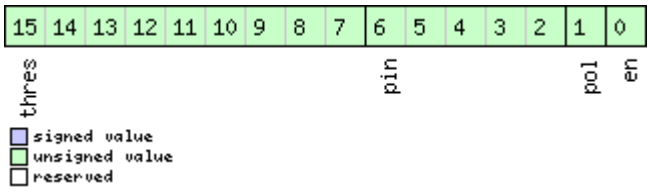
Message	**CFG-PRT**				
Description	**Port Configuration for UART**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	Several configurations can be concatenated to one input message. In this case the payload length can be a multiple of the normal length (see the other versions of CFG-PRT). Output messages from the module contain only one configuration unit.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x00	20	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`portID`	-	Port Identifier Number (see Serial Communication Ports Description for valid UART port IDs)
1	U1	-	`reserved0`	-	Reserved
2	X2	-	`txReady`	-	TX ready PIN configuration (see graphic below)
4	X4	-	`mode`	-	A bit mask describing the UART mode (see graphic below)
8	U4	-	`baudRate`	Bits/s	Baudrate in bits/second
12	X2	-	`inProtoMask`	-	A mask describing which input protocols are active. Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port. (see graphic below)
14	X2	-	`outProtoMask`	-	A mask describing which output protocols are active. Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port. (see graphic below)
16	X2	-	`flags`	-	Flags bit mask (see graphic below)
18	U2	-	`reserved5`	-	Always set to zero

Bitfield txReady

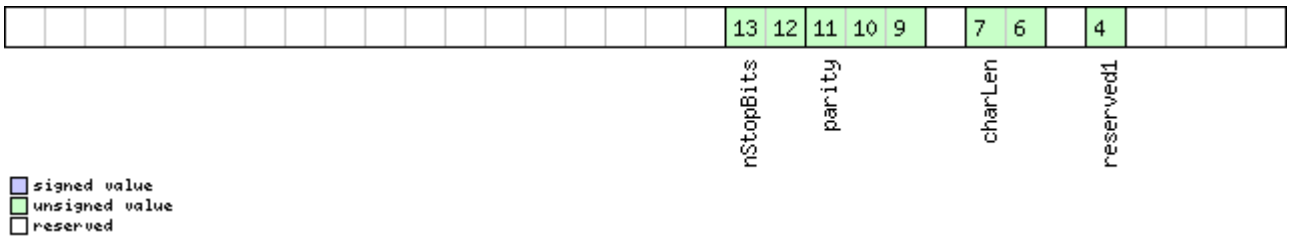
This Graphic explains the bits of `txReady`



Name	*Description*
`en`	Enable TX ready feature for this port
`pol`	Polarity 0 High-active 1 Low-active
`pin`	PIO to be used (must not be in use already by another function)
`thres`	Threshold The given threshold is multiplied by 8 bytes. The TX ready PIN goes active after >= thres*8 bytes are pending for the port and going inactive after the last pending bytes have been written to hardware (0-4 bytes before end of stream). 0x000 no threshold 0x001 8byte 0x002 16byte ... 0x1FE 4080byte 0x1FF 4088byte

Bitfield mode

This Graphic explains the bits of `mode`



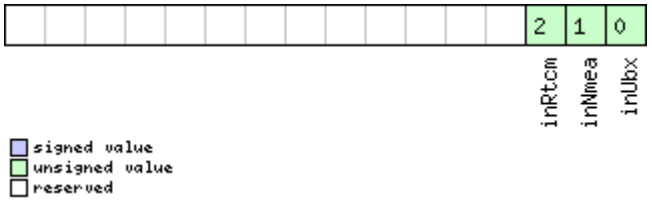
Name	*Description*
`reserved1`	Default 1 for compatibility with A4
`charLen`	Character Length 00 5bit (not supported) 01 6bit (not supported) 10 7bit (supported only with parity) 11 8bit
`parity`	000 Even Parity 001 Odd Parity 10X No Parity X1X Reserved

Bitfield mode Description continued

Name	*Description*
`nStopBits`	Number of Stop Bits 00 1 Stop Bit 01 1.5 Stop Bit 10 2 Stop Bit 11 0.5 Stop Bit

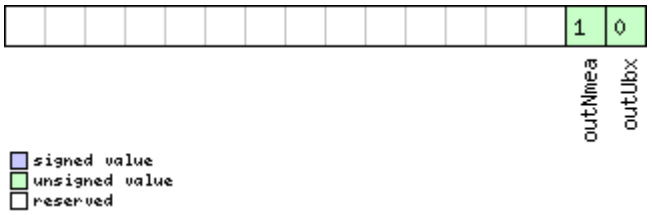
Bitfield inProtoMask

This Graphic explains the bits of `inProtoMask`



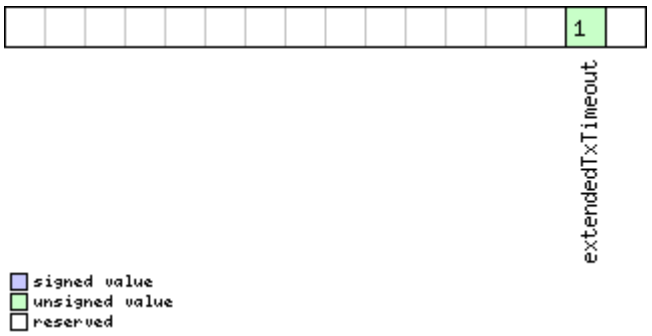
Bitfield outProtoMask

This Graphic explains the bits of `outProtoMask`



Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`extendedTxTimeout`	Extended TX timeout: if set, the port will timeout if allocated TX memory >=4 kB and no activity for 1.5s.

35.14.4 Port Configuration for USB Port

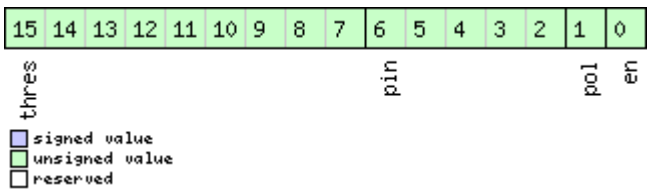
Message	**CFG-PRT**				
Description	**Port Configuration for USB Port**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	Several configurations can be concatenated to one input message. In this case the payload length can be a multiple of the normal length (see the other versions of CFG-PRT). Output messages from the module contain only one configuration unit.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x00	20	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`portID`	-	Port Identifier Number (= 3 for USB port)
1	U1	-	`reserved0`	-	Reserved
2	X2	-	`txReady`	-	TX ready PIN configuration (see graphic below)
4	U4	-	`reserved2`	-	Reserved
8	U4	-	`reserved3`	-	Reserved
12	X2	-	`inProtoMask`	-	A mask describing which input protocols are active. Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port. (see graphic below)
14	X2	-	`outProtoMask`	-	A mask describing which output protocols are active. Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port. (see graphic below)
16	U2	-	`reserved4`	-	Always set to zero
18	U2	-	`reserved5`	-	Always set to zero

Bitfield txReady

This Graphic explains the bits of `txReady`



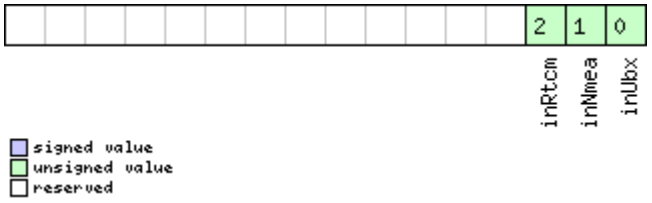
Name	*Description*
`en`	Enable TX ready feature for this port
`pol`	Polarity 0 High-active 1 Low-active
`pin`	PIO to be used (must not be in use already by another function)

Bitfield txReady Description continued

Name	*Description*
`thres`	Threshold The given threshold is multiplied by 8 bytes. The TX ready PIN goes active after >= thres*8 bytes are pending for the port and going inactive after the last pending bytes have been written to hardware (0-4 bytes before end of stream). 0x000 no threshold 0x001 8byte 0x002 16byte ... 0x1FE 4080byte 0x1FF 4088byte

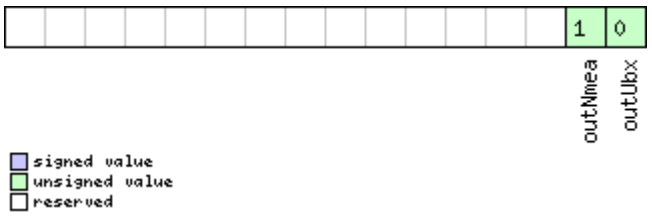
Bitfield inProtoMask

This Graphic explains the bits of `inProtoMask`



Bitfield outProtoMask

This Graphic explains the bits of `outProtoMask`



35.14.5 Port Configuration for SPI Port

Message	**CFG-PRT**				
Description	**Port Configuration for SPI Port**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	Several configurations can be concatenated to one input message. In this case the payload length can be a multiple of the normal length (see the other versions of CFG-PRT). Output messages from the module contain only one configuration unit.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x00	20	*see below*	CK_A CK_B

Payload Contents:

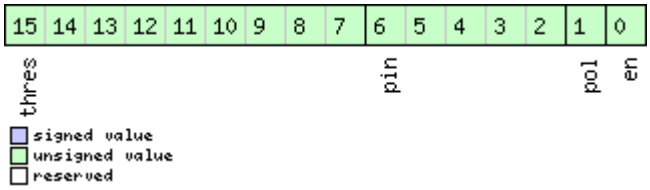
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`portID`	-	Port Identifier Number (= 4 for SPI port)
1	U1	-	`reserved0`	-	Reserved

CFG-PRT continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
2	X2	-	`txReady`	-	TX ready PIN configuration (see graphic below)
4	X4	-	`mode`	-	SPI Mode Flags (see graphic below)
8	U4	-	`reserved3`	-	Reserved
12	X2	-	`inProtoMask`	-	A mask describing which input protocols are active. Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port. (see graphic below)
14	X2	-	`outProtoMask`	-	A mask describing which output protocols are active. Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port. (see graphic below)
16	X2	-	`flags`	-	Flags bit mask (see graphic below)
18	U2	-	`reserved5`	-	Always set to zero

Bitfield txReady

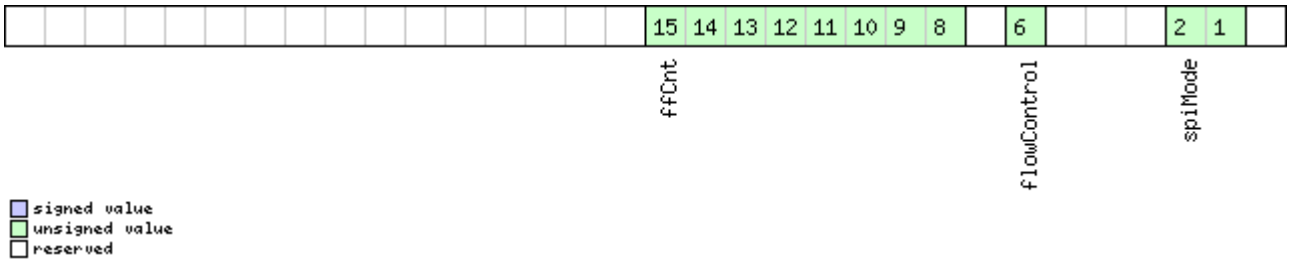
This Graphic explains the bits of `txReady`



Name	*Description*
`en`	Enable TX ready feature for this port
`pol`	Polarity 0 High-active 1 Low-active
`pin`	PIO to be used (must not be in use already by another function)
`thres`	Threshold The given threshold is multiplied by 8 bytes. The TX ready PIN goes active after >= thres*8 bytes are pending for the port and going inactive after the last pending bytes have been written to hardware (0-4 bytes before end of stream). 0x000 no threshold 0x001 8byte 0x002 16byte ... 0x1FE 4080byte 0x1FF 4088byte

Bitfield mode

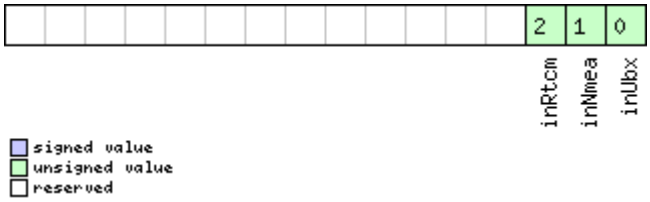
This Graphic explains the bits of `mode`



Name	*Description*
`spiMode`	00 SPI Mode 0: CPOL = 0, CPHA = 0 01 SPI Mode 1: CPOL = 0, CPHA = 1 10 SPI Mode 2: CPOL = 1, CPHA = 0 11 SPI Mode 3: CPOL = 1, CPHA = 1
`flowControl`	(u-blox 6 only) 0 Flow control disabled 1 Flow control enabled (9-bit mode)
`ffCnt`	Number of bytes containing 0xFF to receive before switching off reception. Range: 0(mechanism off)-255

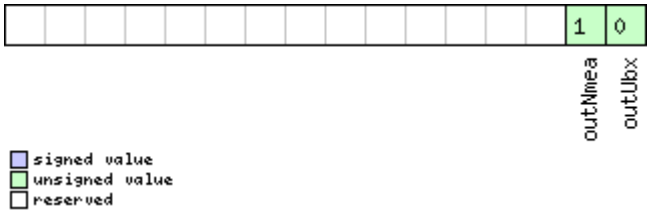
Bitfield inProtoMask

This Graphic explains the bits of `inProtoMask`



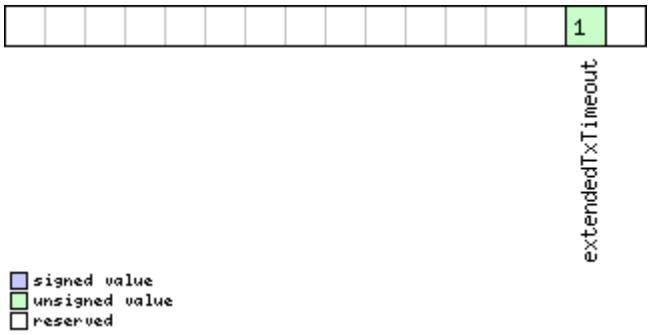
Bitfield outProtoMask

This Graphic explains the bits of `outProtoMask`



Bitfield flags

This Graphic explains the bits of `flags`



Name	Description
`extendedTxTimeout`	Extended TX timeout: if set, the port will timeout if allocated TX memory >=4 kB and no activity for 1.5s.

