u-blox 5
NMEA, UBX Protocol Specification

u-blox 5 GNSS Receiver

Public Release

u-blox AG
Zürcherstrasse 68
8800 Thalwil
Switzerland
www.u-blox.com

Phone +41 44 722 7444
Fax +41 44 722 7447
info@u-blox.com

Specification

your position is our focus

Title	NMEA, UBX Protocol Specification	
Subtitle	u-blox 5 GNSS Receiver	Public Release
Doc Type	Specification	
Doc Id	GPS.G5-X-07036-D	

Revision	**Date**	**Author**	**Status / Comment**
29328	12 Aug 2008	EF	Draft

For most recent documents, please visit www.u-blox.com

Receiver Description

Serial Communication Ports Description

The u-blox 5 positioning technology comes with a highly flexible communication interface. It supports both the NMEA and the proprietary UBX protocol. It is truly multi-port and multi-protocol capable. Each protocol (UBX, NMEA) can be assigned to several ports at the same time (multi-port capability) with individual settings (e.g. baud rate, messages enabled, etc.) for each port. It is even possible to assign more than one protocol (e.g. UBX protocol and NMEA at the same time) to a single port (multi-protocol capability), which is particularly useful for debugging purposes.

The UBX and/or NMEA protocol must be activated to get a message on a port using the UBX proprietary message UBX-CFG-PRT, which also allows to change port-specific settings (baud rate, address etc.). See CFG-MSG for a description of the mechanism of enabling and disabling messages.

UART Ports

The receivers feature one or two universal asynchronous receiver/transmitter (UART) ports that can be used to transmit GPS measurements, monitor status information and configure the receiver. The availability of the second port depends on the type of module or chip set (see our online product selector matrix for modules and chip sets).

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

Possible UART Interface Configurations

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

If too much data is being configured for a certain port's bandwidth (e.g. all UBX messages shall be output on a UART port with a baud rate of 9600), the buffer will fill up. Once the buffer's space is exceeded, the receiver will deactivate messages automatically.

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

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

USB Port

The receivers feature one USB (Universal Serial Bus) port, depending on the type of module or chip set (see our online product selector matrix for modules and chip sets). This port can be used not only for communication purposes, but also to power the GPS receiver.

The USB interface supports two different power modes:

- In the *Self Powered Mode* the receiver is powered by its own power supply. **VDDUSB** is used to detect the availability of the USB port, i.e. whether the the receiver is connected to a USB host.
- In the *Bus Powered Mode* the device is powered by the USB bus, therefore no additional power supply is needed. The default maximum current that can be drawn by the receiver is 120mA in that mode. See CFG-USB for a description on how to change this maximum. Configuring the Bus Powered Mode implies that the device enters a low power state with disabled GPS functionality when the host suspends the device, e.g. when the host is put into stand-by mode.

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

DDC Port

A DDC Bus (Display Data Channel) is implemented, which is a 2-wire communication interface compatible with the I2C standard (Inter-Integrated Circuit). Its availability is depending on the type of module or chip set (see our online product selector matrix for modules and chip sets).

In contrast to all other interfaces, the DDC is not able to communicate in full-duplex mode, i.e. TX and RX are mutually exclusive. u-blox 5 acts as a slave in the communication setup, therefore it cannot initiate data transfers on its own. The master provides the data clock, therefore master and slave don't need to be configured to use the same baud rate. Moreover, a baud rate setting is not applicable for the slave.

The baud rate clock provided by the master must not exceed 100kHz

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

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

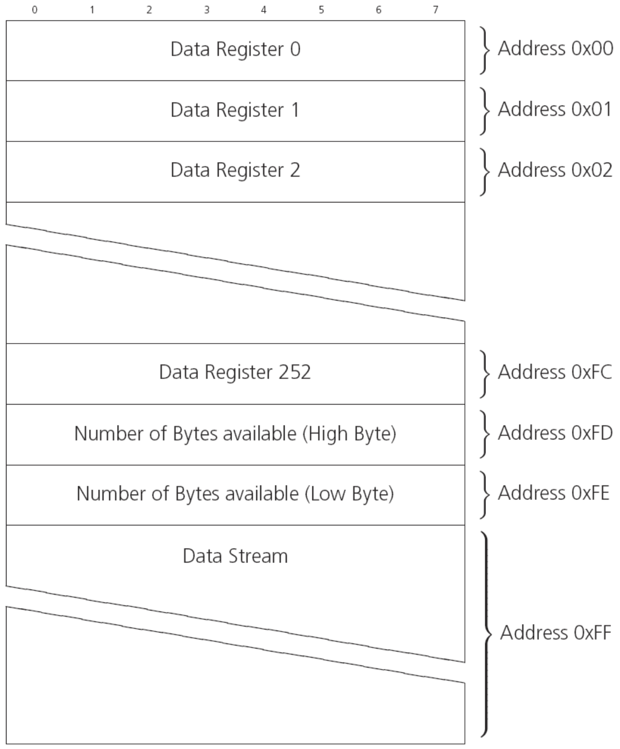
If no data is polled for an extended period, the receiver temporarily stops writing data to the output buffer to prevent overflowing.

