POSITIONING

u-blox 8 / u-blox M8

Addendum to Protocol Specification for HPG 1.20

Abstract

This document should be used in conjunction with u-blox 8 / u-blox M8 Receiver Description including Protocol Specification UBX-13003221 revision R10. The document provides the receiver description for High Precision GNSS (HPG) and a summary of the new and modified UBX protocol messages applicable to HPG functions of u-blox HPG 1.20 firmware. Where references are made to details outside the summary scope of this document, please contact your supporting u-blox Applications team for further information if necessary.

www.u-blox.com

UBX-16004304 - R03

Document Information	
Title	**u-blox 8 / u-blox M8 Addendum to Protocol Specification for HPG 1.20**
Subtitle	
Document type	Manual
Document number	UBX-16004304
Revision and date	R03 (f215677) 6 October 2016
Document status	Advance Information

Document status explanation	
Objective Specification	Document contains target values. Revised and supplementary data will be published later.
Advance Information	Document contains data based on early testing. Revised and supplementary data will be published later.
Early Production Information	Document contains data from product verification. Revised and supplementary data may be published later.
Production Information	Document contains the final product specification.

Table of Contents

Receiver Description

1 Time Mode Configuration

This feature is only available with Timing, FTS or High Precision GNSS (HPG) products

This section relates to the configuration message UBX-CFG-TMODE2 (for Timing or FTS products) and to the configuration message UBX-CFG-TMODE3 (for HPG products).

1.1 Introduction

Time Mode is a special receiver mode where the position of the receiver is known and fixed and only the time is calculated using all available satellites. This mode allows for maximum time accuracy, for single-SV solutions, and also for using the receiver as a stationary reference station.

1.2 Fixed Position

In order to use the *Time Mode*, the receiver's position must be known as exactly as possible. Either the user already knows and enters the position, or it is determined using Survey-in. Errors in the fixed position will translate into time errors depending on the satellite constellation.

For Timing products, as a rule of thumb the position should be known with an accuracy of better than 1 m for a timing accuracy in the order of nanoseconds. If an accuracy is required only in the order of microseconds, a position accuracy of roughly 300 m is sufficient.

For HPG products, errors in the reference station position will directly translate into rover position errors. The reference station position accuracy should therefore be at least as good as the desired rover absolute position accuracy.

1.3 Survey-in

Survey-in is the procedure that is carried out prior to using *Time Mode*. It determines a stationary receiver's position by building a weighted mean of all valid 3D position solutions.

Two requirements for stopping the procedure must be specified:

- The **minimum observation time** defines a minimum amount of observation time regardless of the actual number of valid fixes that were used for the position calculation. Reasonable values range from one day for high accuracy requirements to a few minutes for coarse position determination.
- The **required 3D position standard deviation** defines a limit on the spread of positions that contribute to the calculated mean. As the position error translates into a time error when using *Time Mode* (see above), one should carefully evaluate the time accuracy requirements and choose an appropriate value.

Survey-in ends, when **both** requirements are met. After Survey-in has finished successfully, the receiver will automatically enter fixed position *Time Mode*. The Survey-in status can queried using the UBX-TIM-SVIN message for Timing or FTS products or the UBX-NAV-SVIN message for HPG products.

The "Standard Deviation" parameter defines uncertainty of the manually provided "True Position" set of parameters. This uncertainty directly affects the accuracy of the timepulse. This is to prevent an error that would otherwise be present in the timepulse because of the initially inaccurate position (assumed to be correct by the receiver) without users being aware of it. The "3D accuracy" parameter in "Fixed Position" as well as the "Position accuracy limit" in "Survey-in" affect the produced time information and the timepulse in the same way. Please note that the availability of the position accuracy does not mitigate the error in the timepulse but only accounts for it when

calculating the resulting time accuracy.

Once a survey-in has been started, its progress is saved in non-volatile memory, and hence continues over events such as a reset, receiver restart, or change of satellite constellation. If a survey-in position is required using data only for a particular receiver configuration, then any on-going survey-in should be stopped by either a `UBX-CFG-TMODE2` *or a* `UBX-CFG-TMODE3` *message with the timeMode field set to 0, then the receiver configured as required, and then a new* `UBX-CFG-TMODE2` *or* `UBX-CFG-TMODE3` *message sent with the new survey-in parameters.*

2 RTK Mode Configuration

This feature is only available with the High Precision GNSS products

u-blox RTK technology introduces the concept of a *reference station* and a *rover*. Using the RTCM3 protocol, the reference station sends corrections to the rover via a communication link enabling the rover to compute its position relative to the reference with high accuracy.

In the high precision GNSS context, the terms reference station and base station can be used interchangeably.

The distance between the reference station and the rover is called baseline length.

The reference station can provide correction to several rovers but the rover cannot concurrently process corrections from several reference stations.

The remainder of this chapter describes how to configure the reference station and the rover. More details about the RTCM3 protocol can be found in the RTCM3 section.

2.1 Reference Station Mode Configuration

Reference Station Mode is a special receiver mode where the receiver uses measurements from all available satellites to broadcast corrections. Configuring a stationary reference station is done in two steps:

- The receiver must be set in *Time Mode* using the configuration steps described in the Time Mode Configuration section.
- The RTCM3 correction stream must be configured following the rules detailed in the RTCM3 Configuration section. Each RTCM message must be individually enabled using `UBX-CFG-MSG`.

By default the reference station will begin operation in standard GNSS mode without any RTCM output. Messages for observations will be streamed as soon as they are configured for output. However messages for the reference station position will only be output when both the reference station is in fixed position mode, and the message is configured for output. As explained in the Time Mode Configuration section, this mode can be directly configured or reached at the end of a successful survey-in.

The rover will need to have received both reference station observation messages and reference station position messages in order to attempt ambiguity fixes.

When the reference station is in Time Mode, some error checking is performed on the entered, or surveyed-in, fixed position. If the result of these checks indicates that the fixed position may be incorrect, then a `UBX-INF-WARNING` *message will be sent, with the text "Reference Station position seems incorrect".*

2.2 Rover Mode Configuration

The RTK rover can be configured to work in either of these two differential modes using UBX-CFG-DGNSS:

- **RTK fixed:** In this mode, the rover will attempt to fix ambiguities whenever possible.
- **RTK float:** In this mode, the rover will estimate the ambiguities as float but will make no attempts at fixing them.

By default the rover will begin operation in RTK fixed mode. Upon receiving an RTCM3 correction stream on any of its communication interfaces, the rover will parse the data, apply the correction and, if possible, fix ambiguities. In absence of correction data or if the correction data times out, the rover will operate in standard GNSS mode.

The GLONASS ambiguities are only estimated as float, even in RTK fixed mode.

The time needed to resolve the ambiguity is affected by the baseline length as well as by multipath and satellite visibility at both rover and reference station.

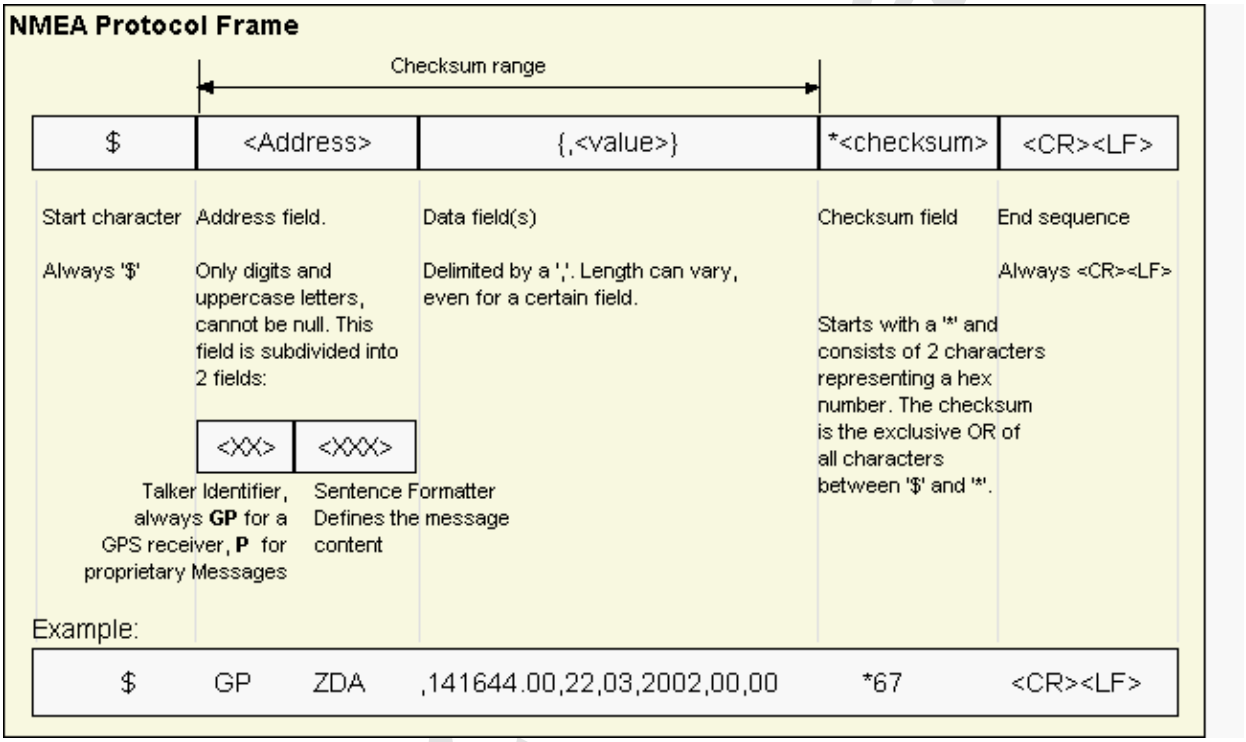
Protocol Specification

3 NMEA Protocol

3.1 Protocol Overview

3.1.1 Message Format

NMEA messages sent by the GNSS receiver are based on NMEA 0183 Version 4.0. The following picture shows the structure of a NMEA protocol message.



For further information on the NMEA Standard, refer to *NMEA 0183 Standard For Interfacing Marine Electronic Devices*, Version 4.00, November 1, 2008. 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.

3.1.2 Talker ID

One of the ways the NMEA standard differentiates between GNSS is by using a two-letter message identifier, the 'Talker ID'. The specific Talker ID used by a u-blox receiver will depend on the device model and system configuration. The table below shows the Talker ID that will be used for various GNSS configurations.

NMEA Talker IDs

Configured GNSS	*Talker ID*
GPS, SBAS, QZSS	GP
GLONASS	GL
Galileo	GA

NMEA Talker IDs continued

Configured GNSS	*Talker ID*
BeiDou	GB
Any combination of GNSS	GN

3.1.3 Protocol Configuration

The NMEA protocol on u-blox receivers can be configured to the need of customer applications using CFG-NMEA. For backwards compatibility various versions of this message are supported, however, any new users should use the version that is not marked as deprecated.