35.14.6 Port Configuration for DDC Port

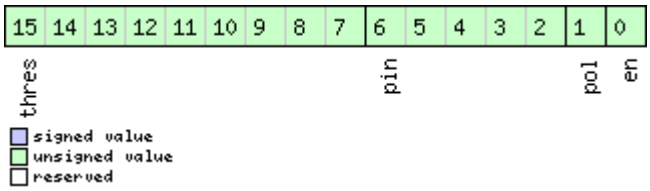
Message	**CFG-PRT**				
Description	**Port Configuration for DDC Port**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	Several configurations can be concatenated to one input message. In this case the payload length can be a multiple of the normal length (see the other versions of CFG-PRT). Output messages from the module contain only one configuration unit.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x00	20	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`portID`	-	Port Identifier Number (= 0 for DDC port)
1	U1	-	`reserved0`	-	Reserved
2	X2	-	`txReady`	-	TX ready PIN configuration (see graphic below)
4	X4	-	`mode`	-	DDC Mode Flags (see graphic below)
8	U4	-	`reserved3`	-	Reserved
12	X2	-	`inProtoMask`	-	A mask describing which input protocols are active. Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port. (see graphic below)
14	X2	-	`outProtoMask`	-	A mask describing which output protocols are active. Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port. (see graphic below)
16	X2	-	`flags`	-	Flags bit mask (see graphic below)
18	U2	-	`reserved5`	-	Always set to zero

Bitfield txReady

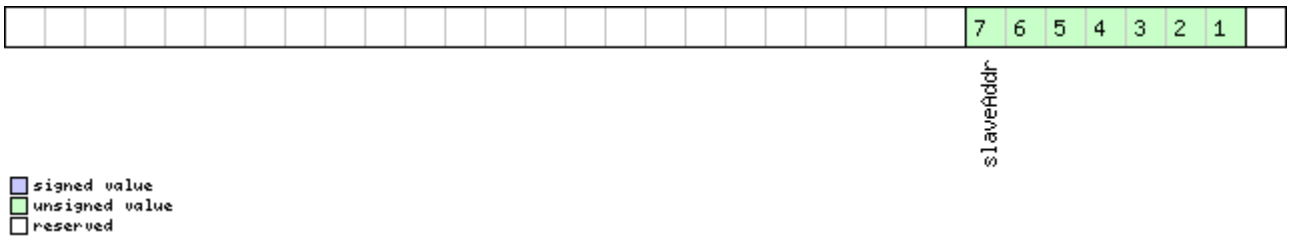
This Graphic explains the bits of `txReady`



Name	*Description*
`en`	Enable TX ready feature for this port
`pol`	Polarity 0 High-active 1 Low-active
`pin`	PIO to be used (must not be in use already by another function)
`thres`	Threshold The given threshold is multiplied by 8 bytes. The TX ready PIN goes active after >= thres*8 bytes are pending for the port and going inactive after the last pending bytes have been written to hardware (0-4 bytes before end of stream). 0x000 no threshold 0x001 8byte 0x002 16byte ... 0x1FE 4080byte 0x1FF 4088byte

Bitfield mode

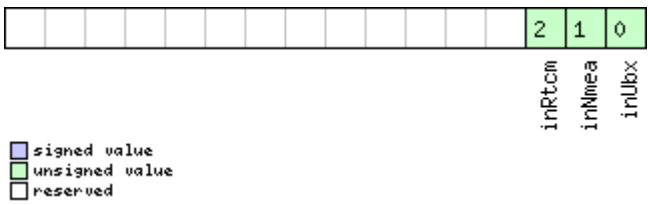
This Graphic explains the bits of `mode`



Name	*Description*
`slaveAddr`	Slave address Range: 0x07 < slaveAddr < 0x78. Bit 0 must be 0

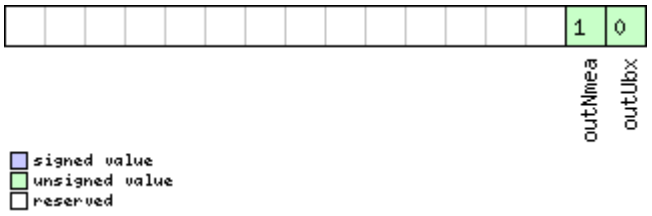
Bitfield inProtoMask

This Graphic explains the bits of `inProtoMask`



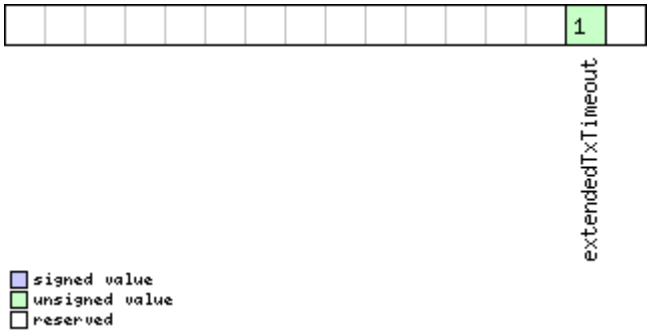
Bitfield outProtoMask

This Graphic explains the bits of `outProtoMask`



Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`extendedTxTimeout`	Extended TX timeout: if set, the port will timeout if allocated TX memory >=4 kB and no activity for 1.5s.

35.15 CFG-RATE (0x06 0x08)

35.15.1 Poll Navigation/Measurement Rate Settings

Message	**CFG-RATE**				
Description	**Poll Navigation/Measurement Rate Settings**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	Sending this (empty / no-payload) message to the receiver results in the receiver returning a message of type CFG-RATE with a payload as defined below				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x08	0	*see below*	CK_A CK_B
No payload					

35.15.2 Navigation/Measurement Rate Settings

Message	**CFG-RATE**				
Description	**Navigation/Measurement Rate Settings**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	The u-blox positioning technology supports navigation update rates higher or lower than 1 update per second. The calculation of the navigation solution will always be aligned to the top of a second. • The update rate has a direct influence on the power consumption. The more fixes that are required, the more CPU power and communication resources are required. • For most applications a 1 Hz update rate would be sufficient. • When using Power Save Mode, measurement and navigation rate can differ from the values configured here. See Measurement and navigation rate with Power Save Mode for details.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x08	6	*see below*	CK_A CK_B
Payload Contents:					

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U2	-	`measRate`	ms	Measurement Rate, GPS measurements are taken every measRate milliseconds
2	U2	-	`navRate`	cycles	Navigation Rate, in number of measurement cycles. This parameter cannot be changed, and must be set to 1.
4	U2	-	`timeRef`	-	Alignment to reference time: 0 = UTC time, 1 = GPS time

35.16 CFG-RINV (0x06 0x34)

35.16.1 Poll contents of Remote Inventory

Message	**CFG-RINV**				
Description	**Poll contents of Remote Inventory**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x34	0	*see below*	CK_A CK_B
No payload					

35.16.2 Contents of Remote Inventory

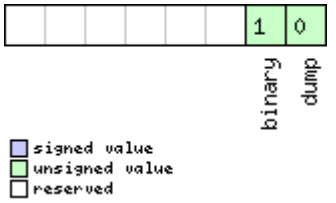
Message	**CFG-RINV**				
Description	**Contents of Remote Inventory**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	If *N* is greater than 30, the excess bytes are discarded. In future firmware versions, this limit may change.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x34	1 + 1*N	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	X1	-	`flags`	-	Flags (see graphic below)
Start of repeated block (N times)					
1 + 1*N	U1	-	`data`	-	Data to store/stored in Remote Inventory
End of repeated block					

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`dump`	Dump data at startup. Does not work if flag `binary` is set.
`binary`	Data is binary

35.17 CFG-RST (0x06 0x04)

35.17.1 Reset Receiver / Clear Backup Data Structures

Message	**CFG-RST**				
Description	**Reset Receiver / Clear Backup Data Structures**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Command				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x04	4	*see below*	CK_A CK_B

Payload Contents:

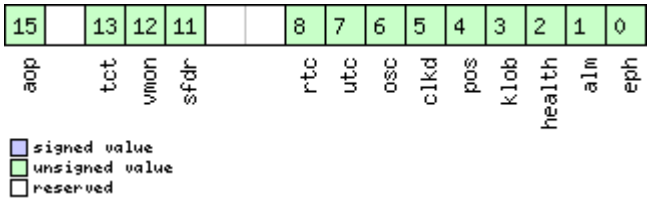
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*

CFG-RST continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	X2	-	`navBbrMask`	-	BBR Sections to clear. The following Special Sets apply: 0x0000 Hotstart 0x0001 Warmstart 0xFFFF Coldstart (see graphic below)
2	U1	-	`resetMode`	-	Reset Type 0x00 - Hardware reset (Watchdog) immediately 0x01 - Controlled Software reset 0x02 - Controlled Software reset (GNSS only) 0x04 - Hardware reset (Watchdog) after shutdown 0x08 - Controlled GNSS stop 0x09 - Controlled GNSS start
3	U1	-	`reserved1`	-	Reserved

Bitfield navBbrMask

This Graphic explains the bits of `navBbrMask`



Name	*Description*
`eph`	Ephemeris
`alm`	Almanac
`health`	Health
`klob`	Klobuchar parameters
`pos`	Position
`clkd`	Clock Drift
`osc`	Oscillator Parameter
`utc`	UTC Correction + GPS Leap Seconds Parameters
`rtc`	RTC
`sfdr`	SFDR Parameters
`vmon`	SFDR Vehicle Monitoring Parameters
`tct`	TCT Parameters
`aop`	Autonomous Orbit Parameters

35.18 CFG-RXM (0x06 0x11)

35.18.1 Poll RXM configuration

Message	**CFG-RXM**				
Description	**Poll RXM configuration**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	Upon sending of this message, the receiver returns CFG-RXM as defined below				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x11	0	*see below*	CK_A CK_B
No payload					

35.18.2 RXM configuration

Message	**CFG-RXM**				
Description	**RXM configuration**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	For a detailed description see section Power Management. Note that Power Save Mode cannot be selected when the receiver is configured to process GLONASS signals (using `CFG-GNSS`).				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x11	2	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`reserved1`	-	Always set to 8
1	U1	-	`lpMode`	-	Low Power Mode 0: Continous Mode 1: Power Save Mode 2-3: reserved 4: Continuous Mode 5-255: reserved Note that for receivers with protocol versions larger or equal 14 both Low Power Mode settings 0 and 4 configure the receiver to Continuous Mode.

35.19 CFG-SBAS (0x06 0x16)

35.19.1 Poll contents of SBAS Configuration

Message	**CFG-SBAS**				
Description	**Poll contents of SBAS Configuration**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x16	0	*see below*	CK_A CK_B
No payload					

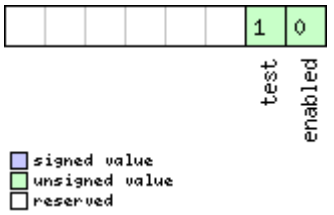
35.19.2 SBAS Configuration

Message	**CFG-SBAS**				
Description	**SBAS Configuration**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	This message configures the SBAS receiver subsystem (i.e. WAAS, EGNOS, MSAS). See the SBAS Configuration Settings Description for a detailed description of how these settings affect receiver operation.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x16	8	*see below*	CK_A CK_B
Payload Contents:					

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	X1	-	`mode`	-	SBAS Mode (see graphic below)
1	X1	-	`usage`	-	SBAS Usage (see graphic below)
2	U1	-	`maxSBAS`	-	Maximum Number of SBAS prioritized tracking channels (valid range: 0 - 3) to use (obsolete and superseeded by UBX-CFG-GNSS in protocol versions 14.00+).
3	X1	-	`scanmode2`	-	Continuation of scanmode bitmask below (see graphic below)
4	X4	-	`scanmode1`	-	Which SBAS PRN numbers to search for (Bitmask) If all Bits are set to zero, auto-scan (i.e. all valid PRNs) are searched. Every bit corresponds to a PRN number (see graphic below)

Bitfield mode

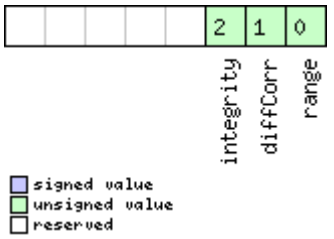
This Graphic explains the bits of `mode`



Name	*Description*
`enabled`	SBAS Enabled (1) / Disabled (0)
`test`	SBAS Testbed: Use data anyhow (1) / Ignore data when in Test Mode (SBAS Msg 0)

Bitfield usage

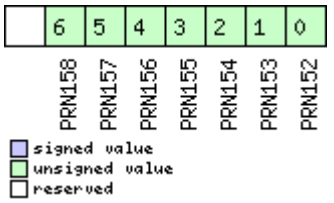
This Graphic explains the bits of `usage`



Name	*Description*
`range`	Use SBAS GEOs as a ranging source (for navigation)
`diffCorr`	Use SBAS Differential Corrections
`integrity`	Use SBAS Integrity Information

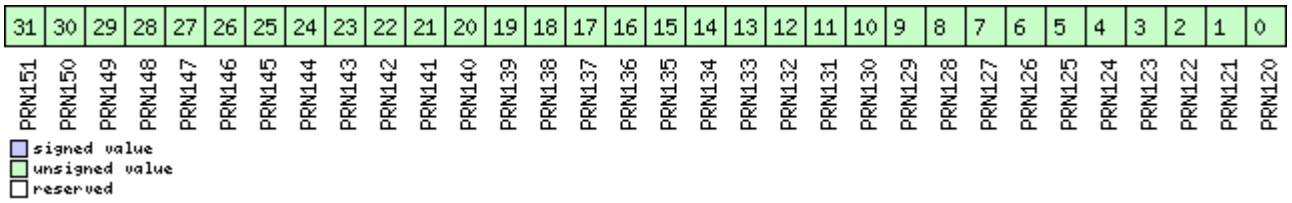
Bitfield scanmode2

This Graphic explains the bits of `scanmode2`



Bitfield scanmode1

This Graphic explains the bits of `scanmode1`



35.20 CFG-TP5 (0x06 0x31)

35.20.1 Poll Time Pulse Parameters

Message	**CFG-TP5**				
Description	**Poll Time Pulse Parameters**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	Sending this (empty / no-payload) message to the receiver results in the receiver returning a message of type `CFG-TP5` with a payload as defined below for timepulse 0.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x31	0	*see below*	CK_A CK_B
No payload					

35.20.2 Poll Time Pulse Parameters

Message	**CFG-TP5**				
Description	**Poll Time Pulse Parameters**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	Sending this message to the receiver results in the receiver returning a message of type `CFG-TP5` with a payload as defined below for the specified time pulse.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x31	1	*see below*	CK_A CK_B
Payload Contents:					

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`tpIdx`	-	Time pulse selection (0 = TIMEPULSE, 1 = TIMEPULSE2)

35.20.3 Time Pulse Parameters

Message	**CFG-TP5**				
Description	**Time Pulse Parameters**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	This message is used to get/set time pulse parameters. For more information see section Time pulse.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x31	32	*see below*	CK_A CK_B
Payload Contents:					