Read Access

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

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

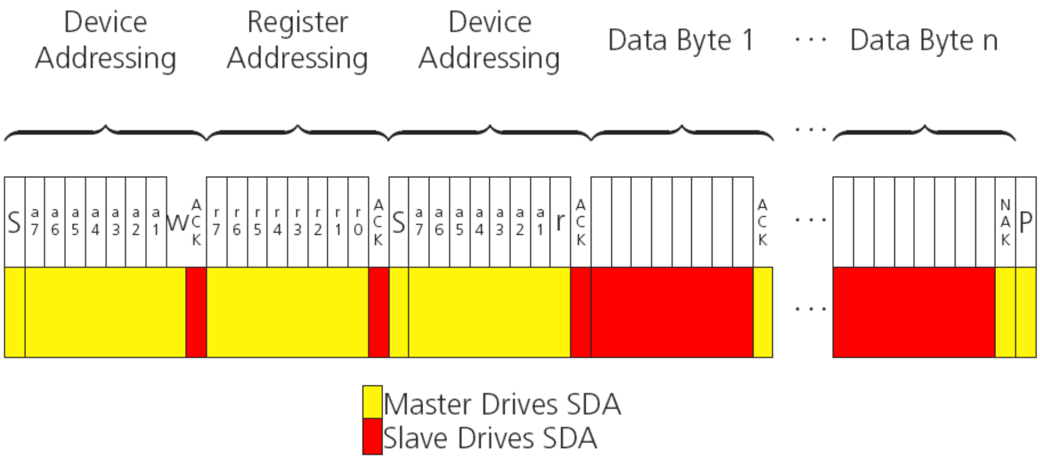
DDC Register Layout



Random Read Access

Random read operations allow the master to access any register in a random manner. To perform this type of read operation, first the register address to read from must be written to the receiver (see Figure *DDC Random Read Access*). Following the start condition from the master, the 7-bit device address including the `RW` bit (which is a logic low for write access) are clocked onto the bus by the master transmitter. The receiver answers with an acknowledge (logic low) to indicate that it is responsible for the given address. Next, the 8-bit address of the register to be read must be written to the bus. Following the receiver's acknowledge, the master again triggers a start condition and writes the device address, but this time the `RW` bit is a logic high to initiate the read access. Now, the master can read 1 to `N` bytes from the receiver, generating a not-acknowledge and a stop condition after the last byte being read. After every byte being read, the internal address counter is incremented by one, saturating at 0xFF. This saturation means, that, after having read all registers coming after the initially set register address, the raw message stream can be read.

DDC Random Read Access

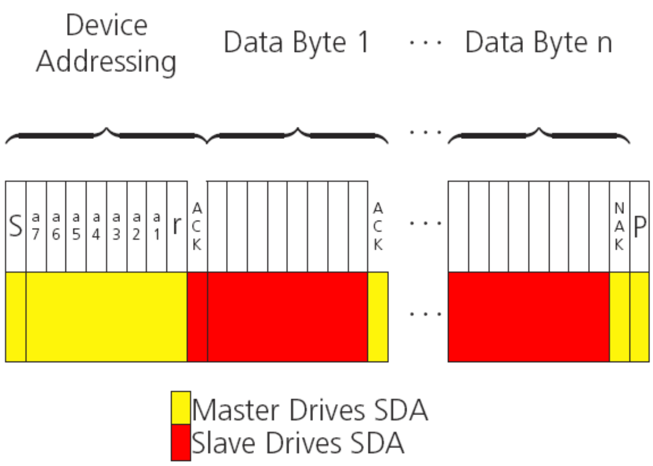


Current Address Read

The receiver contains an address counter that maintains the address of the last register accessed, internally incremented by one. Therefore, if the previous read access was to address `n` (`n` is any legal address), the next current address read operation would access data from address `n+1` (see Figure *DDC Current Address Read Access*). Upon receipt of the device address with the `RW` bit set to one, the receiver issues an acknowledge and the master can read `1` to `N` bytes from the receiver, generating a not-acknowledge and a stop condition after the last byte being read.

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

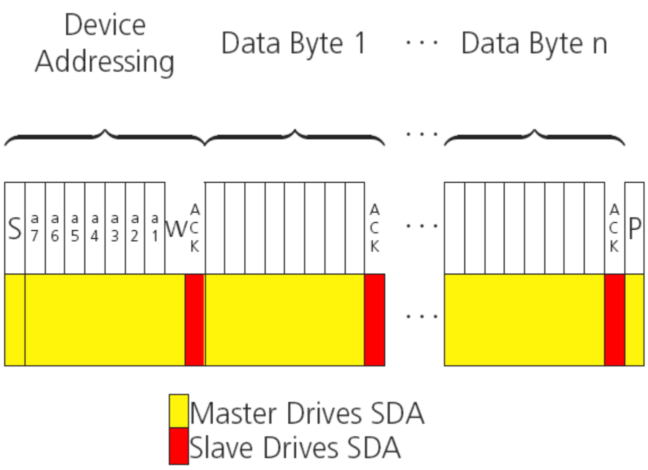
DDC Current Address Read Access



Write Access

The receiver does not provide any write access except for writing UBX messages (and NMEA messages) to the receiver, such as configuration or aiding data. Therefore, the register set mentioned in section Read Access is not writable. Following the start condition from the master, the 7-bit device address including the `RW` bit (which is a logic low for write access) are clocked onto the bus by the master transmitter. The receiver answers with an acknowledge (logic low) to indicate that it is responsible for the given address. Now, the master can write 2 to `N` bytes to the receiver, generating stop condition after the last byte being written. The number of bytes must exceed 2 to properly distinguish from the write access to set the address counter in random read accesses.

DDC Write Access



SPI Port

An SPI Bus (Serial Peripheral Interface is provided, depending on the type of module or chip set (see our online product selector matrix for modules and chip sets). The SPI is a 3-wire synchronous communication interface; In contrast to UART the master provides a clock, meaning that master and slave don't need to be configured to use the same baud rate. Moreover, a baud rate setting is not applicable for the slave. SPI modes 0-3 are implemented and can be configured using the field `mode.spiMode` in CFG-PRT for SPI (default is SPI mode 0).