There are four NMEA standards supported. The default NMEA version is 4.0. Alternatively versions 4.1, 2.3, and 2.1 can be enabled (for details on how this affects the output refer to section Position Fix Flags in NMEA Mode).

Customers using BeiDou and/or Galileo are recommended to select NMEA version 4.1, as earlier versions have no support for these two GNSS.

Customers using High Precision GNSS (HPG) products are recommended to select NMEA version 4.1, as earlier versions do no support the Float RTK (F) and Real Time Kinematic (R) mode indicator flags in all messages.

NMEA defines satellite numbering systems for some, but not all GNSS (this is partly dependent on the NMEA version). Satellite numbers for unsupported 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 receivers 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.

NMEA flags continued

Parameter	*Description*
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.
Limit82 Mode	Enabling this mode will limit the NMEA sentence length to a maximum of 82 characters.
High Precision Mode	Enabling this mode increases precision of the position output. Latitude and longitude then have seven digits after the decimal point, and altitude has three digits after the decimal point. Note: The High Precision Mode cannot be set in conjunction with either Compatibility Mode or Limit82 Mode.

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.
BDS Talker ID	By default the Talker ID for BeiDou is 'GB'. This field enableds the BeiDou Talker ID to be overridden.

Extra fields in NMEA 4.1 and above

Message	*Extra fields*
GBS	systemId, signalId
GNS	navStatus
GRS	systemId, signalId
GSA	systemId
GSV	signalId
RMC	navStatus

3.1.4 Satellite Numbering

The NMEA protocol (V4.0) 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.

See Satellite Numbering Summary for a complete list of satellite numbers.

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).

3.1.5 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

3.1.6 Position Fix Flags

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

Flags in NMEA 4.1 and above

NMEA Message *Field*	*GLL, RMC* *status*	*GGA* *quality*	*GLL, VTG* *posMode*	*RMC, GNS* *posMode*
No position fix (at power-up, after losing satellite lock)	V	0	N	N
GNSS fix, but user limits exceeded	V	0	N	N
Dead reckoning fix, but user limits exceeded	V	6	E	E
Dead reckoning fix	A	6	E	E
RTK float	A	5	D	F
RTK fixed	A	4	D	R
2D GNSS fix	A	1 / 2	A / D	A / D
3D GNSS fix	A	1 / 2	A / D	A / D
Combined GNSS/dead reckoning fix	A	1 / 2	A / D	A / D
	See below (1)	See below (2)	See below (3)	See below (3)

(1) Possible values for *status*: V = Data invalid, A = Data valid

(2) Possible values for *quality*: 0 = No fix, 1 = Autonomous GNSS fix, 2 = Differential GNSS fix, 4 = RTK fixed, 5 = RTK float, 6 = Estimated/Dead reckoning fix

(3) Possible values for *posMode*: N = No fix, E = Estimated/Dead reckoning fix, A = Autonomous GNSS fix, D = Differential GNSS fix, F = RTK float, R = RTK fixed

Flags in NMEA 2.3 and above

Flags in NMEA 2.3 and above continued

NMEA Message *Field*	*GLL, RMC* *status*	*GGA* *quality*	*GSA* *navMode*	*GLL, VTG, RMC, GNS* *posMode*
NMEA Message *Field*	*GLL, RMC* *status*	*GGA* *quality*	*GSA* *navMode*	*GLL, VTG, RMC, GNS* *posMode*
No position fix (at power-up, after losing satellite lock)	V	0	1	N
GNSS fix, but user limits exceeded	V	0	1	N
Dead reckoning fix, but user limits exceeded	V	6	2	E
Dead reckoning fix	A	6	2	E
2D GNSS fix	A	1 / 2	2	A / D
3D GNSS fix	A	1 / 2	2	A / D
Combined GNSS/dead reckoning fix	A	1 / 2	3	A / D
	See below (1)	See below (2)	See below (3)	See below (4)

(1) Possible values for *status*: V = Data invalid, A = Data valid

(2) Possible values for *quality*: 0 = No fix, 1 = Autonomous GNSS fix, 2 = Differential GNSS fix, 4 = RTK fixed, 5 = RTK float, 6 = Estimated/Dead reckoning fix

(3) Possible values for *navMode*: 1 = No fix, 2 = 2D fix, 3 = 3D fix

(4) Possible values for *posMode*: N = No fix, E = Estimated/Dead reckoning fix, A = Autonomous GNSS fix, D = Differential GNSS fix, F = RTK float, R = RTK fixed

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.

3.1.7 Multi-GNSS considerations

Many applications which process NMEA messages assume that only a single GNSS is active. However, when multiple GNSS are configured, the NMEA specification requires the output to change in the following ways:

NMEA output for Multi-GNSS

Change	*Description*
Main Talker ID	The main Talker ID will be 'GN' (e.g. instead of 'GP' for a GPS receiver)
GSV Talker IDs	The GSV message reports the signal strength of the visible satellites. However, the Talker ID it uses is specific to the GNSS it is reporting information for, so for a multi-GNSS receiver it will not be the same as the main Talker ID. (e.g. other messages will be using the 'GN' Talker ID but the GSV message will use GNSS-sepcific Talker IDs)
Multiple GSA and GRS Messages	Multiple GSA and GRS messages are output for each fix, one for each GNSS. This may confuse applications which assume they are output only once per position fix (as is the case for a single GNSS receiver).

3.1.8 Output 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
```

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).

3.1.9 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**
11	**GGA**	0xF0 0x00	Global positioning system fix data
12	**GLL**	0xF0 0x01	Latitude and longitude, with time of position fix and status
13	**GNS**	0xF0 0x0D	GNSS fix data
14	**RMC**	0xF0 0x04	Recommended Minimum data
15	**VTG**	0xF0 0x05	Course over ground and Ground speed

3.2 Standard Messages

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

3.2.1 GGA

3.2.1.1 Global positioning system fix data

Message	**GGA**		
Description	**Global positioning system fix data**		
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1		
Type	Output Message		
Comment	**The output of this message is dependent on the currently selected datum (default: WGS84). The NMEA specification indicates that the GGA message is GPS specific. However, when the receiver is configured for multi-GNSS, the GGA message contents will be generated from the multi-GNSS solution. For multi-GNSS use, it is recommended that the `NMEA-GNS` message is used instead.** 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 ellipsoid 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)

GGA continued

Field No.	Name	Unit	Format	Example	Description
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
4	RTK fixed solution
5	RTK float solution
6	Estimated (DR) Fix

3.2.2 GLL

3.2.2.1 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 8 / u-blox M8 from protocol version 15 up to version 20.1		
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	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.

GLL continued

Field No.	*Name*	*Unit*	*Format*	*Example*	*Description*
7	posMode	-	character	A	Positioning mode, see position fix flags description. **NMEA v2.3 and above only**
8	cs	-	hexadecimal	*60	Checksum
9	<CR><LF>	-	character	-	Carriage return and line feed

3.2.3 GNS

3.2.3.1 GNSS fix data

Message	**GNS**		
Description	**GNSS fix data**		
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1		
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	16	

Message Structure:

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

Example:

```
$GPGNS,091547.00,5114.50897,N,00012.28663,W,AA,10,0.83,111.1,45.6,,,V*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 ellipsoid and mean sea level
11	diffAge	s	numeric	-	Age of differential corrections (blank when DGPS is not used)

GNS continued

Field No.	Name	Unit	Format	Example	Description
12	diffStation	-	numeric	-	ID of station providing differential corrections (blank when DGPS is not used)
13	navStatus	-	character	V	Navigational status indicator (V = Equipment is not providing navigational status information) **NMEA v4.1 and above only**
14	cs	-	hexadecimal	*71	Checksum
15	<CR><LF>	-	character	-	Carriage return and line feed

3.2.4 RMC

3.2.4.1 Recommended Minimum data

Message	**RMC**		
Description	**Recommended Minimum data**		
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1		
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	16	

Message Structure:

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

Example:

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

RMC continued

Field No.	Name	Unit	Format	Example	Description
10	mv	degrees	numeric	-	Magnetic variation value (blank - not supported)
11	mvEW	-	character	-	Magnetic variation E/W indicator (blank - not supported)
12	posMode	-	character	A	Mode Indicator, see position fix flags description **NMEA v2.3 and above only**
13	navStatus	-	character	V	Navigational status indicator (V = Equipment is not providing navigational status information) **NMEA v4.1 and above only**
14	cs	-	hexadecimal	*57	Checksum
15	<CR><LF>	-	character	-	Carriage return and line feed

3.2.5 VTG

3.2.5.1 Course over ground and Ground speed

Message	**VTG**	
Description	**Course over ground and Ground speed**	
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1	
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 **NMEA v2.3 and above only**

VTG continued

Field No.	Name	Unit	Format	Example	Description
10	cs	-	hexadecimal	*06	Checksum
11	<CR><LF>	-	character	-	Carriage return and line feed

4 UBX Protocol

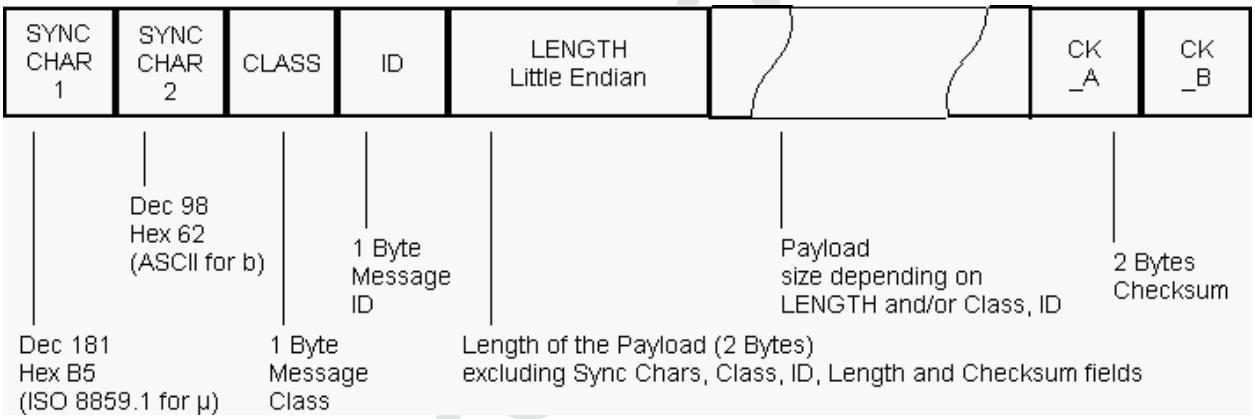
4.1 UBX Protocol Key Features

u-blox receivers support a u-blox proprietary protocol to communicate with 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)

4.2 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.

4.3 UBX Payload Definition Rules

4.3.1 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.