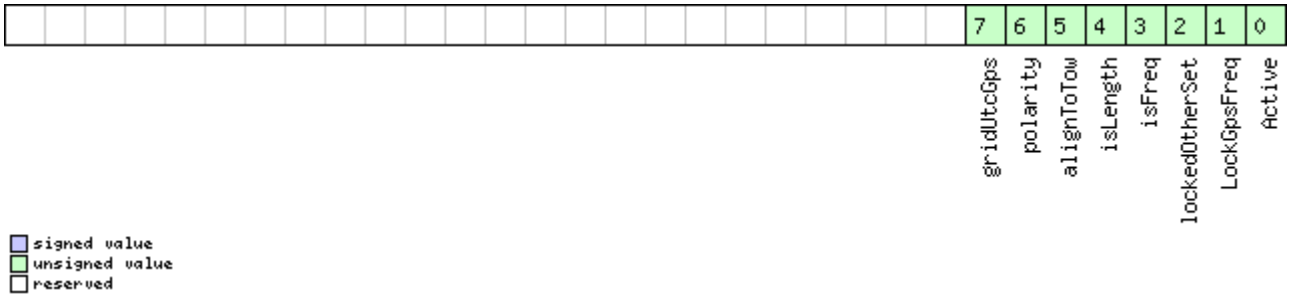
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`tpIdx`	-	Time pulse selection (0 = TIMEPULSE, 1 = TIMEPULSE2)
1	U1	-	`reserved0`	-	Reserved

CFG-TP5 continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
2	U2	-	`reserved1`	-	Reserved
4	I2	-	`antCableDelay`	ns	Antenna cable delay
6	I2	-	`rfGroupDelay`	ns	RF group delay
8	U4	-	`freqPeriod`	Hz_or_us	Frequency or period time, depending on setting of bit 'isFreq'
12	U4	-	`freqPeriodLock`	Hz_or_us	Frequency or period time when locked to GPS time, only used if 'lockedOtherSet' is set
16	U4	-	`pulseLenRatio`	us_or_2^-32	Pulse length or duty cycle, depending on 'isLength'
20	U4	-	`pulseLenRatioLock`	us_or_2^-32	Pulse length or duty cycle when locked to GPS time, only used if 'lockedOtherSet' is set
24	I4	-	`userConfigDelay`	ns	User configurable time pulse delay
28	X4	-	`flags`	-	Configuration flags (see graphic below)

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`Active`	if set enable time pulse; if pin assigned to another function, other function takes precedence
`LockGpsFreq`	if set synchronize time pulse to GPS as soon as GPS time is valid, otherwise use local clock
`lockedOtherSet`	if set use 'freqPeriodLock' and 'pulseLenRatioLock' as soon as GPS time is valid and 'freqPeriod' and 'pulseLenRatio' if GPS time is invalid, if flag is cleared 'freqPeriod' and 'pulseLenRatio' used regardless of GPS time
`isFreq`	if set 'freqPeriodLock' and 'freqPeriod' interpreted as frequency, otherwise interpreted as period
`isLength`	if set 'pulseLenRatioLock' and 'pulseLenRatio' interpreted as pulse length, otherwise interpreted as duty cycle
`alignToTow`	align pulse to top of second (period time must be integer fraction of 1s)
`polarity`	pulse polarity: 0 = falling edge at top of second 1 = rising edge at top of second
`gridUtcGps`	timegrid to use: 0 = UTC 1 = GPS

35.21 CFG-USB (0x06 0x1B)

35.21.1 Poll a USB configuration

Message	**CFG-USB**				
Description	**Poll a USB configuration**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x1B	0	*see below*	CK_A CK_B
No payload					

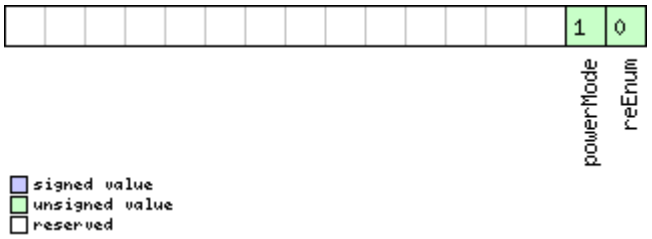
35.21.2 USB Configuration

Message	**CFG-USB**				
Description	**USB Configuration**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input/Output				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x1B	108	*see below*	CK_A CK_B
Payload Contents:					

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U2	-	`vendorID`	-	Vendor ID. This field shall only be set to registered Vendor IDs. Changing this field requires special Host drivers.
2	U2	-	`productID`	-	Product ID. Changing this field requires special Host drivers.
4	U2	-	`reserved1`	-	Always set to zero
6	U2	-	`reserved2`	-	Always set to 1
8	U2	-	`powerConsumption`	mA	Power consumed by the device
10	X2	-	`flags`	-	various configuration flags (see graphic below)
12	CH[32]	-	`vendorString`	-	String containing the vendor name. 32 ASCII bytes including 0-termination.
44	CH[32]	-	`productString`	-	String containing the product name. 32 ASCII bytes including 0-termination.
76	CH[32]	-	`serialNumber`	-	String containing the serial number. 32 ASCII bytes including 0-termination. Changing the String fields requires special Host drivers.

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`reEnum`	force re-enumeration
`powerMode`	self-powered (1), bus-powered (0)

36 INF (0x04)

Information Messages: i.e. Printf-Style Messages, with IDs such as Error, Warning, Notice.
The INF Class is basically an output class that allows the firmware and application code to output strings with a printf-style call. All INF messages have an associated type to indicate the kind of message.

36.1 INF-DEBUG (0x04 0x04)

36.1.1 ASCII String output, indicating debug output

Message	**INF-DEBUG**				
Description	**ASCII String output, indicating debug output**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Output				
Comment	This message has a variable length payload, representing an ASCII string.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x04 0x04	0 + 1*N	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
Start of repeated block (N times)					
N*1	CH	-	`str`	-	ASCII Character
End of repeated block					

36.2 INF-ERROR (0x04 0x00)

36.2.1 ASCII String output, indicating an error

Message	**INF-ERROR**				
Description	**ASCII String output, indicating an error**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Output				
Comment	This message has a variable length payload, representing an ASCII string.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x04 0x00	0 + 1*N	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
Start of repeated block (N times)					
N*1	CH	-	`str`	-	ASCII Character
End of repeated block					

36.3 INF-NOTICE (0x04 0x02)

36.3.1 ASCII String output, with informational contents

Message	**INF-NOTICE**				
Description	**ASCII String output, with informational contents**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Output				
Comment	This message has a variable length payload, representing an ASCII string.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x04 0x02	0 + 1*N	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
Start of repeated block (N times)					
N*1	CH	-	`str`	-	ASCII Character
End of repeated block					

36.4 INF-TEST (0x04 0x03)

36.4.1 ASCII String output, indicating test output

Message	**INF-TEST**				
Description	**ASCII String output, indicating test output**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Output				
Comment	This message has a variable length payload, representing an ASCII string.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x04 0x03	0 + 1*N	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
Start of repeated block (N times)					
N*1	CH	-	`str`	-	ASCII Character
End of repeated block					

36.5 INF-WARNING (0x04 0x01)

36.5.1 ASCII String output, indicating a warning

Message	**INF-WARNING**				
Description	**ASCII String output, indicating a warning**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Output				
Comment	This message has a variable length payload, representing an ASCII string.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x04 0x01	0 + 1*N	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
Start of repeated block (N times)					
N*1	CH	-	`str`	-	ASCII Character
End of repeated block					

37 LOG (0x21)

Logging Messages: i.e. Log creation, deletion, info and retrieval.
The logging feature allows position fixes and arbitrary byte strings to be logged in flash memory attached to the receiver. For a full description of this feature see Logging.

37.1 LOG-CREATE (0x21 0x07)

37.1.1 Create Log File

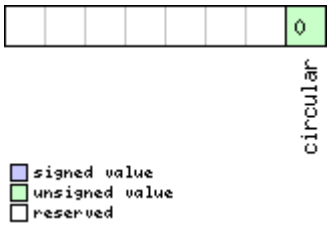
Message	**LOG-CREATE**				
Description	**Create Log File**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Command				
Comment	This message is used to create an initial logging file and activate the logging subsystem. `UBX-ACK-ACK` or `UBX-ACK-NAK` are returned to indicate success or failure. This message does not handle activation of recording or filtering of log entries (see `UBX-CFG-LOGFILTER`).				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x21 0x07	8	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`version`	-	The version of this message. Set to 0
1	X1	-	`logCfg`	-	Config flags (see graphic below)
2	U1	-	`reserved`	-	Reserved. Set to zero
3	U1	-	`logSize`	-	Indicates the size of the log: 0 (maximum safe size): Ensures that logging will not be interupted and enough space will be left avaiable for all other uses of the filestore 1 (minimum size): 2 (user defined): See 'userDefinedSize' below
4	U4	-	`userDefinedSize`	bytes	Sets the maximum amount of space in the filestore that can be used by the logging task. This field is only applicable if logSize is set to user defined.

Bitfield logCfg

This Graphic explains the bits of `logCfg`



Name	*Description*
`circular`	Log is circular (new entries overwrite old ones in a full log) if this bit set

37.2 LOG-ERASE (0x21 0x03)

37.2.1 Erase Logged Data

Message	**LOG-ERASE**				
Description	**Erase Logged Data**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Command				
Comment	This message deactivates the logging system and erases all logged data. `UBX-ACK-ACK` or `UBX-ACK-NAK` are returned to indicate success or failure.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x21 0x03	0	*see below*	CK_A CK_B
No payload					

37.3 LOG-FINDTIME (0x21 0x0E)

37.3.1 Finds the index of the first log entry <= given time

Message	**LOG-FINDTIME**				
Description	**Finds the index of the first log entry <= given time**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Input				
Comment	This message can be used to search a log for the index of the first entry less than or equal to the given time. This index can then be used with the `UBX-LOG-RETRIEVE` message to provide time-based retrieval of log entries.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x21 0x0E	12	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`version`	-	Message version (=0 for this version)
1	U1	-	`type`	-	Message type, 0 for request
2	U2	-	`reserved1`	-	Reserved
4	U2	-	`year`	-	Year (1-65635) of UTC time
6	U1	-	`month`	-	Month (1-12) of UTC time
7	U1	-	`day`	-	Day (1-31) of UTC time
8	U1	-	`hour`	-	Hour (0-23) of UTC time
9	U1	-	`minute`	-	Minute (0-59) of UTC time
10	U1	-	`second`	-	Second (0-60) of UTC time
11	U1	-	`reserved2`	-	Reserved

37.3.2 This message is the response to FINDTIME request.

Message	**LOG-FINDTIME**				
Description	**This message is the response to FINDTIME request.**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Output				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x21 0x0E	8	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`version`	-	Message version (=1 for this version)
1	U1	-	`type`	-	Message type, 1 for response
2	U2	-	`reserved1`	-	Reserved
4	U4	-	`entryNumber`	-	Index of the most recent entry with time <= specified

37.4 LOG-INFO (0x21 0x08)

37.4.1 Poll for log information

Message	**LOG-INFO**				
Description	**Poll for log information**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	Upon sending of this message, the receiver returns UBX-LOG-INFO as defined below.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x21 0x08	0	*see below*	CK_A CK_B
No payload					

37.4.2 Log information

Message	**LOG-INFO**
Description	**Log information**
Firmware	Supported on: • u-blox 7 firmware version 1.00
Type	Output
Comment	This message is used to report information about the logging subsystem. Note: • The reported maximum log size will be smaller than that originally specified in LOG-CREATE due to logging and filestore implementation overheads. • Log entries are compressed in a variable length fashion, so it may be difficult to predict log space usage with any precision. • There may be times when the receiver does not have an accurate time (e.g. if the week number is not yet known), in which case some entries will not have a timestamp - this may result in the oldest/newest entry time values not taking account of these entries.

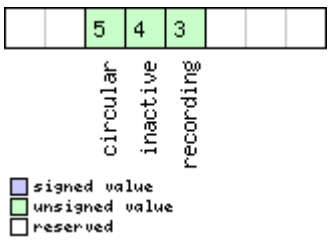
Message Structure	Header	ID	Length (Bytes)	Payload	Checksum
	0xB5 0x62	0x21 0x08	48	see below	CK_A CK_B

Payload Contents:

Byte Offset	Number Format	Scaling	Name	Unit	Description
0	U1	-	`version`	-	The version of this message. Set to 1
1	U1[3]	-	`reserved1`	-	Reserved
4	U4	-	`filestoreCapacity`	bytes	The capacity of the filestore
8	U4	-	`reserved2`	-	Reserved
12	U4	-	`reserved3`	-	Reserved
16	U4	-	`currentMaxLogSize`	bytes	The maximum size the current log is allowed to grow to
20	U4	-	`currentLogSize`	bytes	Approximate amount of space in log currently occupied
24	U4	-	`entryCount`	-	Number of entries in the log. Note: for circular logs this value will decrease when a group of entries is deleted to make space for new ones.
28	U2	-	`oldestYear`	-	Oldest entry UTC year year (1-65635) or zero if there are no entries with known time
30	U1	-	`oldestMonth`	-	Oldest month (1-12)
31	U1	-	`oldestDay`	-	Oldest day (1-31)
32	U1	-	`oldestHour`	-	Oldest hour (0-23)
33	U1	-	`oldestMinute`	-	Oldest minute (0-59)
34	U1	-	`oldestSecond`	-	Oldest second (0-60)
35	U1	-	`reserved4`	-	Reserved.
36	U2	-	`newestYear`	-	Newest year (1-65635) or zero if there are no entries with known time
38	U1	-	`newestMonth`	-	Newest month (1-12)
39	U1	-	`newestDay`	-	Newest day (1-31)
40	U1	-	`newestHour`	-	Newest hour (0-23)
41	U1	-	`newestMinute`	-	Newest minute (0-59)
42	U1	-	`newestSecond`	-	Newest second (0-60)
43	U1	-	`reserved5`	-	Reserved.
44	X1	-	`status`	-	Log status flags (see graphic below)
45	U1[3]	-	`reserved6`	-	Reserved

Bitfield status

This Graphic explains the bits of `status`



Name	Description

Bitfield status Description continued

Name	*Description*
`recording`	Log entry recording is currently turned on
`inactive`	Logging system not active - no log present
`circular`	The current log is circular

37.5 LOG-RETRIEVEPOS (0x21 0x0b)

37.5.1 Position fix log entry

Message	**LOG-RETRIEVEPOS**				
Description	**Position fix log entry**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Output				
Comment	This message is used to report a position fix log entry				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x21 0x0b	40	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`entryIndex`	-	The index of this log entry
4	I4	1e-7	`lon`	deg	Longitude
8	I4	1e-7	`lat`	deg	Latitude
12	I4	-	`hMSL`	mm	Height above mean sea level
16	U4	-	`hAcc`	mm	Horizontal accuracy estimate
20	U4	-	`gSpeed`	mm/s	Ground speed (2-D)
24	U4	-	`heading`	deg	Heading
28	U1	-	`version`	-	The version of this message. Set to 0
29	U1	-	`fixType`	-	Fix type: 2: 2D-Fix 3: 3D-Fix
30	U2	-	`year`	-	Year (1-65635) of UTC time
32	U1	-	`month`	-	Month (1-12) of UTC time
33	U1	-	`day`	-	Day (1-31) of UTC time
34	U1	-	`hour`	-	Hour (0-23) of UTC time
35	U1	-	`minute`	-	Minute (0-59) of UTC time
36	U1	-	`second`	-	Second (0-60) of UTC time
37	U1	-	`reserved1`	-	Reserved
38	U1	-	`numSV`	-	Number of satellites used in the position fix
39	U1	-	`reserved2`	-	Reserved