The baud rate clock provided by the master must not exceed 250kHz

Read Access

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

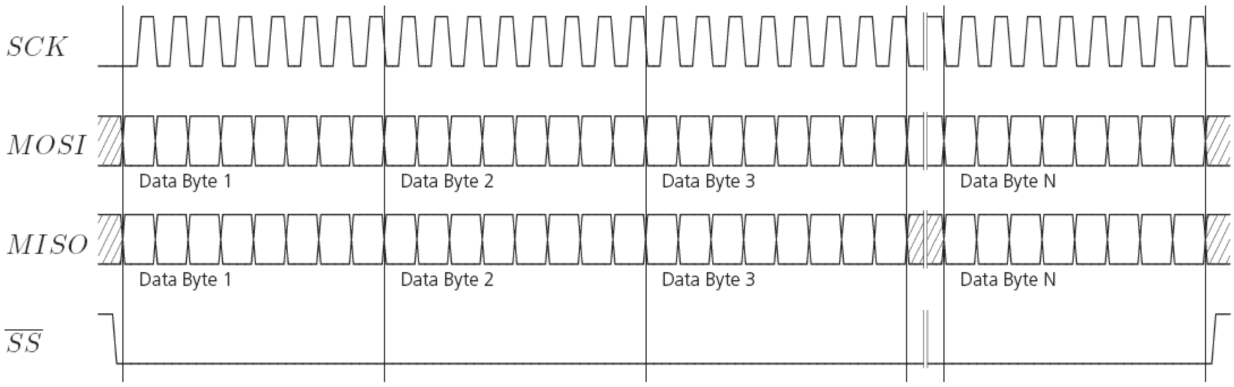
In order to prevent the receiver from being busy parsing the incoming data, the parsing process is stopped after 20 subsequent bytes containing 0xFF. The parsing process gets re-enabled with the first byte not equal to 0xFF. The number of bytes to wait for deactivation (20 by default) can be adjusted using the field `mode.ffCnt` in CFG-PRT for SPI.

If the receiver has no more data to send, it pulls `MISO` to logic high, i.e. all bytes transmitted are set to 0xFF. This means that the master should ignore all 0xFF which are not part of a message. It can resume data processing as soon as the first byte not equalling 0xFF is received.

Back-To-Back Read and Write Access

The receiver does not provide any write access except for writing UBX messages (and eventually NMEA messages) to the receiver, such as configuration or aiding data. For every byte written to the receiver, a byte must be read from the receiver; the master writes to `MOSI` and, at the same time, it reads from `MISO`. The data on `MISO` represents the results from a current address read, returning 0xFF when no more data is available.

SPI Back-To-Back Read/Write Access



How to change between protocols

Reconfiguring a port from one protocol to another is a two-step process. First of all, the preferred protocol(s) needs to be enabled to a port using `CFG-PRT`. One port can handle several protocols at the same time (e.g. NMEA and UBX). By default, all ports are configured for UBX and NMEA protocol so in most cases, it's not necessary to change the port settings at all. Port settings can be viewed and changed using the `CFG-PRT` messages.

As a second step, activate certain messages on each port using `CFG-MSG`.

Despite the fact that concatenation of several configurations is still possible on receivers before u-blox 5, the use of this feature is discouraged as it won't work on u-blox 5. u-blox 5 has 6 I/O ports, so backwards compatibility is dropped at this point.

This message can be used to initiate receiver restart scenarios, optionally erasing information the receiver has acquired.

Typically, in GPS receivers, one distinguishes between Cold-, Warm- and Hotstarts, depending on the type of valid information the receiver has at the time of the restart.

- **Coldstart** In this startup mode, the receiver has **no** a-priori information on last position, time, velocity, frequency etc. Therefore, the receiver has to search the full time- and frequency space, and also all possible satellite numbers. If a satellite signal is found, it is being tracked to decode ephemeris (18-36 seconds under strong signal conditions), whereas the other channels continue to search satellites. Once there are sufficient number of satellites with valid ephemeris, the receiver can calculate position- and velocity data. Please note that some competitors call this startup mode `Factory Startup`.
- **Warmstart** In warmstart mode, the receiver has approximate information of time, position, and coarse data on Satellite positions (Almanac). In this mode, after power-up, the receiver basically needs to download ephemeris until it can calculate position- and velocity data. As the ephemeris data usually is outdated after 4

hours, the receiver will typically start with a warmstart if it was powered down for more than that amount of time. For this scenario, several augmentations exist. See the sections on AssistNOW online and offline.

- **Hotstart** In Hotstart, the receiver was powered down only for a short time (4 hours or less), so that its ephemeris is still valid. Since the receiver doesn't need to download ephemeris again, this is the fastest startup method.

In the UBX-CFG-RST message, one can force the receiver to reset and clear data, in order to see the effects of maintaining/losing such a-priori data between restarts. For that, the CFG-RST message offers the `navBbrMask` field, where Hot-, Warm- and Coldstarts can be initiated, and also other combinations thereof.

The Reset Type can also be specified. This is not GPS-related, but the way the software restarts the system.

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

Geodetic Datum

Predefined Datum

The following, predefined Datum Values are available and can be configured using UBX message CFG-DAT.

For the ellipsoid parameters, see ellipsoid section below. For the rotation and scale parameters, see rotation and scale section below.

The receiver defaults to WGS84 datum

Geodetic Datum Defined in Firmware

Index	*Description*	*Short*	*Ellipsoid Index*	*Rotation, Scale*	*dX [m]*	*dY [m]*	*dZ [m]*
0	World Geodetic System - 84	WGS84	0	0	0.0	0.0	0.0
1	World Geodetic System - 72	WGS72	23	1	0.0	0.0	4.5
2	Earth-90 - GLONASS Coordinate system	ETH90	8	0	0.0	0.0	4.0
3	Adindan - Mean Solution (Ethiopia & Sudan)	ADI-M	7	0	-166.0	-15.0	204.0
4	Adindan - Burkina Faso	ADI-E	7	0	-118.0	-14.0	218.0
5	Adindan - Cameroon	ADI-F	7	0	-134.0	-2.0	210.0
6	Adindan - Ethiopia	ADI-A	7	0	-165.0	-11.0	206.0
7	Adindan - Mali	ADI-C	7	0	-123.0	-20.0	220.0
8	Adindan - Senegal	ADI-D	7	0	-128.0	-18.0	224.0
9	Adindan - Sudan	ADI-B	7	0	-161.0	-14.0	205.0
10	Afgooye - Somalia	AFG	21	0	-43.0	-163.0	45.0