4.3.2 Reserved Elements

Some messages contain reserved fields or bits to allow for future expansion. The contents of these elements should be ignored in output messages and must be set to zero in input messages. Where a message is output and subsequently returned to the receiver as input message, reserved elements can either be explicitly set to zero or left with whatever value they were output with.

4.3.3 Undefined Values

The description of some fields provide specific meanings for specific values. For example, the field gnssId appears in many UBX messages and uses 0 to indicate GPS, 1 for SBAS and so on (see Satellite Numbering for details); however it is usually stored in a byte with far more possible values than the handful currently defined. All such undefined values are reserved for future expansion and therefore should not be used.

4.3.4 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 UBX-NAV-POSECEF. Referring to values is done by adding a dash and the name, e.g. UBX-NAV-POSECEF-X

4.3.5 Number Formats

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

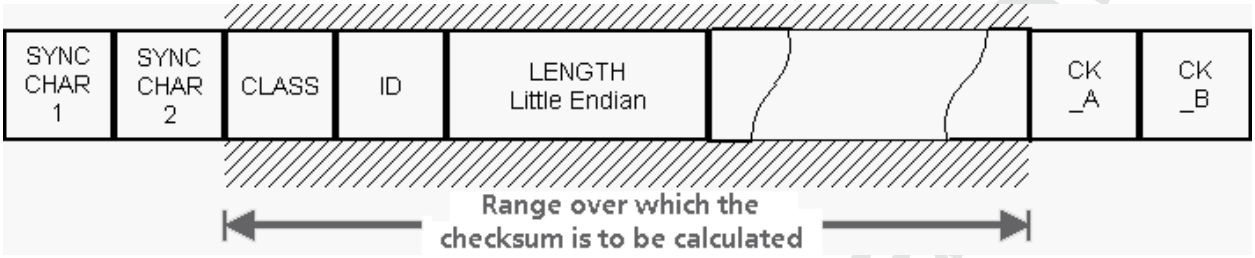
All floating point values are transmitted in IEEE754 single or double precision.

Variable Type Definitions

Short	*Type*	*Size (Bytes)*	*Comment*	*Min/Max*	*Resolution*
U1	Unsigned Char	1		0..255	1
RU1_3	Unsigned Char	1	binary floating point with 3 bit exponent, eeeb bbbb, (Value & 0x1F) << (Value >> 5)	0..(31*2^7) non-continuous	~ 2^(Value >> 5)
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			

4.4 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.

4.5 UBX Message Flow

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

4.5.1 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.

4.5.2 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.

The UBX protocol is designed so that messages can be polled by sending the message required to the receiver but without a payload (or with just a single parameter that identifies the poll request). The receiver then responds with the same message with the payload populated.

4.6 UBX Satellite Numbering

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 numbering similar 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 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*.

See Satellite Numbering Summary for a complete list of satellite numbers.

GNSS Identifiers

gnssId	*GNSS*
0	GPS
1	SBAS
2	Galileo
3	BeiDou
4	IMES
5	QZSS
6	GLONASS

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

u-blox designates GPS, Galileo, BeiDou and GLONASS as major GNSS, and the others as augmentation systems. These designations are described in the section on GNSS Types.

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

4.7 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 Messages: Position, Speed, Time, Acceleration, 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
CFG	0x06	Configuration Input Messages: Set Dynamic Model, Set DOP Mask, Set Baud Rate, etc.

All remaining class IDs are reserved.

4.8 UBX Messages Overview

Page	Mnemonic	Cls/ID	Length	Type	Description
UBX Class CFG				**Configuration Input Messages**	
21	**CFG-DGNSS**	0x06 0x70	4	Get/Set	DGNSS configuration
21	**CFG-MSG**	0x06 0x01	2	Poll Request	Poll a message configuration
22	**CFG-MSG**	0x06 0x01	8	Get/Set	Set Message Rate(s)
22	**CFG-MSG**	0x06 0x01	3	Get/Set	Set Message Rate
23	**CFG-NAV5**	0x06 0x24	36	Get/Set	Navigation Engine Settings
25	**CFG-NMEA**	0x06 0x17	4	Get/Set	NMEA protocol configuration (deprecated)
26	**CFG-NMEA**	0x06 0x17	12	Get/Set	NMEA protocol configuration V0 (deprecated)
29	**CFG-NMEA**	0x06 0x17	20	Get/Set	Extended NMEA protocol configuration V1
31	**CFG-PRT**	0x06 0x00	1	Poll Request	Polls the configuration for one I/O Port
32	**CFG-PRT**	0x06 0x00	20	Get/Set	Port Configuration for UART
35	**CFG-PRT**	0x06 0x00	20	Get/Set	Port Configuration for USB Port
37	**CFG-PRT**	0x06 0x00	20	Get/Set	Port Configuration for SPI Port
40	**CFG-PRT**	0x06 0x00	20	Get/Set	Port Configuration for DDC Port
42	**CFG-TMODE3**	0x06 0x71	40	Get/Set	Time Mode Settings 3
UBX Class INF				**Information Messages**	
45	**INF-WARNING**	0x04 0x01	0 + 1*N	Output	ASCII output with warning contents
UBX Class NAV				**Navigation Results Messages**	
46	**NAV-HPPOSECEF**	0x01 0x13	28	Periodic/Polled	High Precision Position Solution in ECEF
47	**NAV-HPPOSLLH**	0x01 0x14	36	Periodic/Polled	High Precision Geodetic Position Solution
48	**NAV-PVT**	0x01 0x07	92	Periodic/Polled	Navigation Position Velocity Time Solution
50	**NAV-RELPOSNED**	0x01 0x3C	40	Periodic/Polled	Relative Positioning Information in NED frame
52	**NAV-SAT**	0x01 0x35	8 + 12*numSvs	Periodic/Polled	Satellite Information
54	**NAV-STATUS**	0x01 0x03	16	Periodic/Polled	Receiver Navigation Status
56	**NAV-SVIN**	0x01 0x3B	40	Periodic/Polled	Survey-in data
UBX Class RXM				**Receiver Manager Messages**	
58	**RXM-RTCM**	0x02 0x32	8	Output	RTCM input status

4.9 UBX-CFG (0x06)

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

4.9.1 UBX-CFG-DGNSS (0x06 0x70)

4.9.1.1 DGNSS configuration

Message	**CFG-DGNSS**					
Description	**DGNSS configuration**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 20.01 up to version 20.1 (**only with High Precision GNSS product**)					
Type	Get/Set					
Comment	This message allows the user to configure the DGNSS configuration of the receiver.					
Message Structure	*Header*	*Class*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06	0x70	4	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`dgnssMode`	-	Specifies differential mode: 2: RTK float: No attempts are made to fix ambiguities. 3: RTK fixed: Ambiguities are fixed whenever possible.
1	U1[3]	-	`reserved1`	-	Reserved

4.9.2 UBX-CFG-MSG (0x06 0x01)

4.9.2.1 Poll a message configuration

Message	**CFG-MSG**					
Description	**Poll a message configuration**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1					
Type	Poll Request					
Comment	-					
Message Structure	*Header*	*Class*	*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

4.9.2.2 Set Message Rate(s)

Message	**CFG-MSG**					
Description	**Set Message Rate(s)**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1					
Type	Get/Set					
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*	*Class*	*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)

4.9.2.3 Set Message Rate

Message	**CFG-MSG**					
Description	**Set Message Rate**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1					
Type	Get/Set					
Comment	Set message rate configuration for the current port. See also section How to change between protocols.					
Message Structure	*Header*	*Class*	*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

4.9.3 UBX-CFG-NAV5 (0x06 0x24)

4.9.3.1 Navigation Engine Settings

Message	**CFG-NAV5**					
Description	**Navigation Engine Settings**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1					
Type	Get/Set					
Comment	See the Navigation Configuration Settings Description for a detailed description of how these settings affect receiver operation.					
Message Structure	*Header*	*Class*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06	0x24	36	*see below*	CK_A CK_B

Payload Contents:

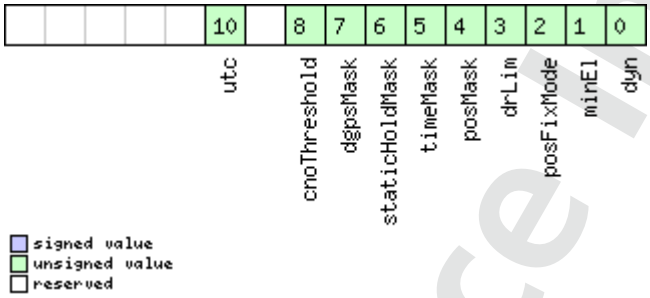
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	X2	-	`mask`	-	Parameters Bitmask. Only the masked parameters will be applied. (see graphic below)
2	U1	-	`dynModel`	-	Dynamic platform model: 0: portable 2: stationary 3: pedestrian 4: automotive 5: sea 6: airborne with <1g acceleration 7: airborne with <2g acceleration 8: airborne with <4g acceleration 9: wrist worn watch (not supported in protocol versions less than 18)
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	-	`dgnssTimeout`	s	DGNSS timeout
24	U1	-	`cnoThreshNumSVs`	-	Number of satellites required to have C/N0 above cnoThresh for a fix to be attempted

CFG-NAV5 continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
25	U1	-	`cnoThresh`	dBHz	C/N0 threshold for deciding whether to attempt a fix
26	U1[2]	-	`reserved1`	-	Reserved
28	U2	-	`staticHoldMaxDist`	m	Static hold distance threshold (before quitting static hold)
30	U1	-	`utcStandard`	-	UTC standard to be used: 0: Automatic; receiver selects based on GNSS configuration (see GNSS time bases). 3: UTC as operated by the U.S. Naval Observatory (USNO); derived from GPS time 6: UTC as operated by the former Soviet Union; derived from GLONASS time 7: UTC as operated by the National Time Service Center, China; derived from BeiDou time (not supported in protocol versions less than 16).
31	U1[5]	-	`reserved2`	-	Reserved

Bitfield mask

This graphic explains the bits of `mask`



Name	*Description*
`dyn`	Apply dynamic model settings
`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.
`cnoThreshold`	Apply CNO threshold settings (cnoThresh, cnoThreshNumSVs).
`utc`	Apply UTC settings. (not supported in protocol versions less than 16).

4.9.4 UBX-CFG-NMEA (0x06 0x17)

4.9.4.1 NMEA protocol configuration (deprecated)

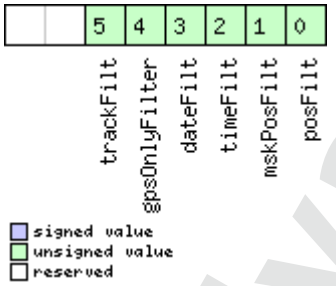
Message	**CFG-NMEA**					
Description	**NMEA protocol configuration (deprecated)**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1					
Type	Get/Set					
Comment	**This message version is provided for backwards compatibility only. Use the last version listed below instead (its fields are backwards compatible with this version, it just has extra fields defined).** 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*	*Class*	*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
2	U1	-	`numSV`	-	Maximum Number of SVs to report per TalkerId. 0: unlimited 8: 8 SVs 12: 12 SVs 16: 16 SVs
3	X1	-	`flags`	-	flags (see graphic below)

Bitfield filter

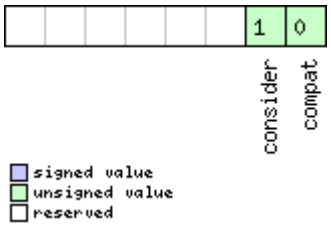
This graphic explains the bits of `filter`



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.