37.6 LOG-RETRIEVESTRING (0x21 0x0d)

37.6.1 Byte string log entry

Message	**LOG-RETRIEVESTRING**				
Description	**Byte string log entry**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Output				
Comment	This message is used to report a byte string log entry				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x21 0x0d	16 + 1*byteCount	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`entryIndex`	-	The index of this log entry
4	U1	-	`version`	-	The version of this message. Set to 0
5	U1	-	`reserved1`	-	Reserved
6	U2	-	`year`	-	Year (1-65635) of UTC time. Will be zero if time not known
8	U1	-	`month`	-	Month (1-12) of UTC time
9	U1	-	`day`	-	Day (1-31) of UTC time
10	U1	-	`hour`	-	Hour (0-23) of UTC time
11	U1	-	`minute`	-	Minute (0-59) of UTC time
12	U1	-	`second`	-	Second (0-60) of UTC time
13	U1	-	`reserved2`	-	Reserved
14	U2	-	`byteCount`	-	Size of string in bytes
Start of repeated block (byteCount times)					
16 + 1*N	U1	-	`bytes`	-	The bytes of the string
End of repeated block					

37.7 LOG-RETRIEVE (0x21 0x09)

37.7.1 Request log data

Message	**LOG-RETRIEVE**
Description	**Request log data**
Firmware	Supported on: • u-blox 7 firmware version 1.00
Type	Command
Comment	This message is used to request logged data (log recording must first be disabled, see `UBX-CFG-LOGFILTER`). Log entries are returned in chronological order, using the messages `UBX-LOG-RETRIEVEPOS` and `UBX-LOG-RETRIEVESTRING`. The maximum number of entries that can be returned in response to a single UBX-LOG-RETRIEVE message is 256. If more entries than this are required the mesage will need to be sent multiple times with different startNumbers. The retrieve will be stopped if any UBX-LOG message is received. The speed of transfer can be maximised by using a high data rate and temporarily stopping the GPS processing (see `UBX-CFG-RST`)

	Header	ID	Length (Bytes)	Payload	Checksum
Message Structure	0xB5 0x62	0x21 0x09	12	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	Number Format	Scaling	Name	Unit	Description
0	U4	-	`startNumber`	-	Index of first entry to be transferred
4	U4	-	`entryCount`	-	Number of log entries to transfer. The maximum is 256
8	U1	-	`version`	-	The version of this message. Set to 0
9	U1[3]	-	`reserved`	-	Reserved

37.8 LOG-STRING (0x21 0x04)

37.8.1 Store arbitrary string in on-board Flash memory

Message	**LOG-STRING**
Description	**Store arbitrary string in on-board Flash memory**
Firmware	Supported on: • u-blox 7 firmware version 1.00
Type	Command
Comment	This message can be used to store an arbitrary byte string in the on-board flash memory. The maximum length that can be stored is 256 bytes.

	Header	ID	Length (Bytes)	Payload	Checksum
Message Structure	0xB5 0x62	0x21 0x04	0 + 1*N	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	Number Format	Scaling	Name	Unit	Description
Start of repeated block (N times)					
N*1	U1	-	`bytes`	-	The string of bytes to be logged (maximum 256)
End of repeated block					

38 MON (0x0A)

Monitoring Messages: i.e. Comunication Status, CPU Load, Stack Usage, Task Status.
Messages in this class are sent to report GPS receiver status, such as CPU load, stack usage, I/O subsystem statistics etc.

38.1 MON-HW2 (0x0A 0x0B)

38.1.1 Extended Hardware Status

Message	**MON-HW2**				
Description	**Extended Hardware Status**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	Status of different aspects of the hardware such as Imbalance, Low-Level Configuration and POST Results. The first four parameters of this message represent the complex signal from the RF front end. The following rules of thumb apply: • The smaller the absolute value of the variable `ofsI` and `ofsQ` respectively, the better. • Ideally, the magnitude of the I-part (`magI`) and the Q-part (`magQ`) of the complex signal should be the same.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0A 0x0B	28	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	I1	-	`ofsI`	-	Imbalance of I-part of complex signal, scaled (-128 = max. negative imbalance, 127 = max. positive imbalance)
1	U1	-	`magI`	-	Magnitude of I-part of complex signal, scaled (0 = no signal, 255 = max. magnitude)
2	I1	-	`ofsQ`	-	Imbalance of Q-part of complex signal, scaled (-128 = max. negative imbalance, 127 = max. positive imbalance)
3	U1	-	`magQ`	-	Magnitude of Q-part of complex signal, scaled (0 = no signal, 255 = max. magnitude)
4	U1	-	`cfgSource`	-	Source of low-level configuration (114 = ROM, 111 = OTP, 112 = config pins, 102 = flash image)
5	U1[3]	-	`reserved0`	-	Reserved
8	U4	-	`lowLevCfg`	-	Low-level configuration
12	U4[2]	-	`reserved1`	-	Reserved
20	U4	-	`postStatus`	-	POST status word
24	U4	-	`reserved2`	-	Reserved

38.2 MON-HW (0x0A 0x09)

38.2.1 Hardware Status

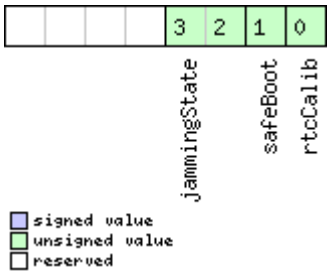
Message	**MON-HW**				
Description	**Hardware Status**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	Status of different aspect of the hardware, such as Antenna, PIO/Peripheral Pins, Noise Level, Automatic Gain Control (AGC)				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0A 0x09	60	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	X4	-	`pinSel`	-	Mask of Pins Set as Peripheral/PIO
4	X4	-	`pinBank`	-	Mask of Pins Set as Bank A/B
8	X4	-	`pinDir`	-	Mask of Pins Set as Input/Output
12	X4	-	`pinVal`	-	Mask of Pins Value Low/High
16	U2	-	`noisePerMS`	-	Noise Level as measured by the GPS Core
18	U2	-	`agcCnt`	-	AGC Monitor (counts SIGHI xor SIGLO, range 0 to 8191)
20	U1	-	`aStatus`	-	Status of the Antenna Supervisor State Machine (0=INIT, 1=DONTKNOW, 2=OK, 3=SHORT, 4=OPEN)
21	U1	-	`aPower`	-	Current PowerStatus of Antenna (0=OFF, 1=ON, 2=DONTKNOW)
22	X1	-	`flags`	-	Flags (see graphic below)
23	U1	-	`reserved1`	-	Reserved
24	X4	-	`usedMask`	-	Mask of Pins that are used by the Virtual Pin Manager
28	U1[17]	-	`VP`	-	Array of Pin Mappings for each of the 17 Physical Pins
45	U1	-	`jamInd`	-	CW Jamming indicator, scaled (0 = no CW jamming, 255 = strong CW jamming)
46	U2	-	`reserved3`	-	Reserved
48	X4	-	`pinIrq`	-	Mask of Pins Value using the PIO Irq
52	X4	-	`pullH`	-	Mask of Pins Value using the PIO Pull High Resistor
56	X4	-	`pullL`	-	Mask of Pins Value using the PIO Pull Low Resistor

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`rtcCalib`	RTC is calibrated
`safeBoot`	safeBoot mode (0 = inactive, 1 = active)
`jammingState`	output from Jamming/Interference Monitor (0 = unknown or feature disabled, 1 = ok - no significant jamming, 2 = warning - interference visible but fix OK, 3 = critical - interference visible and no fix)

38.3 MON-IO (0x0A 0x02)

38.3.1 I/O Subsystem Status

Message	**MON-IO**				
Description	**I/O Subsystem Status**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	The size of the message is determined by the number of ports 'N' the receiver supports, i.e. on u-blox 5 the number of ports is 6.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0A 0x02	0 + 20*N	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
Start of repeated block (N times)					
N*20	U4	-	`rxBytes`	bytes	Number of bytes ever received
4 + 20*N	U4	-	`txBytes`	bytes	Number of bytes ever sent
8 + 20*N	U2	-	`parityErrs`	-	Number of 100ms timeslots with parity errors
10 + 20*N	U2	-	`framingErrs`	-	Number of 100ms timeslots with framing errors
12 + 20*N	U2	-	`overrunErrs`	-	Number of 100ms timeslots with overrun errors
14 + 20*N	U2	-	`breakCond`	-	Number of 100ms timeslots with break conditions
16 + 20*N	U1	-	`rxBusy`	-	Flag is receiver is busy
17 + 20*N	U1	-	`txBusy`	-	Flag is transmitter is busy
18 + 20*N	U2	-	`reserved1`	-	Reserved
End of repeated block					

38.4 MON-MSGPP (0x0A 0x06)

38.4.1 Message Parse and Process Status

Message	**MON-MSGPP**				
Description	**Message Parse and Process Status**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0A 0x06	120	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U2[8]	-	`msg1`	msgs	Number of successfully parsed messages for each protocol on port0
16	U2[8]	-	`msg2`	msgs	Number of successfully parsed messages for each protocol on port1
32	U2[8]	-	`msg3`	msgs	Number of successfully parsed messages for each protocol on port2
48	U2[8]	-	`msg4`	msgs	Number of successfully parsed messages for each protocol on port3
64	U2[8]	-	`msg5`	msgs	Number of successfully parsed messages for each protocol on port4
80	U2[8]	-	`msg6`	msgs	Number of successfully parsed messages for each protocol on port5
96	U4[6]	-	`skipped`	bytes	Number skipped bytes for each port

38.5 MON-RXBUF (0x0A 0x07)

38.5.1 Receiver Buffer Status

Message	**MON-RXBUF**				
Description	**Receiver Buffer Status**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0A 0x07	24	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U2[6]	-	`pending`	bytes	Number of bytes pending in receiver buffer for each target
12	U1[6]	-	`usage`	%	Maximum usage receiver buffer during the last sysmon period for each target
18	U1[6]	-	`peakUsage`	%	Maximum usage receiver buffer for each target

38.6 MON-RXR (0x0A 0x21)

38.6.1 Receiver Status Information

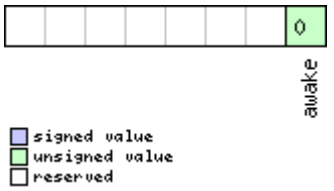
Message	**MON-RXR**				
Description	**Receiver Status Information**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Output				
Comment	The receiver ready message is sent when the receiver changes from or to backup mode.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0A 0x21	1	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	X1	-	`flags`	-	Receiver status flags (see graphic below)

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`awake`	not in Backup mode

38.7 MON-TXBUF (0x0A 0x08)

38.7.1 Transmitter Buffer Status

Message	**MON-TXBUF**				
Description	**Transmitter Buffer Status**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0A 0x08	28	*see below*	CK_A CK_B

Payload Contents:

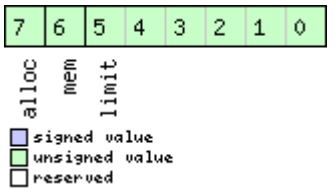
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U2[6]	-	`pending`	bytes	Number of bytes pending in transmitter buffer for each target
12	U1[6]	-	`usage`	%	Maximum usage transmitter buffer during the last sysmon period for each target
18	U1[6]	-	`peakUsage`	%	Maximum usage transmitter buffer for each target

MON-TXBUF continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
24	U1	-	`tUsage`	%	Maximum usage of transmitter buffer during the last sysmon period for all targets
25	U1	-	`tPeakusage`	%	Maximum usage of transmitter buffer for all targets
26	X1	-	`errors`	-	Error bitmask (see graphic below)
27	U1	-	`reserved1`	-	Reserved

Bitfield errors

This Graphic explains the bits of `errors`



Name	*Description*
`limit`	Buffer limit of corresponding target reached
`mem`	Memory Allocation error
`alloc`	Allocation error (TX buffer full)

38.8 MON-VER (0x0A 0x04)

38.8.1 Poll Receiver/Software Version

Message	**MON-VER**				
Description	**Poll Receiver/Software Version**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Poll Request				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0A 0x04	0	*see below*	CK_A CK_B
No payload					

38.8.2 Receiver/Software Version

Message	**MON-VER**				
Description	**Receiver/Software Version**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Answer to Poll				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0A 0x04	40 + 30*N	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	CH[30]	-	`swVersion`	-	Zero-terminated Software Version String.
30	CH[10]	-	`hwVersion`	-	Zero-terminated Hardware Version String
Start of repeated block (N times)					
40 + 30*N	CH[30]	-	`extension`	-	Extended receiver/software information. If the receiver's firmware is running from flash, the first extension field will contain the Software Version String of the underlying ROM. Additional fields may also indicate the supported protocol version and any product variants, capabilities or extensions.
End of repeated block					

39 NAV (0x01)

Navigation Results: i.e. Position, Speed, Time, Acc, Heading, DOP, SVs used.
Messages in the NAV Class output Navigation Data such as position, altitude and velocity in a number of formats. Additionally, status flags and accuracy figures are output.