Geodetic Datum Defined in Firmware continued

Index	Description	Short	Ellipsoid Index	Rotation, Scale	dX [m]	dY [m]	dZ [m]
11	ARC 1950 - Mean (Botswana, Lesotho, Malawi, Swaziland, Zaire, Zambia, Zimbabwe)	ARF-M	7	0	-143.0	-90.0	-294.0
12	ARC 1950 - Botswana	ARF-A	7	0	-138.0	-105.0	-289.0
13	ARC 1950 - Burundi	ARF-H	7	0	-153.0	-5.0	-292.0
14	ARC 1950 - Lesotho	ARF-B	7	0	-125.0	-108.0	-295.0
15	ARC 1950 - Malawi	ARF-C	7	0	-161.0	-73.0	-317.0
16	ARC 1950 - Swaziland	ARF-D	7	0	-134.0	-105.0	-295.0
17	ARC 1950 - Zaire	ARF-E	7	0	-169.0	-19.0	-278.0
18	ARC 1950 - Zambia	ARF-F	7	0	-147.0	-74.0	-283.0
19	ARC 1950 - Zimbabwe	ARF-G	7	0	-142.0	-96.0	-293.0
20	ARC 1960 - Mean (Kenya, Tanzania)	ARS	7	0	-160.0	-6.0	-302.0
21	Ayabelle Lighthouse - Djibouti	PHA	7	0	-79.0	-129.0	145.0
22	Bissau - Guinea-Bissau	BID	20	0	-173.0	253.0	27.0
23	Cape - South Africa	CAP	7	0	-136.0	-108.0	-292.0
24	Carthage - Tunisia	CGE	7	0	-263.0	6.0	431.0
25	Dabola - Guinea	DAL	7	0	-83.0	37.0	124.0
26	Leigon - Ghana	LEH	7	0	-130.0	29.0	364.0
27	Liberia 1964	LIB	7	0	-90.0	40.0	88.0
28	Massawa - Eritrea (Ethiopia)	MAS	5	0	639.0	405.0	60.0
29	Merchich - Morocco	MER	7	0	31.0	146.0	47.0
30	Minna - Cameroon	MIN-A	7	0	-81.0	-84.0	115.0
31	Minna - Nigeria	MIN-B	7	0	-92.0	-93.0	122.0
32	M'Poraloko - Gabon	MPO	7	0	-74.0	-130.0	42.0
33	North Sahara 1959 - Algeria	NSD	7	0	-186.0	-93.0	310.0
34	Old Egyptian 1907 - Egypt	OEG	17	0	-130.0	110.0	-13.0
35	Point 58 - Mean Solution (Burkina Faso & Niger)	PTB	7	0	-106.0	-129.0	165.0
36	Pointe Noire 1948 - Congo	PTN	7	0	-148.0	51.0	-291.0
37	Schwarzeck - Namibia	SCK	5	0	616.0	97.0	-251.0
38	Voirol 1960 - Algeria	VOR	7	0	-123.0	-206.0	219.0
39	Ain El Abd 1970 - Bahrain Island	AIN-A	20	0	-150.0	-250.0	-1.0
40	Ain El Abd 1970 - Saudi Arabia	AIN-B	20	0	-143.0	-236.0	7.0
41	Djakarta (Batavia)- Sumatra (Indonesia)	BAT	5	0	-377.0	681.0	-50.0
42	Hong Kong 1963 - Hong Kong	HKD	20	0	-156.0	-271.0	-189.0
43	Hu-Tzu-Shan - Taiwan	HTN	20	0	-637.0	-549.0	-203.0
44	Indian - Bangladesh	IND-B	9	0	282.0	726.0	254.0
45	Indian - India & Nepal	IND-I	11	0	295.0	736.0	257.0
46	Indian 1954 - Thailand	INF-A	9	0	217.0	823.0	299.0
47	Indian 1960 - Vietnam (near 16N)	ING-A	9	0	198.0	881.0	317.0
48	Indian 1960 - Con Son Island (Vietnam)	ING-B	9	0	182.0	915.0	344.0
49	Indian 1975 - Thailand	INH-A	9	0	209.0	818.0	290.0
50	Indonesian 1974	IDN	19	0	-24.0	-15.0	5.0
51	Kandawala - Sri Lanka	KAN	9	0	-97.0	787.0	86.0
52	Kertau 1948 - West Malaysia & Singapore	KEA	13	0	-11.0	851.0	5.0
53	Nahrwan - Masirah Island (Oman)	NAH-A	7	0	-247.0	-148.0	369.0
54	Nahrwan - United Arab Emirates	NAH-B	7	0	-249.0	-156.0	381.0