4.9.4.2 NMEA protocol configuration V0 (deprecated)

Message	**CFG-NMEA**					
Description	**NMEA protocol configuration V0 (deprecated)**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1					
Type	Get/Set					
Comment	**This message version is provided for backwards compatibility only. Use the last version listed below instead (its fields are backwards compatible with this version, it just has extra fields defined).** 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*	*Class*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06	0x17	12	*see below*	CK_A CK_B

Payload Contents:

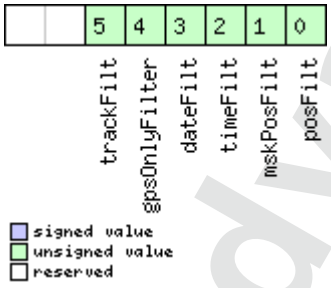
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	X1	-	`filter`	-	filter flags (see graphic below)
1	U1	-	`nmeaVersion`	-	0x23: NMEA version 2.3 0x21: NMEA version 2.1
2	U1	-	`numSV`	-	Maximum Number of SVs to report per TalkerId. 0: unlimited 8: 8 SVs 12: 12 SVs 16: 16 SVs
3	X1	-	`flags`	-	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)

CFG-NMEA continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
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 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' 4: Set main Talker ID to 'GA' 5: Set main Talker ID to 'GB'
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	-	`version`	-	Message version (set to 0 for this version)

Bitfield filter

This graphic explains the bits of `filter`



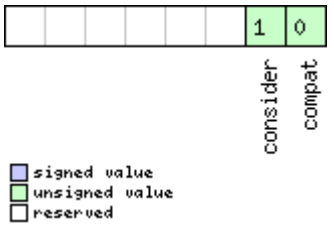
Name	*Description*
`posFilt`	Enable position output for failed or invalid fixes
`mskPosFilt`	Enable position output for invalid fixes
`timeFilt`	Enable time output for invalid times
`dateFilt`	Enable date output for invalid dates
`gpsOnlyFilter`	Restrict output to GPS satellites only

Bitfield filter Description continued

Name	*Description*
`trackFilt`	Enable COG output even if COG is frozen

Bitfield flags

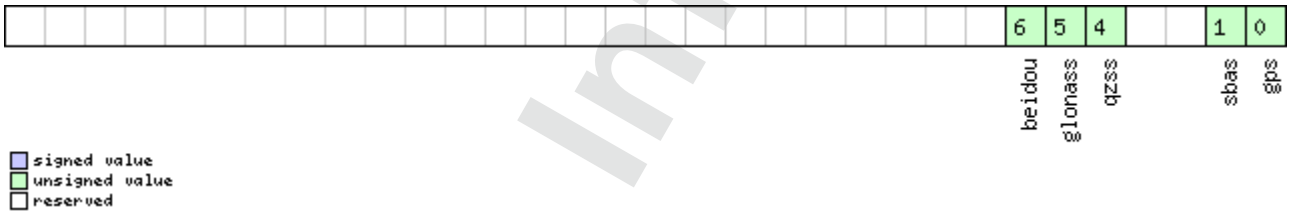
This graphic explains the bits of `flags`



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
`beidou`	Disable reporting of BeiDou satellites

4.9.4.3 Extended NMEA protocol configuration V1

Message	**CFG-NMEA**					
Description	**Extended NMEA protocol configuration V1**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1					
Type	Get/Set					
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*	*Class*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06	0x17	20	*see below*	CK_A CK_B

Payload Contents:

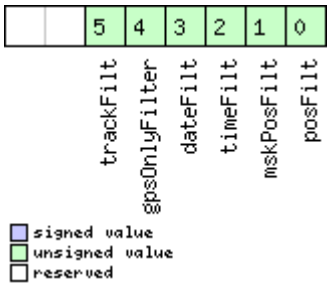
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	X1	-	`filter`	-	filter flags (see graphic below)
1	U1	-	`nmeaVersion`	-	0x41: NMEA version 4.1 0x40: NMEA version 4.0 0x23: NMEA version 2.3 0x21: NMEA version 2.1
2	U1	-	`numSV`	-	Maximum Number of SVs to report per TalkerId. 0: unlimited 8: 8 SVs 12: 12 SVs 16: 16 SVs
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 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' 4: Set main Talker ID to 'GA' 5: Set main Talker ID to 'GB'

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	-	`version`	-	Message version (set to 1 for this version)
12	CH[2]	-	`bdsTalkerId`	-	Sets the two characters that should be used for the BeiDou Talker ID If these are set to zero, the default BeiDou TalkerId will be used
14	U1[6]	-	`reserved1`	-	Reserved

Bitfield filter

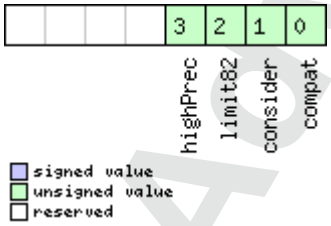
This graphic explains the bits of `filter`



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`



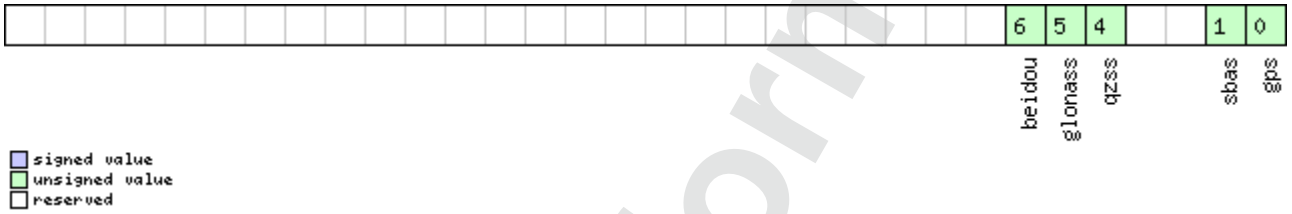
Name	*Description*

Bitfield flags Description continued

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.
`limit82`	enable strict limit to 82 characters maximum.
`highPrec`	enable high precision mode. This flag cannot be set in conjunction with either Compatibility Mode or Limit82 Mode. (not supported in protocol versions less than 20.01)

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
`beidou`	Disable reporting of BeiDou satellites

4.9.5 UBX-CFG-PRT (0x06 0x00)

4.9.5.1 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 8 / u-blox M8 from protocol version 15 up to version 20.1					
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*	*Class*	*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*
0	U1	-	`PortID`	-	Port Identifier Number (see the other versions of CFG-PRT for valid values)

4.9.5.2 Port Configuration for UART

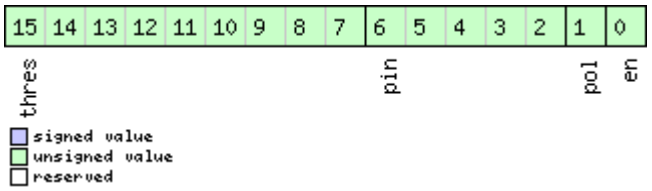
Message	**CFG-PRT**					
Description	**Port Configuration for UART**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1					
Type	Get/Set					
Comment	Several configurations can be concatenated to one input message. In this case the payload length can be a multiple of the normal length (see the other versions of CFG-PRT). Output messages from the module contain only one configuration unit. Note that this message can affect baud rate and other transmission parameters. Because there may be messages queued for transmission there may be uncertainty about which protocol applies to such messages. In addition a message currently in transmission may be corrupted by a protocol change. Host data reception paramaters may have to be changed to be able to receive future messages, including the acknowledge message resulting from the CFG-PRT message.					
Message Structure	*Header*	*Class*	*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	-	`reserved1`	-	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	Baud rate 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	U1[2]	-	`reserved2`	-	Reserved

Bitfield txReady

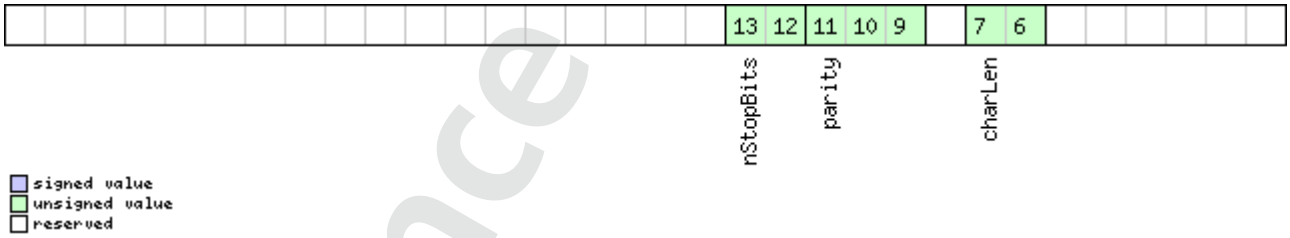
This graphic explains the bits of `txReady`



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`



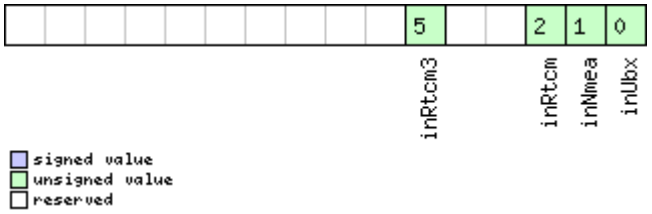
Name	*Description*
`charLen`	Character Length 00 5bit (not supported) 01 6bit (not supported) 10 7bit (supported only with parity) 11 8bit
`parity`	000 Even Parity 001 Odd Parity 10X No Parity X1X Reserved

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

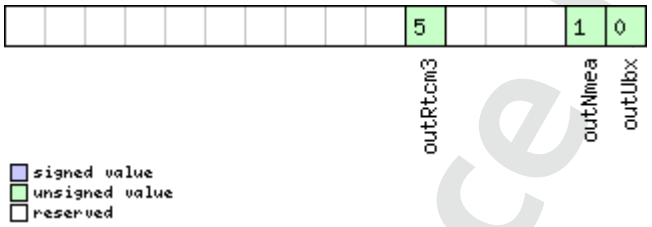
This graphic explains the bits of `inProtoMask`



Name	*Description*
inUbx	UBX protocol
inNmea	NMEA protocol
inRtcm	RTCM2 protocol
inRtcm3	RTCM3 protocol (not supported in protocol versions less than 20)

Bitfield outProtoMask

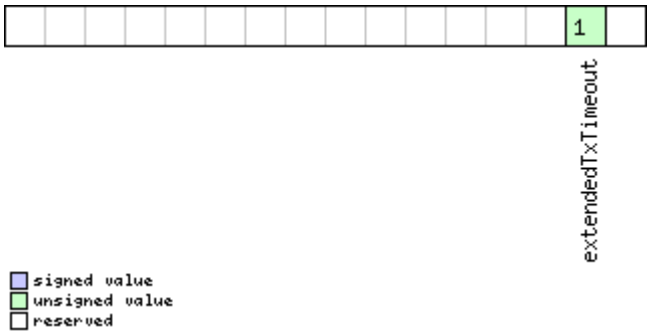
This graphic explains the bits of `outProtoMask`



Name	*Description*
outUbx	UBX protocol
outNmea	NMEA protocol
outRtcm3	RTCM3 protocol (not supported in protocol versions less than 20)

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. If not set the port will timoout if no activity for 1.5s regardless on the amount of allocated TX memory.