39.1 NAV-AOPSTATUS (0x01 0x60)

39.1.1 AssistNow Autonomous Status

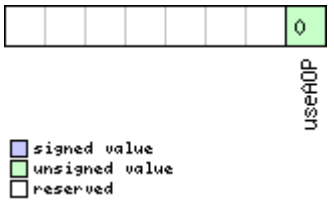
Message	**NAV-AOPSTATUS**				
Description	**AssistNow Autonomous Status**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	This message provides information on the current availability of *AssistNow Autonomous* data and the current state of the subsystem on the receiver. For example, a host application can determine the optimal time to shut down the receiver by monitoring the `status` field for a steady 0. See the chapter AssistNow Autonomous in the receiver description for details on this feature.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x60	20	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	U1	-	`aopCfg`	-	*AssistNow Autonomous* configuration (see graphic below)
5	U1	-	`status`	-	*AssistNow Autonomous* subsystem is idle (0) or running (not 0)
6	U1	-	`reserved0`	-	Always set to zero
7	U1	-	`reserved1`	-	Always set to zero
8	U4	-	`availGPS`	-	data availability mask for GPS SVs (bits 0-31 correspond to GPS PRN 1-32)
12	U4	-	`reserved2`	-	Always set to zero
16	U4	-	`reserved3`	-	Always set to zero

Bitfield aopCfg

This Graphic explains the bits of `aopCfg`



Name	*Description*
`useAOP`	AOP enabled flag

39.2 NAV-CLOCK (0x01 0x22)

39.2.1 Clock Solution

Message	**NAV-CLOCK**				
Description	**Clock Solution**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x22	20	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	I4	-	`clkB`	ns	Clock bias
8	I4	-	`clkD`	ns/s	Clock drift
12	U4	-	`tAcc`	ns	Time accuracy estimate
16	U4	-	`fAcc`	ps/s	Frequency accuracy estimate

39.3 NAV-DGPS (0x01 0x31)

39.3.1 DGPS Data Used for NAV

Message	**NAV-DGPS**				
Description	**DGPS Data Used for NAV**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	This message outputs the DGPS correction data that has been applied to the current NAV Solution. See also the notes on the RTCM protocol.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x31	16 + 12*numCh	*see below*	CK_A CK_B

Payload Contents:

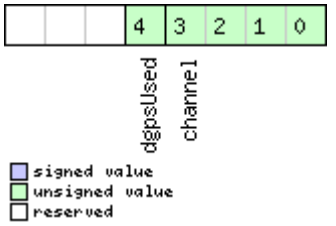
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	I4	-	`age`	ms	Age of newest correction data
8	I2	-	`baseId`	-	DGPS basestation identifier
10	I2	-	`baseHealth`	-	DGPS basestation health status
12	U1	-	`numCh`	-	Number of channels for which correction data is following
13	U1	-	`status`	-	DGPS correction type status: 0x00: none 0x01: PR+PRR correction
14	U2	-	`reserved1`	-	Reserved

NAV-DGPS continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
Start of repeated block (numCh times)					
16 + 12*N	U1	-	`svid`	-	Satellite ID
17 + 12*N	X1	-	`flags`	-	Channel number and usage (see graphic below)
18 + 12*N	U2	-	`ageC`	ms	Age of latest correction data
20 + 12*N	R4	-	`prc`	m	Pseudorange correction
24 + 12*N	R4	-	`prrc`	m/s	Pseudorange rate correction
End of repeated block					

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`channel`	GPS channel number this SV is on
`dgpsUsed`	1 = DGPS used for this SV

39.4 NAV-DOP (0x01 0x04)

39.4.1 Dilution of precision

Message	**NAV-DOP**				
Description	**Dilution of precision**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	• DOP values are dimensionless. • All DOP values are scaled by a factor of 100. If the unit transmits a value of e.g. 156, the DOP value is 1.56.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x04	18	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	U2	0.01	`gDOP`	-	Geometric DOP
6	U2	0.01	`pDOP`	-	Position DOP
8	U2	0.01	`tDOP`	-	Time DOP
10	U2	0.01	`vDOP`	-	Vertical DOP
12	U2	0.01	`hDOP`	-	Horizontal DOP
14	U2	0.01	`nDOP`	-	Northing DOP

NAV-DOP continued

Byte Offset	Number Format	Scaling	Name	Unit	Description
16	U2	0.01	eDOP	-	Easting DOP

39.5 NAV-POSECEF (0x01 0x01)

39.5.1 Position Solution in ECEF

Message	**NAV-POSECEF**				
Description	**Position Solution in ECEF**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	**See important comments concerning validity of position given in section Navigation Output Filters.** -				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x01	20	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	Number Format	Scaling	Name	Unit	Description
0	U4	-	iTOW	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	I4	-	ecefX	cm	ECEF X coordinate
8	I4	-	ecefY	cm	ECEF Y coordinate
12	I4	-	ecefZ	cm	ECEF Z coordinate
16	U4	-	pAcc	cm	Position Accuracy Estimate

39.6 NAV-POSLLH (0x01 0x02)

39.6.1 Geodetic Position Solution

Message	**NAV-POSLLH**				
Description	**Geodetic Position Solution**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	**See important comments concerning validity of position given in section Navigation Output Filters.** This message outputs the Geodetic position in the currently selected ellipsoid. The default is the WGS84 Ellipsoid, but can be changed with the message CFG-DAT.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x02	28	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	Number Format	Scaling	Name	Unit	Description
0	U4	-	iTOW	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	I4	1e-7	lon	deg	Longitude

NAV-POSLLH continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
8	I4	1e-7	`lat`	deg	Latitude
12	I4	-	`height`	mm	Height above ellipsoid
16	I4	-	`hMSL`	mm	Height above mean sea level
20	U4	-	`hAcc`	mm	Horizontal accuracy estimate
24	U4	-	`vAcc`	mm	Vertical accuracy estimate

39.7 NAV-PVT (0x01 0x07)

39.7.1 Navigation Position Velocity Time Solution

Message	**NAV-PVT**				
Description	**Navigation Position Velocity Time Solution**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	**Note that during a leap second there may be more (or less) than 60 seconds in a minute; see the description of leap seconds for details.** This message combines position, velocity and time solution, including accuracy figures				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x07	84	*see below*	CK_A CK_B

Payload Contents:

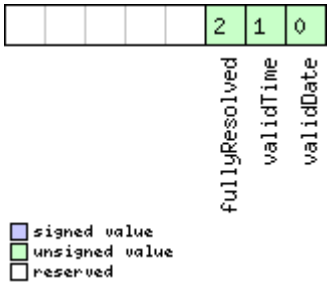
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	U2	-	`year`	y	Year (UTC)
6	U1	-	`month`	month	Month, range 1..12 (UTC)
7	U1	-	`day`	d	Day of month, range 1..31 (UTC)
8	U1	-	`hour`	h	Hour of day, range 0..23 (UTC)
9	U1	-	`min`	min	Minute of hour, range 0..59 (UTC)
10	U1	-	`sec`	s	Seconds of minute, range 0..60 (UTC)
11	X1	-	`valid`	-	Validity Flags (see graphic below)
12	U4	-	`tAcc`	ns	Time accuracy estimate (UTC)
16	I4	-	`nano`	ns	Fraction of second, range -1e9 .. 1e9 (UTC)
20	U1	-	`fixType`	-	GNSSfix Type, range 0..5 0x00 = No Fix 0x01 = Dead Reckoning only 0x02 = 2D-Fix 0x03 = 3D-Fix 0x04 = GNSS + dead reckoning combined 0x05 = Time only fix 0x06..0xff: reserved
21	X1	-	`flags`	-	Fix Status Flags (see graphic below)
22	U1	-	`reserved1`	-	Reserved
23	U1	-	`numSV`	-	Number of satellites used in Nav Solution
24	I4	1e-7	`lon`	deg	Longitude

NAV-PVT continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
28	I4	1e-7	`lat`	deg	Latitude
32	I4	-	`height`	mm	Height above Ellipsoid
36	I4	-	`hMSL`	mm	Height above mean sea level
40	U4	-	`hAcc`	mm	Horizontal Accuracy Estimate
44	U4	-	`vAcc`	mm	Vertical Accuracy Estimate
48	I4	-	`velN`	mm/s	NED north velocity
52	I4	-	`velE`	mm/s	NED east velocity
56	I4	-	`velD`	mm/s	NED down velocity
60	I4	-	`gSpeed`	mm/s	Ground Speed (2-D)
64	I4	1e-5	`heading`	deg	Heading of motion 2-D
68	U4	-	`sAcc`	mm/s	Speed Accuracy Estimate
72	U4	1e-5	`headingAcc`	deg	Heading Accuracy Estimate
76	U2	0.01	`pDOP`	-	Position DOP
78	X2	-	`reserved2`	-	Reserved
80	U4	-	`reserved3`	-	Reserved

Bitfield valid

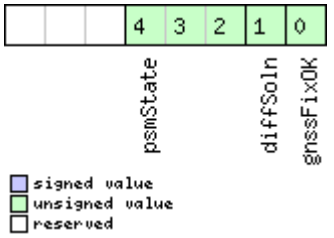
This Graphic explains the bits of `valid`



Name	*Description*
`validDate`	1 = Valid UTC Date
`validTime`	1 = Valid UTC Time of Day
`fullyResolved`	1 = UTC Time of Day has been fully resolved (no seconds uncertainty)

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`gnssFixOK`	A valid fix (i.e within DOP & accuracy masks)
`diffSoln`	1 if differential corrections were applied

Bitfield flags Description continued

Name	*Description*
`psmState`	Power Save Mode state (see Power Management): 0 = n/a (i.e no PSM is active) 1 = ENABLED (an intermediate state before ACQUISITION state 2 = ACQUISITION 3 = TRACKING 4 = POWER OPTIMIZED TRACKING 5 = INACTIVE

39.8 NAV-SBAS (0x01 0x32)

39.8.1 SBAS Status Data

Message	**NAV-SBAS**				
Description	**SBAS Status Data**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	This message outputs the status of the SBAS sub system				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x32	12 + 12*cnt	*see below*	CK_A CK_B

Payload Contents:

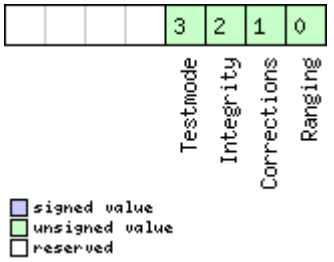
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	U1	-	`geo`	-	PRN Number of the GEO where correction and integrity data is used from
5	U1	-	`mode`	-	SBAS Mode 0 Disabled 1 Enabled Integrity 3 Enabled Testmode
6	I1	-	`sys`	-	SBAS System (WAAS/EGNOS/...) -1 Unknown 0 WAAS 1 EGNOS 2 MSAS 16 GPS
7	X1	-	`service`	-	SBAS Services available (see graphic below)
8	U1	-	`cnt`	-	Number of SV data following
9	U1[3]	-	`reserved0`	-	Reserved
Start of repeated block (cnt times)					
12 + 12*N	U1	-	`svid`	-	SV Id
13 + 12*N	U1	-	`flags`	-	Flags for this SV
14 + 12*N	U1	-	`udre`	-	Monitoring status
15 + 12*N	U1	-	`svSys`	-	System (WAAS/EGNOS/...) same as SYS

NAV-SBAS continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
16 + 12*N	U1	-	`svService`	-	Services available same as SERVICE
17 + 12*N	U1	-	`reserved1`	-	Reserved
18 + 12*N	I2	-	`prc`	cm	Pseudo Range correction in [cm]
20 + 12*N	U2	-	`reserved2`	-	Reserved
22 + 12*N	I2	-	`ic`	cm	Ionosphere correction in [cm]
End of repeated block					

Bitfield service

This Graphic explains the bits of `service`



39.9 NAV-SOL (0x01 0x06)

39.9.1 Navigation Solution Information

Message	**NAV-SOL**				
Description	**Navigation Solution Information**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	This message combines position, velocity and time solution in ECEF, including accuracy figures. This message has only been retained for backwards compatibility; users are recommended to use the `UBX-NAV-PVT` message in preference.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x06	52	*see below*	CK_A CK_B

Payload Contents:

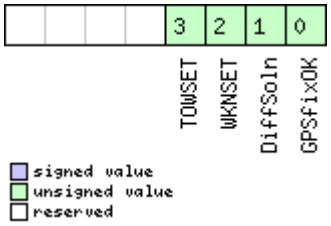
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	I4	-	`fTOW`	ns	Fractional part of iTOW (range: +/-500000). The precise GPS time of week in seconds is: `(iTOW * 1e-3) + (fTOW * 1e-9)`
8	I2	-	`week`	weeks	GPS week number of the navigation epoch

NAV-SOL continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
10	U1	-	`gpsFix`	-	GPSfix Type, range 0..5 0x00 = No Fix 0x01 = Dead Reckoning only 0x02 = 2D-Fix 0x03 = 3D-Fix 0x04 = GPS + dead reckoning combined 0x05 = Time only fix 0x06..0xff: reserved
11	X1	-	`flags`	-	Fix Status Flags (see graphic below)
12	I4	-	`ecefX`	cm	ECEF X coordinate
16	I4	-	`ecefY`	cm	ECEF Y coordinate
20	I4	-	`ecefZ`	cm	ECEF Z coordinate
24	U4	-	`pAcc`	cm	3D Position Accuracy Estimate
28	I4	-	`ecefVX`	cm/s	ECEF X velocity
32	I4	-	`ecefVY`	cm/s	ECEF Y velocity
36	I4	-	`ecefVZ`	cm/s	ECEF Z velocity
40	U4	-	`sAcc`	cm/s	Speed Accuracy Estimate
44	U2	0.01	`pDOP`	-	Position DOP
46	U1	-	`reserved1`	-	Reserved
47	U1	-	`numSV`	-	Number of SVs used in Nav Solution
48	U4	-	`reserved2`	-	Reserved

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`GPSfixOK`	>1 = Fix within limits (e.g. DOP & accuracy)
`DiffSoln`	1 = DGPS used
`WKNSET`	1 = Valid GPS week number
`TOWSET`	1 = Valid GPS time of week (iTOW & fTOW)

39.10 NAV-STATUS (0x01 0x03)

39.10.1 Receiver Navigation Status

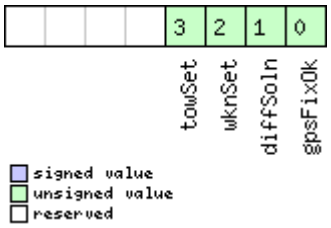
Message	**NAV-STATUS**				
Description	**Receiver Navigation Status**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	**See important comments concerning validity of position and velocity given in section Navigation Output Filters.** -				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x03	16	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	U1	-	`gpsFix`	-	GPSfix Type, this value does **not** qualify a fix as valid and within the limits. See note on flag gpsFixOk below. 0x00 = no fix 0x01 = dead reckoning only 0x02 = 2D-fix 0x03 = 3D-fix 0x04 = GPS + dead reckoning combined 0x05 = Time only fix 0x06..0xff = reserved
5	X1	-	`flags`	-	Navigation Status Flags (see graphic below)
6	X1	-	`fixStat`	-	Fix Status Information (see graphic below)
7	X1	-	`flags2`	-	further information about navigation output (see graphic below)
8	U4	-	`ttff`	-	Time to first fix (millisecond time tag)
12	U4	-	`msss`	-	Milliseconds since Startup / Reset

Bitfield flags

This Graphic explains the bits of `flags`



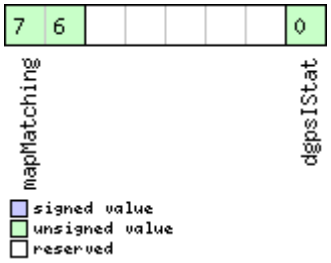
Name	*Description*
`gpsFixOk`	position and velocity valid and within DOP and ACC Masks, see also important comments in section Navigation Output Filters.
`diffSoln`	1 if DGPS used

Bitfield flags Description continued

Name	*Description*
`wknSet`	1 if Week Number valid
`towSet`	1 if Time of Week valid

Bitfield fixStat

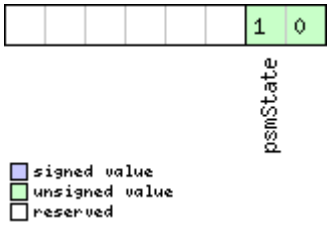
This Graphic explains the bits of `fixStat`



Name	*Description*
`dgpsIStat`	DGPS Input Status 0: none 1: PR+PRR Correction
`mapMatching`	map matching status, see section Map Matching Input for details. 00: none 01: valid, i.e. map matching data was received, but was too old 10: used, map matching data was applied 11: DR, map matching was the reason to enable the dead reckoning `gpsFix` type instead of publishing no fix