Geodetic Datum Defined in Firmware continued

Index	Description	Short	Ellipsoid Index	Rotation, Scale	dX [m]	dY [m]	dZ [m]
55	Nahrwan - Saudi Arabia	NAH-C	7	0	-243.0	-192.0	477.0
56	Oman	FAH	7	0	-346.0	-1.0	224.0
57	Qatar National - Qatar	QAT	20	0	-128.0	-283.0	22.0
58	South Asia - Singapore	SOA	15	0	7.0	-10.0	-26.0
59	Timbalai 1948 - Brunei & East Malaysia (Sarawak & Sabah)	TIL	10	0	-679.0	669.0	-48.0
60	Tokyo - Mean Solution (Japan,Okinawa & South Korea)	TOY-M	5	0	-148.0	507.0	685.0
61	Tokyo - Japan	TOY-A	5	0	-148.0	507.0	685.0
62	Tokyo - Okinawa	TOY-C	5	0	-158.0	507.0	676.0
63	Tokyo - South Korea	TOY-B	5	0	-146.0	507.0	687.0
64	Australian Geodetic 1966 - Australia & Tasmania	AUA	3	0	-133.0	-48.0	148.0
65	Australian Geodetic 1984 - Australia & Tasmania	AUG	3	0	-134.0	-48.0	149.0
66	European 1950 - Mean (AU, B, DK, FN, F, G, GR, I, LUX, NL, N, P, E, S, CH)	EUR-M	20	0	-87.0	-98.0	-121.0
67	European 1950 - Western Europe (AU, DK, FR, G, NL, CH)	EUR-A	20	0	-87.0	-96.0	-120.0
68	European 1950 - Cyprus	EUR-E	20	0	-104.0	-101.0	-140.0
69	European 1950 - Egypt	EUR-F	20	0	-130.0	-117.0	-151.0
70	European 1950 - England, Wales, Scotland & Channel Islands	EUR-G	20	0	-86.0	- 96.0	-120.0
71	European 1950 - England, Wales, Scotland & Ireland	EUR-K	20	0	-86.0	- 96.0	-120.0
72	European 1950 - Greece	EUR-B	20	0	-84.0	-95.0	-130.0
73	European 1950 - Iran	EUR-H	20	0	-117.0	-132.0	-164.0
74	European 1950 - Italy - Sardinia	EUR-I	20	0	-97.0	-103.0	-120.0
75	European 1950 - Italy - Sicily	EUR-J	20	0	-97.0	-88.0	-135.0
76	European 1950 - Malta	EUR-L	20	0	-107.0	-88.0	-149.0
77	European 1950 - Norway & Finland	EUR-C	20	0	-87.0	-95.0	-120.0
78	European 1950 - Portugal & Spain	EUR-D	20	0	-84.0	-107.0	-120.0
79	European 1950 - Tunisia	EUR-T	20	0	-112.0	-77.0	-145.0
80	European 1979 - Mean Solution (AU, FN, NL, N, E, S, CH)	EUS	20	0	-86.0	-98.0	-119.0
81	Hjorsey 1955 - Iceland	HJO	20	0	-73.0	46.0	-86.0
82	Ireland 1965	IRL	2	0	506.0	-122.0	611.0
83	Ordnance Survey of GB 1936 - Mean (E, IoM, S, ShI, W)	OGB-M	1	0	375.0	-111.0	431.0
84	Ordnance Survey of GB 1936 - England	OGB-A	1	0	371.0	-112.0	434.0
85	Ordnance Survey of GB 1936 - England, Isle of Man & Wales	OGB-B	1	0	371.0	-111.0	434.0
86	Ordnance Survey of GB 1936 - Scotland & Shetland Isles	OGB-C	1	0	384.0	-111.0	425.0
87	Ordnance Survey of GB 1936 - Wales	OGB-D	1	0	370.0	-108.0	434.0

Geodetic Datum Defined in Firmware continued

Index	Description	Short	Ellipsoid Index	Rotation, Scale	dX [m]	dY [m]	dZ [m]
88	Rome 1940 - Sardinia Island	MOD	20	0	-225.0	-65.0	9.0
89	S-42 (Pulkovo 1942) - Hungary	SPK	21	0	28.0	-121.0	-77.0
90	S-JTSK Czechoslavakia (prior to 1 Jan 1993)	CCD	5	0	589.0	76.0	480.0
91	Cape Canaveral - Mean Solution (Florida & Bahamas)	CAC	6	0	-2.0	151.0	181.0
92	N. American 1927 - Mean Solution (CONUS)	NAS-C	6	0	-8.0	160.0	176.0
93	N. American 1927 - Western US	NAS-B	6	0	-8.0	159.0	175.0
94	N. American 1927 - Eastern US	NAS-A	6	0	-9.0	161.0	179.0
95	N. American 1927 - Alaska (excluding Aleutian Islands)	NAS-D	6	0	-5.0	135.0	172.0
96	N. American 1927 - Aleutian Islands, East of 180W	NAS-V	6	0	-2.0	152.0	149.0
97	N. American 1927 - Aleutian Islands, West of 180W	NAS-W	6	0	2.0	204.0	105.0
98	N. American 1927 - Bahamas (excluding San Salvador Island)	NAS-Q	6	0	-4.0	154.0	178.0
99	N. American 1927 - San Salvador Island	NAS-R	6	0	1.0	140.0	165.0
100	N. American 1927 - Canada Mean Solution (including Newfoundland)	NAS-E	6	0	-10.0	158.0	187.0
101	N. American 1927 - Alberta & British Columbia	NAS-F	6	0	-7.0	162.0	188.0
102	N. American 1927 - Eastern Canada (Newfoundland, New Brunswick, Nova Scotia & Quebec)	NAS-G	6	0	-22.0	160.0	190.0
103	N. American 1927 - Manitoba & Ontario	NAS-H	6	0	-9.0	157.0	184.0
104	N. American 1927 - Northwest Territories & Saskatchewan	NAS-I	6	0	4.0	159.0	188.0
105	N. American 1927 - Yukon	NAS-J	6	0	-7.0	139.0	181.0
106	N. American 1927 - Canal Zone	NAS-O	6	0	0.0	125.0	201.0
107	N. American 1927 - Caribbean	NAS-P	6	0	-3.0	142.0	183.0
108	N. American 1927 - Central America	NAS-N	6	0	0.0	125.0	194.0
109	N. American 1927 - Cuba	NAS-T	6	0	-9.0	152.0	178.0
110	N. American 1927 - Greenland (Hayes Peninsula)	NAS-U	6	0	11.0	114.0	195.0
111	N. American 1927 - Mexico	NAS-L	6	0	-12.0	130.0	190.0
112	N. American 1983 - Alaska (excluding Aleutian Islands)	NAR-A	16	0	0.0	0.0	0.0
113	N. American 1983 - Aleutian Islands	NAR-E	16	0	-2.0	0.0	4.0
114	N. American 1983 - Canada	NAR-B	16	0	0.0	0.0	0.0
115	N. American 1983 - Mean Solution (CONUS)	NAR-C	16	0	0.0	0.0	0.0
116	N. American 1983 - Hawaii	NAR-H	16	0	1.0	1.0	-1.0
117	N. American 1983 - Mexico & Central America	NAR-D	16	0	0.0	0.0	0.0
118	Bogota Observatory - Colombia	BOO	20	0	307.0	304.0	-318.0
119	Campo Inchauspe 1969 - Argentina	CAI	20	0	-148.0	136.0	90.0
120	Chua Astro - Paraguay	CHU	20	0	-134.0	229.0	-29.0
121	Corrego Alegre - Brazil	COA	20	0	-206.0	172.0	-6.0