4.9.5.3 Port Configuration for USB Port

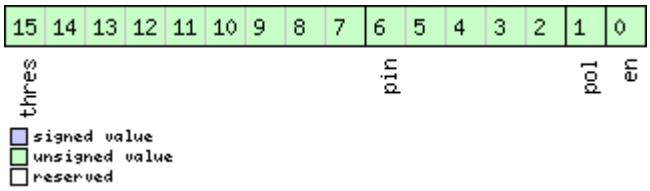
Message	**CFG-PRT**					
Description	**Port Configuration for USB Port**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1					
Type	Get/Set					
Comment	Several configurations can be concatenated to one input message. In this case the payload length can be 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*	*Class*	*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	-	`reserved1`	-	Reserved
2	X2	-	`txReady`	-	TX ready PIN configuration (see graphic below)
4	U1[8]	-	`reserved2`	-	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	U1[2]	-	`reserved3`	-	Reserved
18	U1[2]	-	`reserved4`	-	Reserved

Bitfield txReady

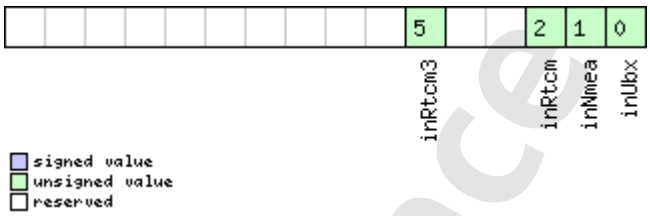
This graphic explains the bits of `txReady`



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 inProtoMask

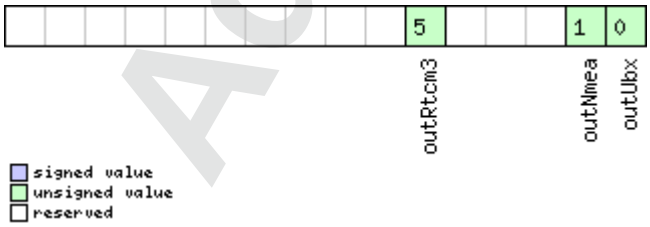
This graphic explains the bits of `inProtoMask`



Name	*Description*
`inUbx`	UBX protocol
`inNmea`	NMEA protocol
`inRtcm`	RTCM2 protocol
`inRtcm3`	RTCM3 protocol (not supported in protocol versions less than 20)

Bitfield outProtoMask

This graphic explains the bits of `outProtoMask`



Bitfield outProtoMask Description continued

Name	*Description*
Name	*Description*
`outUbx`	UBX protocol
`outNmea`	NMEA protocol
`outRtcm3`	RTCM3 protocol (not supported in protocol versions less than 20)

4.9.5.4 Port Configuration for SPI Port

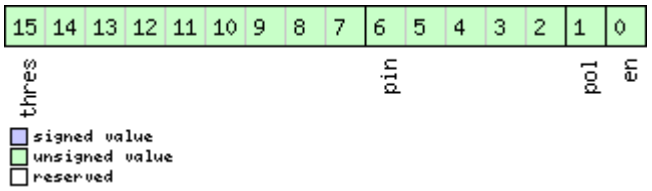
Message	**CFG-PRT**					
Description	**Port Configuration for SPI Port**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1					
Type	Get/Set					
Comment	Several configurations can be concatenated to one input message. In this case the payload length can be 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*	*Class*	*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	-	`reserved1`	-	Reserved
2	X2	-	`txReady`	-	TX ready PIN configuration (see graphic below)
4	X4	-	`mode`	-	SPI Mode Flags (see graphic below)
8	U1[4]	-	`reserved2`	-	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. (The bitfield inRtcm3 is not supported in protocol versions less than 20) (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. (The bitfield outRtcm3 is not supported in protocol versions less than 20) (see graphic below)
16	X2	-	`flags`	-	Flags bit mask (see graphic below)
18	U1[2]	-	`reserved3`	-	Reserved

Bitfield txReady

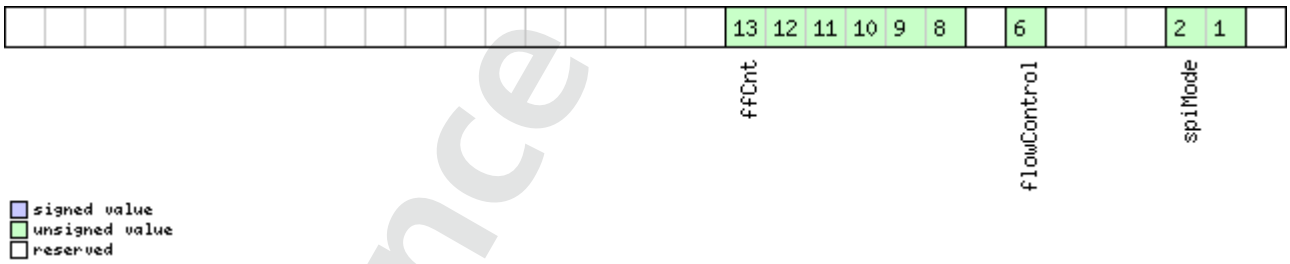
This graphic explains the bits of `txReady`



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

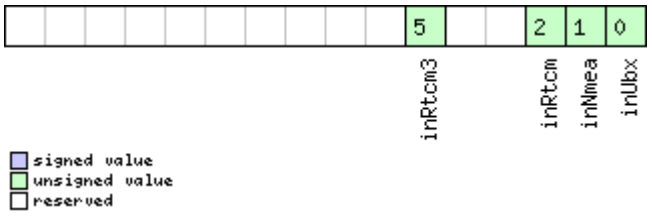
This graphic explains the bits of `mode`



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)-63

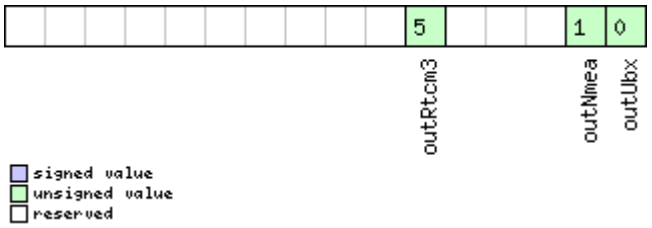
Bitfield inProtoMask

This graphic explains the bits of `inProtoMask`



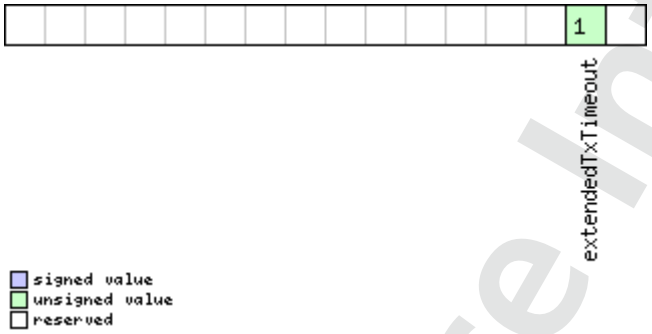
Bitfield outProtoMask

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.

4.9.5.5 Port Configuration for DDC Port

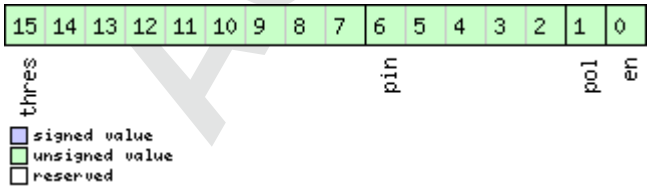
Message	**CFG-PRT**					
Description	**Port Configuration for DDC Port**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1					
Type	Get/Set					
Comment	Several configurations can be concatenated to one input message. In this case the payload length can be 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*	*Class*	*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	-	`reserved1`	-	Reserved
2	X2	-	`txReady`	-	TX ready PIN configuration (see graphic below)
4	X4	-	`mode`	-	DDC Mode Flags (see graphic below)
8	U1[4]	-	`reserved2`	-	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. (The bitfield inRtcm3 is not supported in protocol versions less than 20) (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. (The bitfield outRtcm3 is not supported in protocol versions less than 20) (see graphic below)
16	X2	-	`flags`	-	Flags bit mask (see graphic below)
18	U1[2]	-	`reserved3`	-	Reserved

Bitfield txReady

This graphic explains the bits of `txReady`

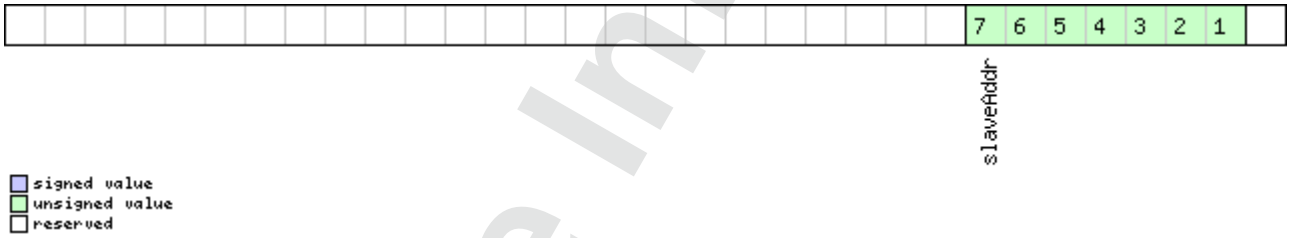


Bitfield txReady Description continued

Name	*Description*
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

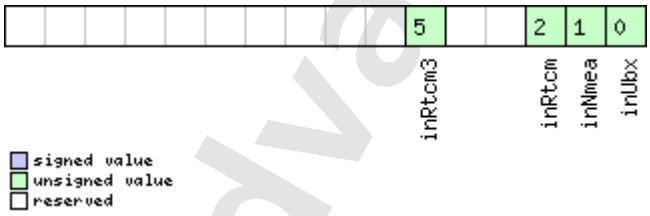
This graphic explains the bits of mode



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

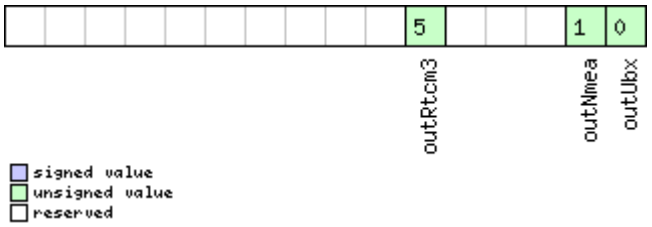
Bitfield inProtoMask

This graphic explains the bits of inProtoMask



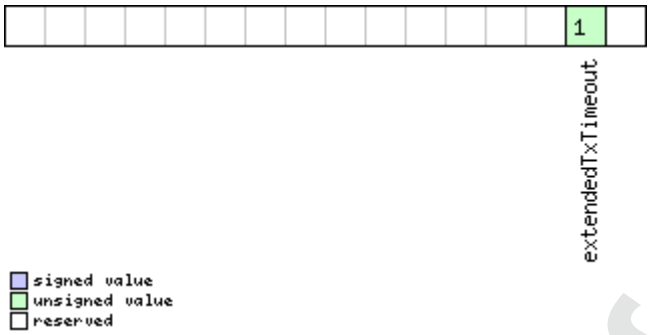
Bitfield outProtoMask

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.