Bitfield flags2

This Graphic explains the bits of `flags2`



Name	*Description*
`psmState`	power save mode state 0: ACQUISITION [or when psm disabled] 1: TRACKING 2: POWER OPTIMIZED TRACKING 3: INACTIVE

39.11 NAV-SVINFO (0x01 0x30)

39.11.1 Space Vehicle Information

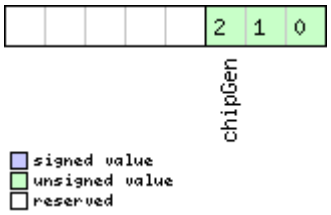
Message	**NAV-SVINFO**				
Description	**Space Vehicle Information**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x30	8 + 12*numCh	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	U1	-	`numCh`	-	Number of channels
5	X1	-	`globalFlags`	-	Bitmask (see graphic below)
6	U2	-	`reserved2`	-	Reserved
Start of repeated block (numCh times)					
8 + 12*N	U1	-	`chn`	-	Channel number, 255 for SVs not assigned to a channel
9 + 12*N	U1	-	`svid`	-	Satellite ID, see Satellite numbering for assignment
10 + 12*N	X1	-	`flags`	-	Bitmask (see graphic below)
11 + 12*N	X1	-	`quality`	-	Bitfield (see graphic below)
12 + 12*N	U1	-	`cno`	dBHz	Carrier to Noise Ratio (Signal Strength)
13 + 12*N	I1	-	`elev`	deg	Elevation in integer degrees
14 + 12*N	I2	-	`azim`	deg	Azimuth in integer degrees
16 + 12*N	I4	-	`prRes`	cm	Pseudo range residual in centimetres
End of repeated block					

Bitfield globalFlags

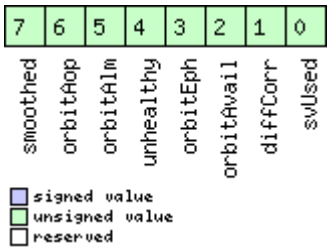
This Graphic explains the bits of `globalFlags`



Name	*Description*
`chipGen`	Chip hardware generation 0: Antaris, Antaris 4 1: u-blox 5 2: u-blox 6

Bitfield flags

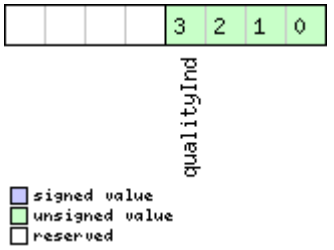
This Graphic explains the bits of `flags`



Name	*Description*
`svUsed`	SV is used for navigation
`diffCorr`	Differential correction data is available for this SV
`orbitAvail`	Orbit information is available for this SV (Ephemeris or Almanac)
`orbitEph`	Orbit information is Ephemeris
`unhealthy`	SV is unhealthy / shall not be used
`orbitAlm`	Orbit information is Almanac Plus
`orbitAop`	Orbit information is AssistNow Autonomous
`smoothed`	Carrier smoothed pseudorange used (see PPP for details)

Bitfield quality

This Graphic explains the bits of `quality`



Name	*Description*
`qualityInd`	Signal Quality indicator (range 0..7). The following list shows the meaning of the different QI values: 0: This channel is idle 1: Channel is searching 2: Signal aquired 3: Signal detected but unusable 4: Code Lock on Signal 5, 6, 7: Code and Carrier locked

39.12 NAV-TIMEGPS (0x01 0x20)

39.12.1 GPS Time Solution

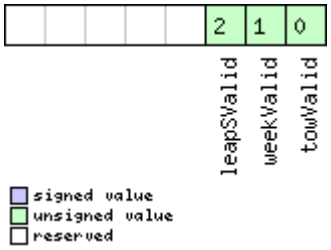
Message	**NAV-TIMEGPS**				
Description	**GPS Time Solution**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	This message reports the precise GPS time of the most recent navigation solution including validity falgs and an accuracy estimate.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x20	16	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	I4	-	`fTOW`	ns	Fractional part of iTOW (range: +/-500000). The precise GPS time of week in seconds is: `(iTOW * 1e-3) + (fTOW * 1e-9)`
8	I2	-	`week`	-	GPS week number of the navigation epoch
10	I1	-	`leapS`	s	GPS leap seconds (GPS-UTC)
11	X1	-	`valid`	-	Validity Flags (see graphic below)
12	U4	-	`tAcc`	ns	Time Accuracy Estimate

Bitfield valid

This Graphic explains the bits of `valid`



Name	*Description*
`towValid`	1 = Valid GPS time of week (iTOW & fTOW)
`weekValid`	1 = Valid GPS week number
`leapSValid`	1 = Valid GPS leap seconds

39.13 NAV-TIMEUTC (0x01 0x21)

39.13.1 UTC Time Solution

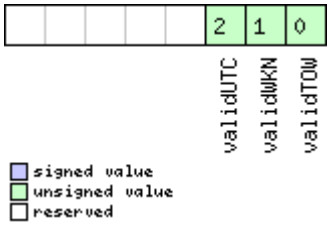
Message	**NAV-TIMEUTC**				
Description	**UTC Time Solution**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	**Note that during a leap second there may be more (or less) than 60 seconds in a minute; see the description of leap seconds for details.** -				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x21	20	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	U4	-	`tAcc`	ns	Time accuracy estimate (UTC)
8	I4	-	`nano`	ns	Fraction of second, range -1e9 .. 1e9 (UTC)
12	U2	-	`year`	y	Year, range 1999..2099 (UTC)
14	U1	-	`month`	month	Month, range 1..12 (UTC)
15	U1	-	`day`	d	Day of month, range 1..31 (UTC)
16	U1	-	`hour`	h	Hour of day, range 0..23 (UTC)
17	U1	-	`min`	min	Minute of hour, range 0..59 (UTC)
18	U1	-	`sec`	s	Seconds of minute, range 0..60 (UTC)
19	X1	-	`valid`	-	Validity Flags (see graphic below)

Bitfield valid

This Graphic explains the bits of `valid`



Name	*Description*
`validTOW`	1 = Valid Time of Week
`validWKN`	1 = Valid Week Number
`validUTC`	1 = Valid UTC Time

39.14 NAV-VELECEF (0x01 0x11)

39.14.1 Velocity Solution in ECEF

Message	**NAV-VELECEF**				
Description	**Velocity Solution in ECEF**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	**See important comments concerning validity of velocity given in section Navigation Output Filters.** -				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x11	20	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	I4	-	`ecefVX`	cm/s	ECEF X velocity
8	I4	-	`ecefVY`	cm/s	ECEF Y velocity
12	I4	-	`ecefVZ`	cm/s	ECEF Z velocity
16	U4	-	`sAcc`	cm/s	Speed accuracy estimate

39.15 NAV-VELNED (0x01 0x12)

39.15.1 Velocity Solution in NED

Message	**NAV-VELNED**				
Description	**Velocity Solution in NED**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	**See important comments concerning validity of velocity given in section Navigation Output Filters.** -				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x12	36	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	I4	-	`velN`	cm/s	North velocity component
8	I4	-	`velE`	cm/s	East velocity component
12	I4	-	`velD`	cm/s	Down velocity component
16	U4	-	`speed`	cm/s	Speed (3-D)
20	U4	-	`gSpeed`	cm/s	Ground speed (2-D)
24	I4	1e-5	`heading`	deg	Heading of motion 2-D

NAV-VELNED continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
28	U4	-	**sAcc**	cm/s	Speed accuracy Estimate
32	U4	1e-5	**cAcc**	deg	Course / Heading accuracy estimate

40 RXM (0x02)

Receiver Manager Messages: i.e. Satellite Status, RTC Status.
Messages in Class RXM output status and result data from the Receiver Manager.

40.1 RXM-ALM (0x02 0x30)

40.1.1 Poll GPS Constellation Almanac Data

Message	**RXM-ALM**				
Description	**Poll GPS Constellation Almanac Data**				
Firmware	Supported on: • u-blox 7 firmware version 1.00 (**only available with raw data product variant**)				
Type	Poll Request				
Comment	**This message has an empty payload!** Poll GPS Constellation Data (Almanac) for all 32 SVs by sending this message to the receiver without any payload. The receiver will return 32 messages of type RXM-ALM as defined below.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x02 0x30	0	*see below*	CK_A CK_B
No payload					

40.1.2 Poll GPS Constellation Almanac Data for a SV

Message	**RXM-ALM**				
Description	**Poll GPS Constellation Almanac Data for a SV**				
Firmware	Supported on: • u-blox 7 firmware version 1.00 (**only available with raw data product variant**)				
Type	Poll Request				
Comment	Poll GPS Constellation Data (Almanac) for an SV by sending this message to the receiver. The receiver will return one message of type RXM-ALM as defined below.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x02 0x30	1	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`svid`	-	SV ID for which the receiver shall return its Almanac Data (Valid Range: 1 .. 32).

40.1.3 GPS Aiding Almanac Input/Output Message

Message	**RXM-ALM**				
Description	**GPS Aiding Almanac Input/Output Message**				
Firmware	Supported on: • u-blox 7 firmware version 1.00 (**only available with raw data product variant**)				
Type	Poll Answer / Periodic				
Comment	**This message is provided considered obsolete, please use AID-ALM instead!** • If the WEEK Value is 0, DWRD0 to DWRD7 are not sent as the Almanac is not available for the given SV. • DWORD0 to DWORD7 contain the 8 words following the Hand-Over Word (HOW) from the GPS navigation message, either pages 1 to 24 of sub-frame 5 or pages 2 to 10 of subframe 4. See IS-GPS-200 for a full description of the contents of the Almanac pages. • In DWORD0 to DWORD7, the parity bits have been removed, and the 24 bits of data are located in Bits 0 to 23. Bits 24 to 31 shall be ignored. • Example: Parameter e (Eccentricity) from Almanac Subframe 4/5, Word 3, Bits 69-84 within the subframe can be found in DWRD0, Bits 15-0 whereas Bit 0 is the LSB.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x02 0x30	(8) or (40)	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`svid`	-	SV ID for which this Almanac Data is (Valid Range: 1 .. 32 or 51, 56, 63).
4	U4	-	`week`	-	Issue Date of Almanac (GPS week number)
Start of optional block					
8	U4[8]	-	`dwrd`	-	Almanac Words
End of optional block					

40.2 RXM-EPH (0x02 0x31)

40.2.1 Poll GPS Constellation Ephemeris Data

Message	**RXM-EPH**				
Description	**Poll GPS Constellation Ephemeris Data**				
Firmware	Supported on: • u-blox 7 firmware version 1.00 (**only available with raw data product variant**)				
Type	Poll Request				
Comment	**This message has an empty payload!** Poll GPS Constellation Data (Ephemeris) for all 32 SVs by sending this message to the receiver without any payload. The receiver will return 32 messages of type RXM-EPH as defined below.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x02 0x31	0	*see below*	CK_A CK_B
No payload					

40.2.2 Poll GPS Constellation Ephemeris Data for a SV

Message	**RXM-EPH**				
Description	**Poll GPS Constellation Ephemeris Data for a SV**				
Firmware	Supported on: • u-blox 7 firmware version 1.00 (**only available with raw data product variant**)				
Type	Poll Request				
Comment	Poll GPS Constellation Data (Ephemeris) for an SV by sending this message to the receiver. The receiver will return one message of type RXM-EPH as defined below.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x02 0x31	1	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`svid`	-	SV ID for which the receiver shall return its Ephemeris Data (Valid Range: 1 .. 32).

40.2.3 GPS Aiding Ephemeris Input/Output Message

Message	**RXM-EPH**				
Description	**GPS Aiding Ephemeris Input/Output Message**				
Firmware	Supported on: • u-blox 7 firmware version 1.00 (**only available with raw data product variant**)				
Type	Poll Answer / Periodic				
Comment	**This message is provided considered obsolete, please use AID-EPH instead!** • SF1D0 to SF3D7 is only sent if ephemeris is available for this SV. If not, the payload may be reduced to 8 Bytes, or all bytes are set to zero, indicating that this SV Number does not have valid ephemeris for the moment. • SF1D0 to SF3D7 contain the 24 words following the Hand-Over Word (HOW) from the GPS navigation message, subframes 1 to 3. See IS-GPS-200 for a full description of the contents of the Subframes. • In SF1D0 to SF3D7, the parity bits have been removed, and the 24 bits of data are located in Bits 0 to 23. Bits 24 to 31 shall be ignored.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x02 0x31	(8) or (104)	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`svid`	-	SV ID for which this ephemeris data is (Valid Range: 1 .. 32).
4	U4	-	`how`	-	Hand-Over Word of first Subframe. This is required if data is sent to the receiver. 0 indicates that no Ephemeris Data is following.
Start of optional block					
8	U4[8]	-	`sf1d`	-	Subframe 1 Words 3..10 (SF1D0..SF1D7)
40	U4[8]	-	`sf2d`	-	Subframe 2 Words 3..10 (SF2D0..SF2D7)
72	U4[8]	-	`sf3d`	-	Subframe 3 Words 3..10 (SF3D0..SF3D7)
End of optional block					

40.3 RXM-PMREQ (0x02 0x41)

40.3.1 Requests a Power Management task

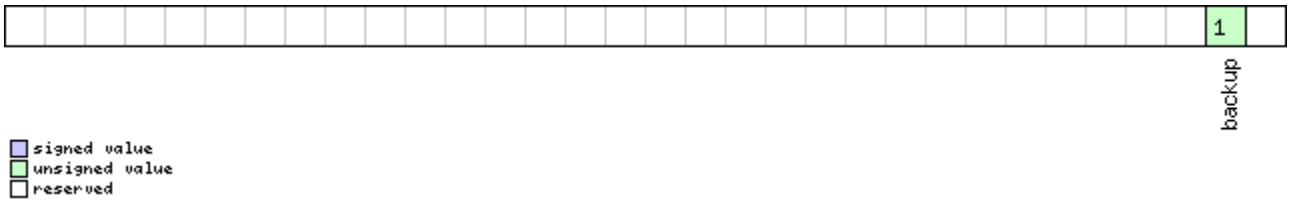
Message	**RXM-PMREQ**				
Description	**Requests a Power Management task**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Command				
Comment	Request of a Power Management related task of the receiver.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x02 0x41	8	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`duration`	ms	Duration of the requested task, set to zero for infinite duration
4	X4	-	`flags`	-	task flags (see graphic below)

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`backup`	The receiver goes into backup mode for a time period defined by duration

40.4 RXM-RAW (0x02 0x10)

40.4.1 Raw Measurement Data

Message	**RXM-RAW**				
Description	**Raw Measurement Data**				
Firmware	Supported on: • u-blox 7 firmware version 1.00 (**only available with raw data product variant**)				
Type	Periodic/Polled				
Comment	This message contains all information needed to be able to generate a RINEX observation file. This message outputs pseudorange, doppler and carrier phase measurements for GPS satellites once signals have been synchronised. No other GNSS types are currently supported.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x02 0x10	8 + 24*numSV	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*