Geodetic Datum Defined in Firmware continued

Index	*Description*	*Short*	*Ellipsoid Index*	*Rotation, Scale*	*dX [m]*	*dY [m]*	*dZ [m]*
122	Prov S. American 1956 - Mean Solution (Bol, Col, Ecu, Guy, Per & Ven)	PRP-M	20	0	-288.0	175.0	-376.0
123	Prov S. American 1956 - Bolivia	PRP-A	20	0	-270.0	188.0	-388.0
124	Prov S. American 1956 - Northern Chile (near 19S)	PRP-B	20	0	-270.0	183.0	-390.0
125	Prov S. American 1956 - Southern Chile (near 43S)	PRP-C	20	0	-305.0	243.0	-442.0
126	Prov S. American 1956 - Colombia	PRP-D	20	0	-282.0	169.0	-371.0
127	Prov S. American 1956 - Ecuador	PRP-E	20	0	-278.0	171.0	-367.0
128	Prov S. American 1956 - Guyana	PRP-F	20	0	-298.0	159.0	-369.0
129	Prov S. American 1956 - Peru	PRP-G	20	0	-279.0	175.0	-379.0
130	Prov S. American 1956 - Venezuela	PRP-H	20	0	-295.0	173.0	-371.0
131	Prov South Chilean 1963	HIT	20	0	16.0	196.0	93.0
132	South American 1969 - Mean Solution (Arg, Bol, Bra, Chi, Col, Ecu, Guy, Par, Per, Tri & Tob, Ven)	SAN-M	22	0	-57.0	1.0	-41.0
133	South American 1969 - Argentina	SAN-A	22	0	-62.0	-1.0	-37.0
134	South American 1969 - Bolivia	SAN-B	22	0	-61.0	2.0	-48.0
135	South American 1969 - Brazil	SAN-C	22	0	-60.0	-2.0	-41.0
136	South American 1969 - Chile	SAN-D	22	0	-75.0	-1.0	-44.0
137	South American 1969 - Colombia	SAN-E	22	0	-44.0	6.0	-36.0
138	South American 1969 - Ecuador (excluding Galapagos Islands)	SAN-F	22	0	-48.0	3.0	-44.0
139	South American 1969 - Baltra, Galapagos Islands	SAN-J	22	0	-47.0	26.0	-42.0
140	South American 1969 - Guyana	SAN-G	22	0	-53.0	3.0	-47.0
141	South American 1969 - Paraguay	SAN-H	22	0	-61.0	2.0	-33.0
142	South American 1969 - Peru	SAN-I	22	0	-58.0	0.0	-44.0
143	South American 1969 - Trinidad & Tobago	SAN-K	22	0	-45.0	12.0	-33.0
144	South American 1969 - Venezuela	SAN-L	22	0	-45.0	8.0	-33.0
145	Zanderij - Suriname	ZAN	20	0	-265.0	120.0	-358.0
146	Antigua Island Astro 1943 - Antigua, Leeward Islands	AIA	7	0	-270.0	13.0	62.0
147	Ascension Island 1958	ASC	20	0	-205.0	107.0	53.0
148	Astro Dos 71/4 - St Helena Island	SHB	20	0	-320.0	550.0	-494.0
149	Bermuda 1957 - Bermuda Islands	BER	6	0	-73.0	213.0	296.0
150	Deception Island, Antarctica	DID	7	0	260.0	12.0	-147.0
151	Fort Thomas 1955 - Nevis, St Kitts, Leeward Islands	FOT	7	0	-7.0	215.0	225.0
152	Graciosa Base SW 1948 - Faial, Graciosa, Pico, Sao Jorge, Terceira Islands (Azores)	GRA	20	0	-104.0	167.0	-38.0
153	ISTS 061 Astro 1968 - South Georgia Islands	ISG	20	0	-794.0	119.0	-298.0
154	L.C. 5 Astro 1961 - Cayman Brac Island	LCF	6	0	42.0	124.0	147.0
155	Montserrat Island Astro 1958 - Montserrat Leeward Islands	ASM	7	0	174.0	359.0	365.0