4.9.6 UBX-CFG-TMODE3 (0x06 0x71)

4.9.6.1 Time Mode Settings 3

Message	**CFG-TMODE3**					
Description	**Time Mode Settings 3**					
Firmware	Supported on: • u-blox 8 / u-blox M8 with protocol version 20 (**only with High Precision GNSS product**)					
Type	Get/Set					
Comment	Configures the receiver to be in Time Mode. The position referred to in this message is that of the Antenna Reference Point (ARP). See the Time Mode Description for details.					
Message Structure	*Header*	*Class*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06	0x71	40	*see below*	CK_A CK_B

Payload Contents:

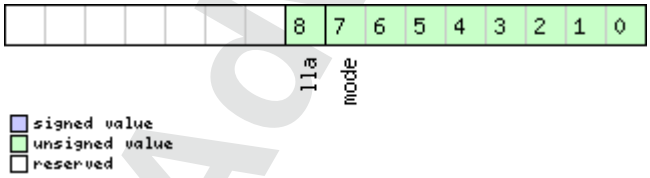
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`version`	-	Message version (0x00 for this version)
1	U1	-	`reserved1`	-	Reserved
2	X2	-	`flags`	-	Receiver mode flags (see graphic below)
4	I4	-	`ecefXOrLat`	cm_or_deg*1e-7	WGS84 ECEF X coordinate (or latitude) of the ARP position, depending on flags above

CFG-TMODE3 continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
8	I4	-	`ecefYOrLon`	cm_or_deg*1e-7	WGS84 ECEF Y coordinate (or longitude) of the ARP position, depending on flags above
12	I4	-	`ecefZOrAlt`	cm	WGS84 ECEF Z coordinate (or altitude) of the ARP position, depending on flags above
16	I1	-	`ecefXOrLatHP`	0.1_mm_or_deg *1e-9	High-precision WGS84 ECEF X coordinate (or latitude) of the ARP position, depending on flags above. Must be in the range -99..+99. The precise WGS84 ECEF X coordinate in units of cm, or the precise WGS84 ECEF latitude in units of 1e-7 degrees, is given by ecefXOrLat + (ecefXOrLatHP * 1e-2)
17	I1	-	`ecefYOrLonHP`	0.1_mm_or_deg *1e-9	High-precision WGS84 ECEF Y coordinate (or longitude) of the ARP position, depending on flags above. Must be in the range -99..+99. The precise WGS84 ECEF Y coordinate in units of cm, or the precise WGS84 ECEF longitude in units of 1e-7 degrees, is given by ecefYOrLon + (ecefYOrLonHP * 1e-2)
18	I1	-	`ecefZOrAltHP`	0.1_mm	High-precision WGS84 ECEF Z coordinate (or altitude) of the ARP position, depending on flags above. Must be in the range -99..+99. The precise WGS84 ECEF Z coordinate, or altitude coordinate, in units of cm is given by ecefZOrAlt + (ecefZOrAltHP * 1e-2)
19	U1	-	`reserved2`	-	Reserved
20	U4	-	`fixedPosAcc`	0.1_mm	Fixed position 3D accuracy
24	U4	-	`svinMinDur`	s	Survey-in minimum duration
28	U4	-	`svinAccLimit`	0.1_mm	Survey-in position accuracy limit
32	U1[8]	-	`reserved3`	-	Reserved

Bitfield flags

This graphic explains the bits of `flags`



Name	*Description*

Bitfield flags Description continued

Name	*Description*
mode	Receiver Mode: 0 Disabled 1 Survey In 2 Fixed Mode (true ARP position information required) 3-255 Reserved
lla	Position is given in LAT/LON/ALT (default is ECEF)

4.10 UBX-INF (0x04)

Information Messages: i.e. Printf-Style Messages, with IDs such as Error, Warning, Notice.
Messages in the INF class are used to output strings in a printf style from the firmware or application code. All INF messages have an associated type to indicate the kind of message.

4.10.1 UBX-INF-WARNING (0x04 0x01)

4.10.1.1 ASCII output with warning contents

Message	**INF-WARNING**					
Description	**ASCII output with warning contents**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1					
Type	Output					
Comment	This message has a variable length payload, representing an ASCII string.					
Message Structure	*Header*	*Class*	*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					

4.11 UBX-NAV (0x01)

Navigation Results Messages: i.e. Position, Speed, Time, Acceleration, Heading, DOP, SVs used.
Messages in the NAV class are used to output navigation data such as position, altitude and velocity in a number of formats. Additionally, status flags and accuracy figures are output. The messages are generated with the configured navigation/measurement rate.

4.11.1 UBX-NAV-HPPOSECEF (0x01 0x13)

4.11.1.1 High Precision Position Solution in ECEF

Message	**NAV-HPPOSECEF**					
Description	**High Precision Position Solution in ECEF**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 20.01 up to version 20.1					
Type	Periodic/Polled					
Comment	**See important comments concerning validity of position given in section Navigation Output Filters.** -					
Message Structure	*Header*	*Class*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01	0x13	28	*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[3]	-	`reserved1`	-	Reserved
4	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
8	I4	-	`ecefX`	cm	ECEF X coordinate
12	I4	-	`ecefY`	cm	ECEF Y coordinate
16	I4	-	`ecefZ`	cm	ECEF Z coordinate
20	I1	0.1	`ecefXHp`	mm	High precision component of ECEF X coordinate. Must be in the range of -99..+99. Precise coordinate in cm = ecefX + (ecefXHp * 1e-2).
21	I1	0.1	`ecefYHp`	mm	High precision component of ECEF Y coordinate. Must be in the range of -99..+99. Precise coordinate in cm = ecefY + (ecefYHp * 1e-2).
22	I1	0.1	`ecefZHp`	mm	High precision component of ECEF Z coordinate. Must be in the range of -99..+99. Precise coordinate in cm = ecefZ + (ecefZHp * 1e-2).
23	U1	-	`reserved2`	-	Reserved
24	U4	0.1	`pAcc`	mm	Position Accuracy Estimate

4.11.2 UBX-NAV-HPPOSLLH (0x01 0x14)

4.11.2.1 High Precision Geodetic Position Solution

Message	**NAV-HPPOSLLH**					
Description	**High Precision Geodetic Position Solution**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 20.01 up to version 20.1					
Type	Periodic/Polled					
Comment	**See important comments concerning validity of position given in section Navigation Output Filters.** This message outputs the Geodetic position with high precision in the currently selected ellipsoid. The default is the WGS84 Ellipsoid, but can be changed with the message CFG-DAT.					
Message Structure	*Header*	*Class*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01	0x14	36	*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[3]	-	`reserved1`	-	Reserved
4	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
8	I4	1e-7	`lon`	deg	Longitude
12	I4	1e-7	`lat`	deg	Latitude
16	I4	-	`height`	mm	Height above ellipsoid.
20	I4	-	`hMSL`	mm	Height above mean sea level
24	I1	1e-9	`lonHp`	deg	High precision component of longitude. Must be in the range -99..+99. Precise longitude in deg * 1e-7 = lon + (lonHp * 1e-2).
25	I1	1e-9	`latHp`	deg	High precision component of latitude. Must be in the range -99..+99. Precise latitude in deg * 1e-7 = lat + (latHp * 1e-2).
26	I1	0.1	`heightHp`	mm	High precision component of height above ellipsoid. Must be in the range -9..+9. Precise height in mm = height + (heightHp * 0.1).
27	I1	0.1	`hMSLHp`	mm	High precision component of height above mean sea level. Must be in range -9..+9. Precise height in mm = hMSL + (hMSLHp * 0.1)
28	U4	0.1	`hAcc`	mm	Horizontal accuracy estimate
32	U4	0.1	`vAcc`	mm	Vertical accuracy estimate

4.11.3 UBX-NAV-PVT (0x01 0x07)

4.11.3.1 Navigation Position Velocity Time Solution

Message	**NAV-PVT**					
Description	**Navigation Position Velocity Time Solution**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1					
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*	*Class*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01	0x07	92	*see below*	CK_A CK_B

Payload Contents:

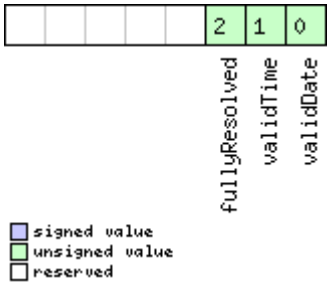
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	U2	-	`year`	y	Year (UTC)
6	U1	-	`month`	month	Month, range 1..12 (UTC)
7	U1	-	`day`	d	Day of month, range 1..31 (UTC)
8	U1	-	`hour`	h	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: 0: no fix 1: dead reckoning only 2: 2D-fix 3: 3D-fix 4: GNSS + dead reckoning combined 5: time only fix
21	X1	-	`flags`	-	Fix status flags (see graphic below)
22	X1	-	`flags2`	-	Additional flags (see graphic below)
23	U1	-	`numSV`	-	Number of satellites used in Nav Solution
24	I4	1e-7	`lon`	deg	Longitude
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)

NAV-PVT continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
64	I4	1e-5	`headMot`	deg	Heading of motion (2-D)
68	U4	-	`sAcc`	mm/s	Speed accuracy estimate
72	U4	1e-5	`headAcc`	deg	Heading accuracy estimate (both motion and vehicle)
76	U2	0.01	`pDOP`	-	Position DOP
78	U1[6]	-	`reserved1`	-	Reserved
84	I4	1e-5	`headVeh`	deg	Heading of vehicle (2-D)
88	U1[4]	-	`reserved2`	-	Reserved