RXM-RAW continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	I4	-	`rcvTow`	ms	Measurement time of week in receiver local time
4	I2	-	`week`	weeks	Measurement week number in receiver local time
6	U1	-	`numSV`	-	Number of satellites following
7	U1	-	`reserved1`	-	Reserved
Start of repeated block (numSV times)					
8 + 24*N	R8	-	`cpMes`	cycles	Carrier phase measurement [L1 cycles]
16 + 24*N	R8	-	`prMes`	m	Pseudorange measurement [m]
24 + 24*N	R4	-	`doMes`	Hz	Doppler measurement (positive sign for approaching satellites) [Hz]
28 + 24*N	U1	-	`sv`	-	Space Vehicle number
29 + 24*N	I1	-	`mesQI`	-	Nav Measurements Quality Indicator: >=4 : PR+DO OK >=5 : PR+DO+CP OK <6 : likely loss of carrier lock in previous interval
30 + 24*N	I1	-	`cno`	dBHz	Signal strength C/No
31 + 24*N	U1	-	`lli`	-	Loss of lock indicator (RINEX definition)
End of repeated block					

40.5 RXM-SFRB (0x02 0x11)

40.5.1 Subframe Buffer

Message	**RXM-SFRB**				
Description	**Subframe Buffer**				
Firmware	Supported on: • u-blox 7 firmware version 1.00 (**only available with raw data product variant**)				
Type	Periodic				
Comment	The content of one single subframe buffer For GPS satellites, the 10 dwrd values contain the parity checked subframe data for 10 Words. Each dwrd has 24 Bits with valid data (Bits 23 to 0). The remaining 8 bits (31 to 24) have an undefined value. The direction within the Word is that the higher order bits are received from the SV first. Example: The Preamble can be found in dwrd[0], at bit position 23 down to 16. For more details on the data format please refer to the ICD-GPS-200C Interface document. For SBAS satellites, the 250 Bit message block can be found in dwrd[0] to dwrd[6] for the first 224 bits. The remaining 26 bits are in dwrd[7], whereas Bits 25 and 24 are the last two data bits, and Bits 23 down to 0 are the parity bits. For more information on SBAS data format, please refer to RTCA/DO-229C (MOPS), Appendix A. No other GNSS types are currently supported.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x02 0x11	42	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*

RXM-SFRB continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`chn`	-	Channel Number
1	U1	-	`svid`	-	ID of Satellite transmitting Subframe
2	X4[10]	-	`dwrd`	-	Words of Data

40.6 RXM-SVSI (0x02 0x20)

40.6.1 SV Status Info

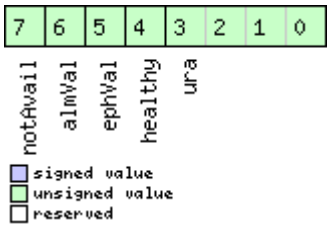
Message	**RXM-SVSI**				
Description	**SV Status Info**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	Status of the receiver manager knowledge about GPS Orbit Validity				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x02 0x20	8 + 6*numSV	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	I2	-	`week`	weeks	GPS week number of the navigation epoch
6	U1	-	`numVis`	-	Number of visible satellites
7	U1	-	`numSV`	-	Number of per-SV data blocks following
Start of repeated block (numSV times)					
8 + 6*N	U1	-	`svid`	-	Satellite ID
9 + 6*N	X1	-	`svFlag`	-	Information Flags (see graphic below)
10 + 6*N	I2	-	`azim`	-	Azimuth
12 + 6*N	I1	-	`elev`	-	Elevation
13 + 6*N	X1	-	`age`	-	Age of Almanac and Ephemeris: (see graphic below)
End of repeated block					

Bitfield svFlag

This Graphic explains the bits of `svFlag`



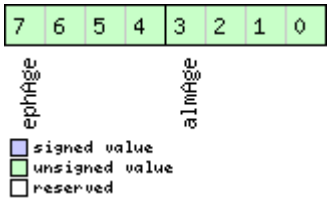
Name	*Description*
`ura`	Figure of Merit (URA) range 0..15
`healthy`	SV healthy flag
`ephVal`	Ephemeris valid

Bitfield svFlag Description continued

Name	*Description*
`almVal`	Almanac valid
`notAvail`	SV not available

Bitfield age

This Graphic explains the bits of `age`



Name	*Description*
`almAge`	Age of ALM in days offset by 4 i.e. the reference time may be in the future: ageOfAlm = (age & 0x0f) - 4
`ephAge`	Age of EPH in hours offset by 4. i.e. the reference time may be in the future: ageOfEph = ((age & 0xf0) >> 4) - 4

41 TIM (0x0D)

Timing Messages: i.e. Time Pulse Output, Timemark Results.
Messages in this class are output by the receiver, giving information on Timepulse and Timemark measurements.

41.1 TIM-TM2 (0x0D 0x03)

41.1.1 Time mark data

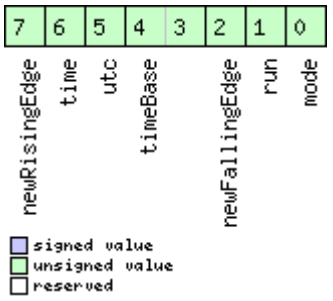
Message	**TIM-TM2**				
Description	**Time mark data**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	This message contains information for high precision time stamping / pulse counting. The delay figures and timebase given in `CFG-TP5` are also applied to the time results output in this message.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0D 0x03	28	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`ch`	time	marker channel 0 or 1
1	X1	-	`flags`	-	Bitmask (see graphic below)
2	U2	-	`count`	-	rising edge counter.
4	U2	-	`wnR`	-	week number of last rising edge
6	U2	-	`wnF`	-	week number of last falling edge
8	U4	-	`towMsR`	ms	tow of rising edge
12	U4	-	`towSubMsR`	ns	millisecond fraction of tow of rising edge in nanoseconds
16	U4	-	`towMsF`	ms	tow of falling edge
20	U4	-	`towSubMsF`	ns	millisecond fraction of tow of falling edge in nanoseconds
24	U4	-	`accEst`	ns	Accuracy estimate

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`mode`	0=single 1=running

Bitfield flags Description continued

Name	*Description*
`run`	0=armed 1=stopped
`newFallingEdge`	new falling edge detected
`timeBase`	0=Time base is Receiver Time 1=Time base is GPS 2=Time base is UTC
`utc`	0=UTC not available 1=UTC available
`time`	0=Time is not valid 1=Time is valid (Valid GPS fix)
`newRisingEdge`	new rising edge detected

41.2 TIM-TP (0x0D 0x01)

41.2.1 Time Pulse Timedata

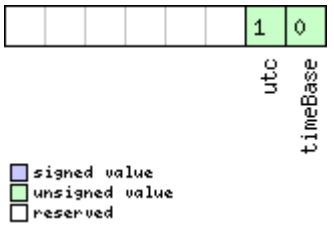
Message	**TIM-TP**				
Description	**Time Pulse Timedata**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Periodic/Polled				
Comment	This message contains information for high precision timing. The recommended configuration when using this message is to set both the measurement rate (`CFG-RATE`) and the timepulse frequency (`CFG-TP5`) to 1Hz. For more information see section Time pulse.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0D 0x01	16	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`towMS`	ms	Time pulse time of week according to time base
4	U4	2^-32	`towSubMS`	ms	Submillisecond part of TOWMS
8	I4	-	`qErr`	ps	Quantization error of time pulse.
12	U2	-	`week`	weeks	Time pulse week number according to time base
14	X1	-	`flags`	-	bitmask (see graphic below)
15	U1	-	`reserved1`	-	Reserved

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`timeBase`	0=Time base is GPS 1=Time base is UTC
`utc`	0=UTC not available 1=UTC available

41.3 TIM-VRFY (0x0D 0x06)

41.3.1 Sourced Time Verification

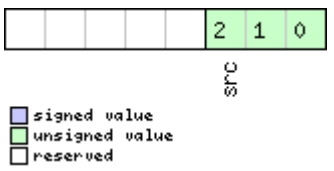
Message	**TIM-VRFY**				
Description	**Sourced Time Verification**				
Firmware	Supported on: • u-blox 7 firmware version 1.00				
Type	Polled/Once				
Comment	This message contains verification information about previous time received via AID-INI or from RTC				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0D 0x06	20	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	I4	-	`itow`	ms	integer millisecond tow received by source
4	I4	-	`frac`	ns	sub-millisecond part of tow
8	I4	-	`deltaMs`	ms	integer milliseconds of delta time (current time minus sourced time)
12	I4	-	`deltaNs`	ns	sub-millisecond part of delta time
16	U2	-	`wno`	week	week number
18	X1	-	`flags`	-	information flags (see graphic below)
19	U1	-	`reserved1`	-	Reserved

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*

Bitfield flags Description continued

Name	*Description*
`src`	aiding time source 0: no time aiding done 2: source was RTC 3: source was AID-INI

RTCM Protocol

42 Introduction

The RTCM (Radio Technical Commission for Maritime Services) protocol is a unidirectional protocol (input to the receiver) that is used to supply the GPS receiver with real-time differential correction data (DGPS). The RTCM protocol specification is available from http://www.rtcm.org.

This feature is only applicable to GPS operation.

43 Supported Messages

The following RTCM 2.3 messages are supported:

Supported RTCM 2.3 Message Types

Message Type	*Description*
1	Differential GPS Corrections
2	Delta Differential GPS Corrections
3	GPS Reference Station Parameters
9	GPS Partial Correction Set

44 Configuration

The DGPS feature does not need any configuration to work properly. When an RTCM stream is input on any of the communication interfaces, the data will be parsed and applied if possible, which will put the receiver into DGPS mode. However, the RTCM protocol must be enabled on the interface used by means of the UBX-CFG-PRT message.

The only configurable parameter of DGPS mode is the timeout that can be specified using UBX-CFG-NAV5. This value defines the time after which old RTCM data will be discarded.

45 Output

DGPS mode will result in following modified output:

- NMEA-GGA: The quality field will be 2 (see NMEA Positon Fix Flags). The age of DGPS corrections and Reference station id will be set.
- NMEA-GLL, NMEA-RMC, NMEA-VTG, NMEA-GNS: The posMode indicator will be D (see NMEA Positon Fix Flags).
- NMEA-PUBX-POSITION: The status will be D2/D3; The age of DGPS corrections will be set.
- UBX-NAV-SOL: The DGPS will be set.
- UBX-NAV-PVT: The DGPS will be set.
- UBX-NAV-STATUS: The DGPS will be set; The DGPS input will be set to "PR+PRR".
- UBX-NAV-SVINFO: The DGPS flag will be set for channels with valid DGPS correction data.
- UBX-NAV-DGPS: This message will contain all valid DGPS data
- If the base line exceeds 100km and a message type 3 is received, a UBX-INF-WARNING will be output, e.g. "WARNING: DGPS baseline big: 330.3km"

46 Restrictions

The following restrictions apply to DGPS mode:

- The DGPS solution will only include measurements from satellites for which DGPS corrections were provided. This is because the navigation algorithms cannot mix corrected with uncorrected measurements.
- SBAS corrections will not be applied when using RTCM correction data.
- Precise Point Positioning will be deactivated when using RTCM correction data.
- RTCM correction data cannot be applied when using *AssistNow Offline* or *AssistNow Autonomous*.

47 Reference

The RTCM support is implemented according to RTCM 10402.3 ("RECOMMENDED STANDARDS FOR DIFFERENTIAL GNSS").

Appendix

A Protocol Versions

The Protocol Version defines a set of messages that are applicable across various u-blox products. Each firmware used by a u-blox receiver supports a specific Protocol Version, which is not configurable.

Each receiver reports its supported Protocol Version in the following ways:

- On start-up in the 'boot screen'
- In the UBX-MON-VER message

The following tables show the supported Protocol Versions for a number of common firmware versions and platforms.

A.1 Supported Protocol Versions

u-blox 5

Firmware Version	*Supported Protocol Version*
4.00	10.00
4.01	10.01
5.00	11.00
6.00	12.00
6.02	12.02

u-blox 6

Firmware Version	*Supported Protocol Version*
6.00	12.00
6.02	12.02
7.01	13.01
7.03	13.03

u-blox 6 GPS/GLONASS/QZSS

Firmware Version	*Supported Protocol Version*
1.00	14.00

u-blox 7

Firmware Version	*Supported Protocol Version*
1.00	14.00

B u-blox 7 Default Settings

The default settings listed in this section apply from u-blox 7 ROM-based receivers with ROM version 1.00 and above. These values assume that the default levels of the configuration pins have been left unchanged and no setting that affects the default configuration was written to the eFuse. Default settings are dependent on the configuration pin and eFuse settings, for information regarding these settings, consult the applicable Data Sheet.

B.1 Antenna Supervisor Settings (UBX-CFG-ANT)

For parameter and protocol description see section UBX-CFG-ANT.

Antenna Settings

Parameter	*Description*	*Default Setting*	*Unit*
flags-svcs	Enable Control Signal	Enabled	
flags-scd	Enable Short Circuit Detection	Enabled	
flags-pdwnOnSCD	Enable Short Circuit Power Down logic	Enabled	
flags-recovery	Enable Automatic Short Circuit Recovery logic	Enabled	
flags-ocd	Enable Open Circuit Detection	Disabled	
pins-pinSwitch	PIO-Pin used for switching antenna supply	16	
pins-pinSCD	PIO-Pin used for detecting a short in the antenna supply	15	
pins-pinOCD	PIO-Pin used for detecting open/not connected antenna	14	

B.2 Datum Settings (UBX-CFG-DAT)

For parameter and protocol description see section UBX-CFG-DAT.

Datum Default Settings

Parameter	*Description*	*Default Setting*	*Unit*
datumNum	Datum number	0	
datumName	Datum name	WGS84	
majA	Semi-major Axis	6378137	m
flat	1.0 / Flattening	298.257223563	
dX	X Axis shift at the origin	0	m
dY	Y Axis shift at the origin	0	m
dZ	Z Axis shift at the origin	0	m
rotX	Rotation about the X Axis	0	s
rotY	Rotation about the Y Axis	0	s
rotZ	Rotation about the Z Axis	0	s
scale	Scale change	0	ppm

B.3 Navigation Settings (UBX-CFG-NAV5)

For parameter and protocol description see section UBX-CFG-NAV5.

Navigation Default Settings

Parameter	*Description*	*Default Setting*	*Unit*
dynModel	Dynamic Platform Model	0 - Portable	
fixMode	Fix Mode	3 - Auto 2D/3D	
fixedAlt	Fixed Altitude	N/A (fixMode=3)	m
fixedAltVar	Fixed Altitude Variance	N/A (fixMode=3)	m^2
minElev	Min SV Elevation	5	deg
pDop	PDOP Mask	25	-
tDop	TDOP Mask	25	-
pAcc	P Accuracy	100	m
tAcc	T Accuracy	300	m
staticHoldThresh	Static Hold Threshold	0.00	cm/s

Navigation Default Settings continued

Parameter	*Description*	*Default Setting*	*Unit*
dgpsTimeOut	DGPS timeout	60	s
cnoThreshNumSVs	Number of SVs required to have C/N0 above cnoThresh for a valid fix	0	
cnoThresh	C/N0 threshold for a valid fix	0	dBHz

The Dynamic Platform Model default setting is different for certain product variants.