Geodetic Datum Defined in Firmware continued

Index	*Description*	*Short*	*Ellipsoid Index*	*Rotation, Scale*	*dX [m]*	*dY [m]*	*dZ [m]*
156	Naparima, BWI - Trinidad & Tobago	NAP	20	0	-10.0	375.0	165.0
157	Observatorio Meteorologico 1939 - Corvo and Flores Islands (Azores)	FLO	20	0	-425.0	-169.0	81.0
158	Pico De Las Nieves - Canary Islands	PLN	20	0	-307.0	-92.0	127.0
159	Porto Santo 1936 - Porto Santo and Madeira Islands	POS	20	0	-499.0	-249.0	314.0
160	Puerto Rico - Puerto Rico & Virgin Islands	PUR	6	0	11.0	72.0	-101.0
161	Qornoq - South Greenland	QUO	20	0	164.0	138.0	-189.0
162	Sao Braz - Soa Miguel, Santa Maria Islands (Azores)	SAO	20	0	-203.0	141.0	53.0
163	Sapper Hill 1943 - East Falkland Island	SAP	20	0	-355.0	21.0	72.0
164	Selvagem Grande 1938 - Salvage Islands	SGM	20	0	-289.0	-124.0	60.0
165	Tristan Astro 1968 - Tristan du Cunha	TDC	20	0	-632.0	438.0	-609.0
166	Anna 1 Astro 1965 - Cocos Islands	ANO	3	0	-491.0	-22.0	435.0
167	Gandajika Base 1970 - Republic of Maldives	GAA	20	0	-133.0	-321.0	50.0
168	ISTS 073 Astro 1969 - Diego Garcia	IST	20	0	208.0	-435.0	-229.0
169	Kerguelen Island 1949 - Kerguelen Island	KEG	20	0	145.0	-187.0	103.0
170	Mahe 1971 - Mahe Island	MIK	7	0	41.0	-220.0	-134.0
171	Reunion - Mascarene Islands	RUE	20	0	94.0	-948.0	-1262.0
172	American Samoa 1962 - American Samoa Islands	AMA	6	0	-115.0	118.0	426.0
173	Astro Beacon E 1945 - Iwo Jima	ATF	20	0	145.0	75.0	-272.0
174	Astro Tern Island (Frig) 1961 - Tern Island	TRN	20	0	114.0	-116.0	-333.0
175	Astronomical Station 1952 - Marcus Island	ASQ	20	0	124.0	-234.0	-25.0
176	Bellevue (IGN) - Efate and Erromango Islands	IBE	20	0	-127.0	-769.0	472.0
177	Canton Astro 1966 - Phoenix Islands	CAO	20	0	298.0	-304.0	-375.0
178	Chatham Island Astro 1971 - Chatham Island (New Zeland)	CHI	20	0	175.0	-38.0	113.0
179	DOS 1968 - Gizo Island (New Georgia Islands)	GIZ	20	0	230.0	-199.0	-752.0
180	Easter Island 1967 - Easter Island	EAS	20	0	211.0	147.0	111.0
181	Geodetic Datum 1949 - New Zealand	GEO	20	0	84.0	-22.0	209.0
182	Guam 1963 - Guam Island	GUA	6	0	-100.0	-248.0	259.0
183	GUX 1 Astro - Guadalcanal Island	DOB	20	0	252.0	-209.0	-751.0
184	Indonesian 1974 - Indonesia	IDN	19	0	-24.0	-15.0	5.0
185	Johnston Island 1961 - Johnston Island	JOH	20	0	189.0	-79.0	-202.0
186	Kusaie Astro 1951 - Caroline Islands, Fed. States of Micronesia	KUS	20	0	647.0	1777.0	-1124.0
187	Luzon - Philippines (excluding Mindanao Island)	LUZ-A	6	0	-133.0	-77.0	-51.0
188	Luzon - Mindanao Island (Philippines)	LUZ-B	6	0	-133.0	-79.0	-72.0
189	Midway Astro 1961 - Midway Islands	MID	20	0	912.0	-58.0	1227.0
190	Old Hawaiian - Mean Solution	OHA-M	6	0	61.0	-285.0	-181.0
191	Old Hawaiian - Hawaii	OHA-A	6	0	89.0	-279.0	-183.0
192	Old Hawaiian - Kauai	OHA-B	6	0	45.0	-290.0	-172.0
193	Old Hawaiian - Maui	OHA-C	6	0	65.0	-290.0	-190.0
194	Old Hawaiian - Oahu	OHA-D	6	0	58.0	-283.0	-182.0

Geodetic Datum Defined in Firmware continued

Index	*Description*	*Short*	*Ellipsoid Index*	*Rotation, Scale*	*dX [m]*	*dY [m]*	*dZ [m]*
195	Pitcairn Astro 1967 - Pitcairn Island	PIT	20	0	185.0	165.0	42.0
196	Santo (Dos) 1965 - Espirito Santo Island	SAE	20	0	170.0	42.0	84.0
197	Viti Levu 1916 - Viti Levu Island (Fiji Islands)	MVS	7	0	51.0	391.0	-36.0
198	Wake-Eniwetok 1960 - Marshall Islands	ENW	18	0	102.0	52.0	-38.0
199	Wake Island Astro 1952 - Wake Atoll	WAK	20	0	276.0	-57.0	149.0
200	Bukit Rimpah - Bangka and Belitung Islands (Indonesia)	BUR	5	0	-384.0	664.0	-48.0
201	Camp Area Astro - Camp McMurdo Area, Antarctica	CAZ	20	0	-104.0	-129.0	239.0
202	European 1950 - Iraq, Israel, Jordan, Kuwait, Lebanon, Saudi Arabia & Syria	EUR-S	20	0	-103.0	-106.0	-141.0
203	Gunung Segara - Kalimantan (Indonesia)	GSE	5	0	-403.0	684.0	41.0
204	Herat North - Afghanistan	HEN	20	0	-333.0	-222.0	114.0
205	Indian - Pakistan	IND-P	9	0	283.0	682.0	231.0
206	Pulkovo 1942 - Russia	PUK	21	0	28.0	-130.0	-95.0
207	Tananarive Observatory 1925 - Madagascar	TAN	20	0	-189.0	-242.0	-91.0
208	Yacare - Uruguay	YAC	20	0	-155.0	171.0	37.0
209	Krassovsky 1942 - Russia	KRA42	21	0	26.0	-139.0	-80.0
210	Lommel Datum 1950 - Belgium & Luxembourg	BLG50	20	0	-55.0	49.0	-158.0
211	Reseau National Belge 1972 - Belgium	RNB72	20	0	-104.0	80.0	-75.0
212	NTF - Nouvelle Triangulation de la France	NTF	7	0	-168.0	-60.0	320.0
213	Netherlands 1921 - Netherlands	NL21	5	0	719.0	47.0	640.0
214	European Datum 1987, IAG RETrig Subcommision.	ED87	20	2	-82.5	-91.7	-117.7
215	Swiss Datum 1903+ (LV95)	CH95	5	0	674.374	15.056	405.346