Bitfield valid

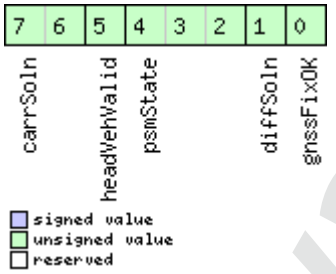
This graphic explains the bits of `valid`



Name	*Description*
`validDate`	1 = valid UTC Date (see Time Validity section for details)
`validTime`	1 = valid UTC Time of Day (see Time Validity section for details)
`fullyResolved`	1 = UTC Time of Day has been fully resolved (no seconds uncertainty)

Bitfield flags

This graphic explains the bits of `flags`



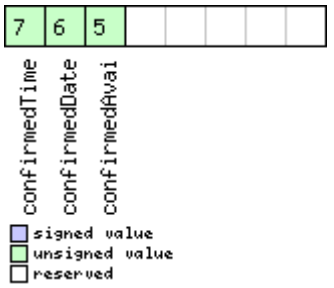
Name	*Description*
`gnssFixOK`	1 = valid fix (i.e within DOP & accuracy masks)
`diffSoln`	1 = differential corrections were applied
`psmState`	Power Save Mode state (see Power Management): 0: PSM is not active 1: Enabled (an intermediate state before Acquisition state 2: Acquisition 3: Tracking 4: Power Optimized Tracking 5: Inactive

Bitfield flags Description continued

Name	*Description*
`headVehValid`	1 = heading of vehicle is valid
`carrSoln`	Carrier phase range solution status: 0: no carrier phase range solution 1: float solution (no fixed integer carrier phase measurements have been used to calculate the solution) 2: fixed solution (one or more fixed integer carrier phase range measurements have been used to calculate the solution) (not supported in protocol versions less than 20)

Bitfield flags2

This graphic explains the bits of `flags2`



Name	*Description*
`confirmedAvai`	1 = information about UTC Date and Time of Day validity confirmation is available (see Time Validity section for details) (This flag is always unset for in protocol versions less than 19)
`confirmedDate`	1 = UTC Date validity could be confirmed (see Time Validity section for details)
`confirmedTime`	1 = UTC Time of Day could be confirmed (see Time Validity section for details)

4.11.4 UBX-NAV-RELPOSNED (0x01 0x3C)

4.11.4.1 Relative Positioning Information in NED frame

Message	**NAV-RELPOSNED**					
Description	**Relative Positioning Information in NED frame**					
Firmware	Supported on: • u-blox 8 / u-blox M8 with protocol version 20 (**only with High Precision GNSS product**)					
Type	Periodic/Polled					
Comment	**The NED frame is defined as the local topological system at the reference station. The relative position vector components in this message, along with their associated accuracies, are given in that local topological system** This message contains the relative position vector from the Reference Station to the Rover, including accuracy figures, in the local topological system defined at the reference station					
Message Structure	*Header*	*Class*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01	0x3C	40	*see below*	CK_A CK_B

Payload Contents:

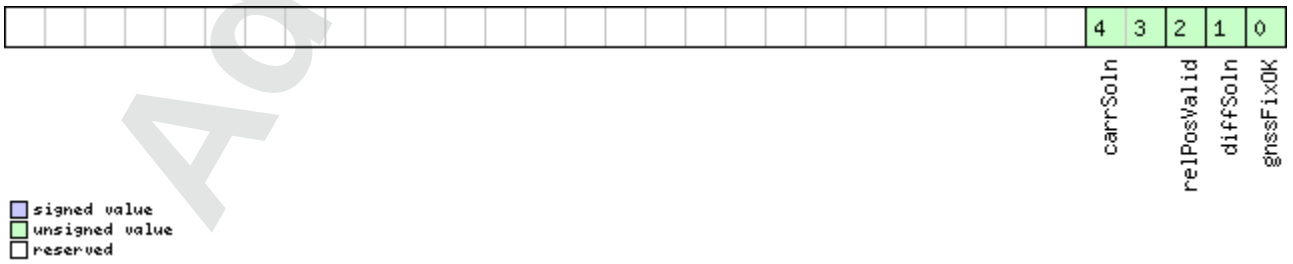
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`version`	-	Message version (0x00 for this version)
1	U1	-	`reserved1`	-	Reserved

NAV-RELPOSNED continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
2	U2	-	`refStationId`	-	Reference Station ID. Must be in the range 0.. 4095
4	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
8	I4	-	`relPosN`	cm	North component of relative position vector
12	I4	-	`relPosE`	cm	East component of relative position vector
16	I4	-	`relPosD`	cm	Down component of relative position vector
20	I1	-	`relPosHPN`	0.1_mm	High-precision North component of relative position vector. Must be in the range -99 to +99. The full North component of the relative position vector, in units of cm, is given by relPosN + (relPosHPN * 1e-2)
21	I1	-	`relPosHPE`	0.1_mm	High-precision East component of relative position vector. Must be in the range -99 to +99. The full East component of the relative position vector, in units of cm, is given by relPosE + (relPosHPE * 1e-2)
22	I1	-	`relPosHPD`	0.1_mm	High-precision Down component of relative position vector. Must be in the range -99 to +99. The full Down component of the relative position vector, in units of cm, is given by relPosD + (relPosHPD * 1e-2)
23	U1	-	`reserved2`	-	Reserved
24	U4	-	`accN`	0.1_mm	Accuracy of relative position North component
28	U4	-	`accE`	0.1_mm	Accuracy of relative position East component
32	U4	-	`accD`	0.1_mm	Accuracy of relative position Down component
36	X4	-	`flags`	-	Flags (see graphic below)

Bitfield flags

This graphic explains the bits of `flags`



Name	*Description*

Bitfield flags Description continued

Name	*Description*
gnssFixOK	A valid fix (i.e within DOP & accuracy masks)
diffSoln	1 if differential corrections were applied
relPosValid	1 if relative position components and accuracies are valid
carrSoln	Carrier phase range solution status: 0 = No carrier phase range solution 1 = Float solution. No fixed integer carrier phase measurements have been used to calculate the solution 2 = Fixed solution. One or more fixed integer carrier phase range measurements have been used to calculate the solution

4.11.5 UBX-NAV-SAT (0x01 0x35)

4.11.5.1 Satellite Information

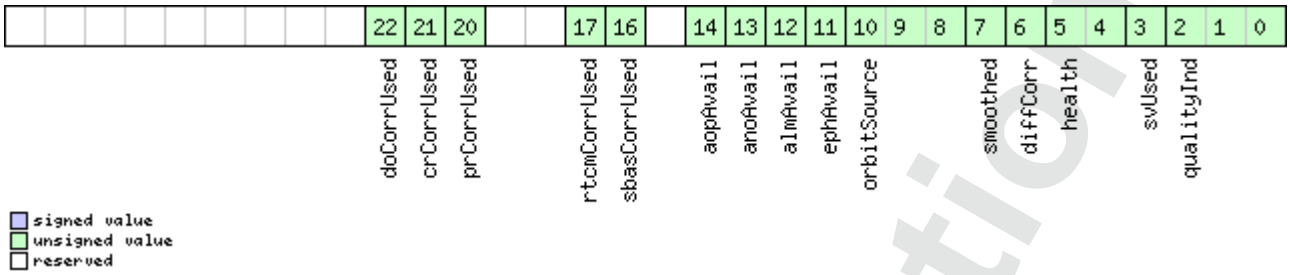
Message	**NAV-SAT**					
Description	**Satellite Information**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1					
Type	Periodic/Polled					
Comment	This message displays information about SVs which are either known to be visible or currently tracked by the receiver.					
Message Structure	*Header*	*Class*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01	0x35	8 + 12*numSvs	*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	-	version	-	Message version (1 for this version)
5	U1	-	numSvs	-	Number of satellites
6	U1[2]	-	reserved1	-	Reserved
Start of repeated block (numSvs times)					
8 + 12*N	U1	-	gnssId	-	GNSS identifier (see Satellite numbering) for assignment
9 + 12*N	U1	-	svId	-	Satellite identifier (see Satellite numbering) for assignment
10 + 12*N	U1	-	cno	dBHz	Carrier to noise ratio (signal strength)
11 + 12*N	I1	-	elev	deg	Elevation (range: +/-90), unknown if out of range
12 + 12*N	I2	-	azim	deg	Azimuth (range 0-360), unknown if elevation is out of range
14 + 12*N	I2	0.1	prRes	m	Pseudo range residual
16 + 12*N	X4	-	flags	-	Bitmask (see graphic below)
End of repeated block					

Bitfield flags

This graphic explains the bits of `flags`



Name	*Description*
`qualityInd`	Signal quality indicator: 0: no signal 1: searching signal 2: signal aquired 3: signal detected but unusable 4: code locked and time synchronized 5, 6, 7: code and carrier locked and time synchronized Note: Since IMES signals are not time synchronized, a channel tracking an IMES signal can never reach a quality indicator value of higher than 3.
`svUsed`	1 = SV is currently being used for navigation
`health`	SV health flag: 0: unknown 1: healthy 2: unhealthy
`diffCorr`	1 = differential correction data is available for this SV
`smoothed`	1 = carrier smoothed pseudorange used
`orbitSource`	Orbit source: 0: no orbit information is available for this SV 1: ephemeris is used 2: almanac is used 3: AssistNow Offline orbit is used 4: AssistNow Autonomous orbit is used 5, 6, 7: other orbit information is used
`ephAvail`	1 = ephemeris is available for this SV
`almAvail`	1 = almanac is available for this SV
`anoAvail`	1 = AssistNow Offline data is available for this SV
`aopAvail`	1 = AssistNow Autonomous data is available for this SV
`sbasCorrUsed`	1 = SBAS corrections have been used for this SV
`rtcmCorrUsed`	1 = RTCM corrections have been used for this SV
`prCorrUsed`	1 = Pseudorange corrections have been used for this SV
`crCorrUsed`	1 = Carrier range corrections have been used for this SV
`doCorrUsed`	1 = Range rate (Doppler) corrections have been used for this SV

4.11.6 UBX-NAV-STATUS (0x01 0x03)

4.11.6.1 Receiver Navigation Status

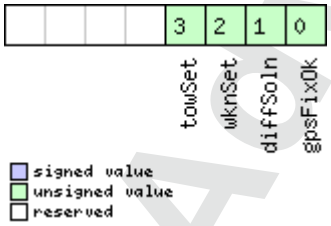
Message	**NAV-STATUS**					
Description	**Receiver Navigation Status**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 20.1					
Type	Periodic/Polled					
Comment	**See important comments concerning validity of position and velocity given in section Navigation Output Filters.** -					
Message Structure	*Header*	*Class*	*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`	ms	Time to first fix (millisecond time tag)
12	U4	-	`msss`	ms	Milliseconds since Startup / Reset

Bitfield flags

This graphic explains the bits of `flags`



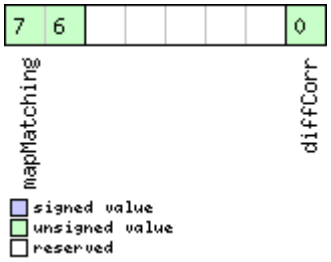
Name	*Description*
`gpsFixOk`	1 = position and velocity valid and within DOP and ACC Masks, see also important comments in section Navigation Output Filters.