B.4 Navigation Settings (UBX-CFG-NAVX5)

For parameter and protocol description see section UBX-CFG-NAVX5.

Navigation Default Settings

Parameter	*Description*	*Default Setting*	*Unit*
minSVs	Minimum number of SV	3	
maxSVs	Maximum number of SV	22	
minCNO	Minimum C/N0 for navigation	7	dBHz
iniFix3D	Initial Fix must be 3D	Disabled	
aopCfg-useAOP	Use *AssistNow Autonomous*	Disabled	
aopOrbMaxErr	*AssistNow Autonomous* max. acceptable orbit error	100	m
wknRollover	Weeknumber rollover	1691	

The minimun number of SV default setting is different for certain product variants.

B.5 Output Rates (UBX-CFG-RATE)

For parameter and protocol description see section UBX-CFG-RATE.

Output Rate Default Settings

Parameter	*Description*	*Default Setting*	*Unit*
timeRef	Time Source	1 – GPS time	
measRate	Measurement Period	1000	ms
navRate	Measurement Rate	1	Cycles

B.6 Power Management 2 Configuration (UBX-CFG-PM2)

For parameter and protocol description see section UBX-CFG-PM2.

Power Management 2 Configuration Default Settings

Parameter	*Description*	*Default Setting*	*Unit*
version	Version	1	
flags-extintSelect	EXTINT pin selection	EXTINT0	
flags-extintWake	EXTINT pin control - keep awake	Disabled	
flags-extintBackup	EXTINT pin control - force backup	Disabled	
flags-limitPeakCurr	Limit peak current	Disabled	
flags-WaitTimeFix	Wait for time fix	Disabled	
flags-updateRTC	Update Real Time Clock	Disabled	
flags-updateEPH	Update ephemeris	Enabled	
flags-doNotEnterOff	Do not enter 'inactive for search' state when no fix	Disabled	

Power Management 2 Configuration Default Settings continued

Parameter	*Description*	*Default Setting*	*Unit*
flags-mode	Mode of operation	Cyclic tracking	
updatePeriod	Update period	1000	ms
searchPeriod	Search period	10000	ms
gridOffset	Grid offset	0	ms
onTime	On time	0	s
minAcqTime	Minimum acquisition time	0	s

B.7 Receiver Manager Configuration (UBX-CFG-RXM)

For parameter and protocol description see section UBX-CFG-RXM.

Power Management Default Settings

Parameter	*Description*	*Default Setting*	*Unit*
lpMode	Low power mode	0 - Continuous Mode	

B.8 GNSS system configuration (UBX-CFG-GNSS)

For parameter and protocol description see section UBX-CFG-GNSS.

UBX-CFG-GNSS Default Settings

Parameter	*Description*	*Default Setting*	*Unit*
numTrkChHw	Number of available tracking channels	22	
numTrkChUse	Number of tracking channels to use	22	
numConfigBlocks	Number of configuration blocks following	4	
gnssId	GNSS identifier (see Satellite Numbering)	0, 1, 5, 6	
flags-enable	Enable this GNSS system	1, 1, 1, 0	
resTrkCh	Minimum number of tracking channels per GNSS	4, 1, 0, 8	
maxTrkCh	Maximum number of tracking channels per GNSS	255, 3, 3, 255	

B.9 SBAS Configuration (UBX-CFG-SBAS)

For parameter and protocol description see section UBX-CFG-SBAS.

SBAS Configuration Default Settings

Parameter	*Description*	*Default Setting*	*Unit*
mode-enabled	SBAS Subsystem	Enabled	
mode-test	Allow test mode usage	Disabled	
usage-range	Ranging (Use SBAS for navigation)	Enabled	
usage-diffCorr	Apply SBAS Correction Data	Enabled	
usage-integrity	Apply integrity information	Disabled	
scanmode1	PRN Codes 120-151	120, 124, 126, 127, 129, 133, 135, 137, 138	
scanmode2	PRN Codes 152-158	None	

B.10 Port Configuration (UBX-CFG-PRT)

For parameter and protocol description see section UBX-CFG-PRT.

B.10.1 UART Port Configuration

For parameter and protocol description see section UBX-CFG-PRT-UART.

UART 1 Default Settings

Parameter	*Description*	*Default Setting*	*Unit*
portID	Port ID	1 (UART 1)	
txReady-en	TX-ready feature	0 (disabled)	
mode-charLen	Character Length	3 (8 bit)	
mode-parity	Parity	4 (No parity)	
mode-nStopBits	Number of Stop Bits	0 (1 stop bit)	
baudRate	Baud rate	9600	baud
inProtoMask	Protocol in	UBX, NMEA, RTCM	
outProtoMask	Protocol out	UBX, NMEA	
flags-extendedTxTimeout	Extended TX timeout	0 - disabled	

B.10.2 USB Port Configuration

For parameter and protocol description see section UBX-CFG-PRT-USB.

USB Default Settings

Parameter	*Description*	*Default Setting*	*Unit*
portID	Port ID	3 (USB)	
txReady-en	TX-ready feature	0 (disabled)	
inProtoMask	Protocol in	UBX, NMEA, RTCM	
outProtoMask	Protocol out	UBX, NMEA	

B.10.3 SPI Port Configuration

For parameter and protocol description see section UBX-CFG-PRT-SPI.

SPI Default Settings

Parameter	*Description*	*Default Setting*	*Unit*
portID	Port ID	4 (SPI)	
txReady-en	TX-ready feature	0 (disabled)	
mode-spiMode	SPI mode	0 (CPOL=0, CPHA=0)	
mode-ffCnt	0xFF count	50	
inProtoMask	Protocol in	UBX, NMEA, RTCM	
outProtoMask	Protocol out	UBX, NMEA	
flags-extendedTxTimeout	Extended TX timeout	0 - disabled	

B.10.4 DDC Port Configuration

For parameter and protocol description see section UBX-CFG-PRT-DDC.

DDC Default Settings

Parameter	*Description*	*Default Setting*	*Unit*
portID	Port ID	0 (DDC)	
txReady-en	TX-ready feature	0 (disabled)	
mode-slaveAddr	Slave address	0x42	
inProtoMask	Protocol in	UBX, NMEA, RTCM	
outProtoMask	Protocol out	UBX, NMEA	
flags-extendedTxTimeout	Extended TX timeout	0 - disabled	

B.11 USB Settings (UBX-CFG-USB)

For parameter and protocol description see section UBX-CFG-USB.

USB default settings

Parameter	*Description*	*Default Setting*	*Unit*
vendorID	Vendor ID	0x1546	
productID	Product ID	0x01A7	
powerConsumption	Bus Current required	100	mA
flags-powerMode	Power Mode	1 (self-powered)	
vendorString	String containing the vendor name	u-blox AG - www.u-blox.com	
productString	String containing the product name	u-blox 7 - GPS/GNSS Receiver	

B.12 Message Settings (UBX-CFG-MSG)

For parameter and protocol description see section UBX-CFG-MSG.

Enabled output messages

Message	*Type*	*All Ports*
NMEA-Standard-GGA	Out	1
NMEA-Standard-GLL	Out	1
NMEA-Standard-GSA	Out	1
NMEA-Standard-GSV	Out	1
NMEA-Standard-RMC	Out	1
NMEA-Standard-VTG	Out	1

B.13 NMEA Protocol Settings (UBX-CFG-NMEA)

For parameter and protocol description see section UBX-CFG-NMEA.

NMEA Protocol Default Settings

Parameter	*Description*	*Default Setting*	*Unit*
filter-posFilt	Enable position output even for failed or invalid fixes	Disabled	
filter-mskPosFilt	Enable position even for invalid fixes	Disabled	
filter-timeFilt	Enable time output even for invalid times	Disabled	

NMEA Protocol Default Settings continued

Parameter	*Description*	*Default Setting*	*Unit*
filter-dateFilt	Enable time output even for invalid dates	Disabled	
filter-gpsOnlyFilter	Restrict output to GPS satellites only	Disabled	
filter-trackFilt	Enable COG output even if COG is frozen	Disabled	
nmeaVersion	NMEA version	2.3	
numSV	Number of SVs to report	Unlimited	
flags-compat	Compatibility Mode	Disabled	
flags-consider	Consideration Mode	Enabled	
gnssToFilter-gps	Disable GPS satellites	False	
gnssToFilter-sbas	Disable SBAS satellites	False	
gnssToFilter-qzss	Disable QZSS satellites	False	
gnssToFilter-glonass	Disable GLONASS satellites	False	
svNumbering	Output of SV's with no NMEA defined value	0 (not output)	
mainTalkerId	Override main Talker ID	0 (not overridden)	
gsvTalkerId	Override GSV Talker ID	0 (not overridden)	

B.14 Logging Configuration (UBX-CFG-LOGFILTER)

For parameter and protocol description see section UBX-CFG-LOGFILTER.

UBX-CFG-LOGFILTER Default Settings

Parameter	*Description*	*Default Setting*	*Unit*
flags-recordEnabled	Recording enabled	0	
flags-applyAllFilterSettings	Apply all filter settings	0	
flags-psmOncePerWakupEnabled	Recording of single position per PSM wake up enabled	0	
minInterval	Minimum time interval	0	s
timeThreshold	Time threshold	0	s
speedThreshold	Speed threshold	0	m/s
positionThreshold	Position threshold	0	m

B.15 Remote Inventory (UBX-CFG-RINV)

For parameter and protocol description see section UBX-CFG-RINV.

UBX-CFG-RINV Default Settings

Parameter	*Description*	*Default Setting*	*Unit*
flags-dump	Dump data at startup	0	
flags-binary	Data is binary	0	
data	Data stored in Remote Inventory	Notice: no data saved!	

B.16 INF Messages Settings (UBX-CFG-INF)

For parameter and protocol description see section UBX-CFG-INF.

INF messages default settings

Parameter	*Type*	*All Ports*	*Range/Remark*
infMsgMask-ERROR	Out	1	In NMEA Protocol only (GPTXT)
infMsgMask-WARNING	Out	1	In NMEA Protocol only (GPTXT)

INF messages default settings continued

Parameter	*Type*	*All Ports*	*Range/Remark*
infMsgMask-NOTICE	Out	1	In NMEA Protocol only (GPTXT)
infMsgMask-TEST	Out		
infMsgMask-DEBUG	Out		

B.17 Timepulse Settings (UBX-CFG-TP5)

For parameter and protocol description see section UBX-CFG-TP5.

TIMEPULSE default settings

Parameter	*Description*	*Default Setting*	*Unit*
tpIdx	Time pulse selection	0	ns
antCableDelay	Cable Delay	50	ns
rfGroupDelay	RF Groupdelay	0	ns
freqPeriod	Period	1000000	us
freqPeriodLock	Period Locked	1000000	us
pulseLenRatio	Pulse Length	0	us
pulseLenRatioLock	Pulse Length Locked	100000	us
userConfigDelay	User Delay	0	ns
flags-gridUtcGps	Timegrid	1 (GPS Time)	
flags-polarity	Polarity	1 (rising edge at top of second)	
flags-alignToTow	Align to TOW	1	
flags-isLength	IsLength	1	
flags-isFreq	IsFreq	0	
flags-lockedOtherSet	Locked other setting	1	
flags-LockGpsFreq	Lock to GPS freq	1	
flags-Active	Active	1	

TIMEPULSE2 default settings

Parameter	*Description*	*Default Setting*	*Unit*
tpIdx	Time pulse selection	1	ns
antCableDelay	Cable Delay	50	ns
rfGroupDelay	RF Groupdelay	0	ns
freqPeriod	Frequency	4	Hz
freqPeriodLock	Frequency Locked	1	Hz
pulseLenRatio	Pulse Length	125000	us
pulseLenRatioLock	Pulse Length Locked	100000	us
userConfigDelay	User Delay	0	ns
flags-gridUtcGps	Timegrid	1 (GPS Time)	
flags-polarity	Polarity	1 (rising edge at top of second)	
flags-alignToTow	Align to TOW	1	
flags-isLength	IsLength	1	
flags-isFreq	IsFreq	1	
flags-lockedOtherSet	Locked other setting	1	
flags-LockGpsFreq	Lock to GPS freq	1	
flags-Active	Active	0	

B.18 Jammer/Interference Monitor (UBX-CFG-ITFM)

For parameter and protocol description see section UBX-CFG-ITFM.

Jamming/Interference monitor default settings

Parameter	*Description*	*Default Setting*	*Unit*
config-enable	Enable	Disabled	
config-bbThreshold	Broadband interference detection threshold	3	dB
config-cwThreshold	CW interference detection threshold	15	dB
config-antSetting	Antenna setting	0	

C u-blox 7 Standard firmware versions

Standard FW version strings

Generation	*Version*	*String*	*ROM BASE*
u-blox 7		ROM CORE 1.00 (59842) Jun 27 2012 17:43:52	-
u-blox 7	FW 1.00	EXT CORE 1.00 (59843) Jun 27 2012 18:25:33	u-blox 7 ROM 1.00

Related Documents

Overview

As part of our commitment to customer support, u-blox maintains an extensive volume of technical documentation for our products. In addition to product-specific data sheets and integration manuals, general documents are also available. These include:

- GPS Compendium, Docu. No GPS-X-02007
- GPS Antennas - RF Design Considerations for u-blox GPS Receivers, Docu. No GPS-X-08014

Our website www.u-blox.com is a valuable resource for general and product specific documentation.

For design and integration projects the Receiver Description Including Protocol Specification should be used together with the Data Sheet and Hardware Integration Manual of the GPS receiver.

Contact

For complete contact information visit us at www.u-blox.com

u-blox Offices

North, Central and South America

u-blox America, Inc.
Phone: +1 703 483 3180
E-mail: info_us@u-blox.com

Regional Office West Coast:
Phone: +1 408 573 3640
E-mail: info_us@u-blox.com

Technical Support:
Phone: +1 703 483 3185
E-mail: support_us@u-blox.com

Headquarters
Europe, Middle East, Africa

u-blox AG
Phone: +41 44 722 74 44
E-mail: info@u-blox.com
Support: support@u-blox.com

Asia, Australia, Pacific

u-blox Singapore Pte. Ltd.
Phone: +65 6734 3811
E-mail: info_ap@u-blox.com
Support: support_ap@u-blox.com

Regional Office China (Beijing):
Phone: +86 10 68 133 545
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office China (Shenzhen):
Phone: +86 755 8627 1083
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office India:
Phone: +91 959 1302 450
E-mail: info_in@u-blox.com
Support: support_in@u-blox.com

Regional Office Japan:
Phone: +81 3 5775 3850
E-mail: info_jp@u-blox.com
Support: support_jp@u-blox.com

Regional Office Korea:
Phone: +82 2 542 0861
E-mail: info_kr@u-blox.com
Support: support_kr@u-blox.com

Regional Office Taiwan:
Phone: +886 2 2657 1090
E-mail: info_tw@u-blox.com
Support: support_tw@u-blox.com