Bitfield flags Description continued

Name	*Description*
`diffSoln`	1 = differential corrections were applied
`wknSet`	1 = Week Number valid (see Time Validity section for details)
`towSet`	1 = Time of Week valid (see Time Validity section for details)

Bitfield fixStat

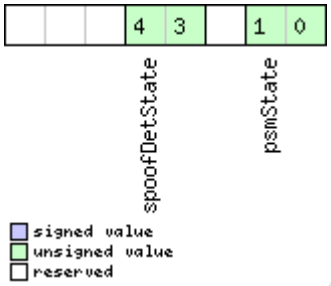
This graphic explains the bits of `fixStat`



Name	*Description*
`diffCorr`	1 = differential corrections available
`mapMatching`	map matching status: 00: none 01: valid but not used, i.e. map matching data was received, but was too old 10: valid and used, map matching data was applied 11: valid and used, map matching data was applied. In case of sensor unavailability map matching data enables dead reckoning. This requires map matched latitude/longitude or heading data.

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

Bitfield flags2 Description continued

Name	*Description*
`spoofDetState`	Spoofing detection state (not supported in protocol versions less than 18) 0: Unknown or deactivated 1: No spoofing indicated 2: Spoofing indicated 3: Multiple spoofing indications Note that the spoofing state value only reflects the dector state for the current navigation epoch. As spoofing can be detected most easily at the transition from real signal to spoofing signal, this is also where the detector is triggered the most. I.e. a value of *1 - No spoofing indicated* does not mean that the receiver is not spoofed, it simply states that the detector was not triggered in this epoch.

4.11.7 UBX-NAV-SVIN (0x01 0x3B)

4.11.7.1 Survey-in data

Message	**NAV-SVIN**					
Description	**Survey-in data**					
Firmware	Supported on: • u-blox 8 / u-blox M8 with protocol version 20 (**only with High Precision GNSS product**)					
Type	Periodic/Polled					
Comment	This message contains information about survey-in parameters.					
Message Structure	*Header*	*Class*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01	0x3B	40	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`version`	-	Message version (0x00 for this version)
1	U1[3]	-	`reserved1`	-	Reserved
4	U4	-	`iTOW`	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
8	U4	-	`dur`	s	Passed survey-in observation time
12	I4	-	`meanX`	cm	Current survey-in mean position ECEF X coordinate
16	I4	-	`meanY`	cm	Current survey-in mean position ECEF Y coordinate
20	I4	-	`meanZ`	cm	Current survey-in mean position ECEF Z coordinate
24	I1	-	`meanXHP`	0.1_mm	Current high-precision survey-in mean position ECEF X coordinate. Must be in the range -99.. +99. The current survey-in mean position ECEF X coordinate, in units of cm, is given by meanX + (0.01 * meanXHP)

NAV-SVIN continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
25	I1	-	`meanYHP`	0.1_mm	Current high-precision survey-in mean position ECEF Y coordinate. Must be in the range -99..+99. The current survey-in mean position ECEF Y coordinate, in units of cm, is given by meanY + (0.01 * meanYHP)
26	I1	-	`meanZHP`	0.1_mm	Current high-precision survey-in mean position ECEF Z coordinate. Must be in the range -99..+99. The current survey-in mean position ECEF Z coordinate, in units of cm, is given by meanZ + (0.01 * meanZHP)
27	U1	-	`reserved2`	-	Reserved
28	U4	-	`meanAcc`	0.1_mm	Current survey-in mean position accuracy
32	U4	-	`obs`	-	Number of position observations used during survey-in
36	U1	-	`valid`	-	Survey-in position validity flag, 1 = valid, otherwise 0
37	U1	-	`active`	-	Survey-in in progress flag, 1 = in-progress, otherwise 0
38	U1[2]	-	`reserved3`	-	Reserved

4.12 UBX-RXM (0x02)

Receiver Manager Messages: i.e. Satellite Status, RTC Status.
Messages in the RXM class are used to output status and result data from the Receiver Manager. The output rate is not bound to the navigation/measurement rate and messages can also be generated on events.

4.12.1 UBX-RXM-RTCM (0x02 0x32)

4.12.1.1 RTCM input status

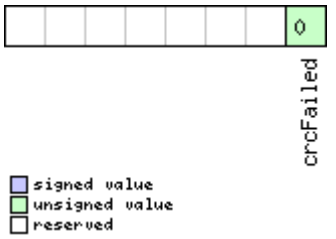
Message	**RXM-RTCM**					
Description	**RTCM input status**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 20.01 up to version 20.1					
Type	Output					
Comment	Output upon processing of an RTCM input message					
Message Structure	*Header*	*Class*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x02	0x32	8	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`version`	-	Message version (0x02 for this version)
1	X1	-	`flags`	-	RTCM input status flags (see graphic below)
2	U1[2]	-	`reserved1`	-	Reserved
4	U2	-	`refStation`	-	Reference station ID
6	U2	-	`msgType`	-	Message type

Bitfield flags

This graphic explains the bits of `flags`



Name	*Description*
`crcFailed`	0 when RTCM message received and passed CRC check, 1 when failed in which case refStation and msgType might be corrupted and misleading

5 RTCM Protocol

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. The RTCM protocol specification is available from http://www.rtcm.org.

5.1 RTCM2

5.1.1 Introduction

This feature is not supported with the HPG 1.20 firmware.

5.2 RTCM3

(Note: the RTCM3 protocol is not supported in protocol versions less than 20).

5.2.1 Introduction

This feature is only applicable to GPS, GLONASS or BeiDou operation.

This feature supports carrier phase differential positioning.

For effective differential positioning accuracy, it is necessary that the reference station antenna is situated in a low multipath environment with an unobstructed view of the sky and continuous phase lock on all visible satellites.

RTCM3 messages can also be transmitted through NTRIP (Networked Transport of RTCM via Internet Protocol). u-center incorporates an NTRIP client and an NTRIP server/caster.

5.2.2 Supported Messages

The following RTCM 3.2 input messages are supported:

Supported RTCM 3.2 Input Messages

Message Type	*Description*
1001	L1-only GPS RTK observables
1002	Extended L1-only GPS RTK observables
1003	L1/L2 GPS RTK observables
1004	Extended L1/L2 GPS RTK observables
1005	Stationary RTK reference station ARP
1006	Stationary RTK reference station ARP with antenna height
1007	Antenna descriptor
1009	L1-only GLONASS RTK observables
1010	Extended L1-only GLONASS RTK observables
1011	L1/L2 GLONASS RTK observables
1012	Extended L1/L2 GLONASS RTK observables
1075	GPS MSM5
1077	GPS MSM7
1085	GLONASS MSM5
1087	GLONASS MSM7
1125	BeiDou MSM5

Supported RTCM 3.2 Input Messages continued

Message Type	*Description*
1127	BeiDou MSM7

The following RTCM 3.2 output messages are supported:

Supported RTCM 3.2 Output Messages

Message Type	*Description*
1005	Stationary RTK reference station ARP
1077	GPS MSM7
1087	GLONASS MSM7
1127	BeiDou MSM7

5.2.3 Configuration

The configuration of the RTK rover and reference station is explained in the RTK Mode Configuration section.

The RTCM3 protocol can be disabled/enabled on communication interfaces by means of the UBX-CFG-PRT message. By default, RTCM3 is enabled.

The configuration of the RTCM3 correction stream must be done according to the following rules:

- The RTCM3 correction stream must contain only one reference station message (1005 or 1006) in addition to the GPS, GLONASS or BeiDou observable messages.
- All observable messages must be broadcast at the same rate.
- The reference station message does not need to be broadcast at the same rate as the observable messages but the rover will not be able to compute its position until it has received a valid reference station message.
- The reference station ID field in the GPS, GLONASS or BeiDou observable messages must be consistent with the reference station ID field in the reference station message otherwise the rover will not be able to compute its position.
- The RTCM3 correction stream should only contain one type of observable messages per constellation.
- When using GPS and GLONASS or GPS and BeiDou, both constellations should use the same type of observable messages.
- If the receiver uses several ports, they must all have the same RTCM configuration.

The time after which old RTCM data will be discarded can be specified using the dgnssTimeout field in UBX-CFG-NAV5.

5.2.4 Output

RTK Rover Mode will result in following modified output:

- NMEA-GGA: The quality field will be 4 for RTK fixed and 5 for RTK float (see NMEA Positon Fix Flags). The age of differential corrections and reference station ID will be set.
- NMEA-GLL, NMEA-VTG: The posMode indicator will be D for RTK float and RTK fixed (see NMEA Positon Fix Flags).
- NMEA-RMC, NMEA-GNS: The posMode indicator will be F for RTK float and R for RTK fixed (see NMEA Positon Fix Flags).
- UBX-NAV-PVT: The carrSoln flag will be set to 1 for RTK float and 2 for RTK fixed.
- UBX-NAV-SAT: The diffCorr flag will be set for satellites with valid RTCM data. The rtcmCorrUsed, prCorrUsed and cpCorrUsed flags will be set for satellites for which the RTCM corrections have been applied.
- UBX-NAV-STATUS: The diffSoln flag will be set; the diffCorr flag will be set.

- If the baseline exceeds 10km and a message type 1005 or 1006 is received, a UBX-INF-WARNING will be output, e.g. "WARNING: DGNSS baseline big: 12.7km"

5.2.5 Restrictions

The RTK solution will only include range measurements from signals for which RTCM3 corrections were provided. This is because the navigation algorithms cannot mix corrected with uncorrected range measurements.

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 GNSS receiver.

Revision History

Revision	*Date*	*Name*	*Status / Comments*
R01	22 February 2016	mfre	Addendum for HPG 1.00
R02	06 May 2016	mfre	Update for HPG 1.10
R03	12 September 2016	mfre	Update for HPG 1.20

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 Australia:
Phone: +61 2 8448 2016
E-mail: info_anz@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 (Chongqing):
Phone: +86 23 6815 1588
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office China (Shanghai):
Phone: +86 21 6090 4832
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 80 4050 9200
E-mail: info_in@u-blox.com
Support: support_in@u-blox.com

Regional Office Japan (Osaka):
Phone: +81 6 6941 3660
E-mail: info_jp@u-blox.com
Support: support_jp@u-blox.com

Regional Office Japan (Tokyo):
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