Ellipsoids

Ellipsoids

Index	*Description*	*Semi Major Axis [m]*	*Flattening*
0	WGS 84	6378137.000	298.257223563
1	Airy 1830	6377563.396	299.3249646
2	Modified Airy	6377340.189	299.3249646
3	Australian National	6378160.000	298.25
4	Bessel 1841 (Namibia)	6377483.865	299.1528128
5	Bessel 1841	6377397.155	299.1528128
6	Clarke 1866	6378206.400	294.9786982
7	Clarke 1880	6378249.145	293.465
8	Earth-90	6378136.000	298.257839303
9	Everest (India 1830)	6377276.345	300.8017
10	Everest (Sabah Sarawak)	6377298.556	300.8017
11	Everest (India 1956)	6377301.243	300.8017
12	Everest (Malaysia 1969)	6377295.664	300.8017
13	Everest (Malay. & Singapore 1948)	6377304.063	300.8017

Ellipsoids continued

Index	Description	Semi Major Axis [m]	Flattening
14	Everest (Pakistan)	6377309.613	300.8017
15	Modified Fischer 1960	6378155.000	298.3
16	GRS 80	6378137.000	298.257222101
17	Helmert 1906	6378200.000	298.3
18	Hough 1960	6378270.000	297.0
19	Indonesian 1974	6378160.000	298.247
20	International 1924	6378388.000	297.0
21	Krassovsky 1940	6378245.000	298.3
22	South American 1969	6378160.000	298.25
23	WGS 72	6378135.000	298.26

Rotation and Scale

Rotation and Scale

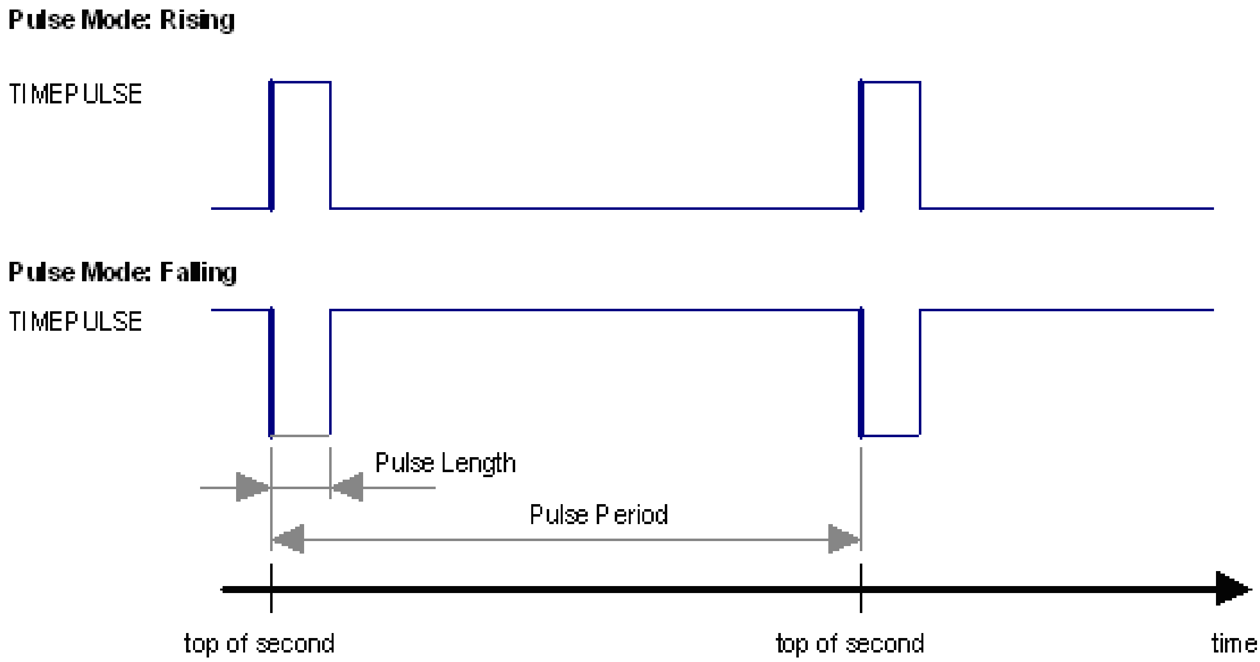
Index	Description	Rot X [seconds]	Rot Y [seconds]	Rot Z [seconds]	Scale
0		+0.0000	+0.0000	+0.0000	0.000
1		+0.0000	+0.0000	-0.5540	0.220
2	European Datum 1987 IAG RETrig Subcommision.	+0.1338	-0.0625	-0.0470	0.045

Timepulse Configuration

The receiver provides a hardware-synchronized timepulse (Pin 29) with a time pulse (TP) period of 1 ms to 60 s. The polarity (rising or falling edge) and the pulse duration can be configured. Use the UBX proprietary message CFG-TP to change the timepulse settings. The UBX-TIM-TP message provides the time information for the next timepulse, time source and a quantization error.

The CFG-TP message comprises the following parameters defining the hardware-synchronized timepulse:

- **pulse interval** - time interval between timepulses
- **pulse length** - duration of the timepulse (time period between rising and falling edge)
- **pulse mode** - if not disabled the synchronization of timepulse can be configured to be done on rising or falling edge
- **time reference** - the reference time source (time base) used for timepulse synchronization and timepulse time given in TIM-TP output message
- **synchronization mode** - the timepulse can be configured to be always synchronized and will be available only in this case. If the timepulse is allowed to be asynchronized it will be available at any time even when the time is not valid.
- **antenna cable delay** - the signal delay due to the cable between antenna and receiver
- **RF group delay** - delay of the signal in the RF module of the u-blox 5 receiver (hard coded)
- **user delay** - the cable delay from u-blox 5 receiver to the user device plus signal delay of any user application



Notes:

- The pulse period must be an integer multiple of 60 seconds.
- The maximum pulse length can't exceed the pulse period minus 1 microsecond.
- A timepulse is only output when the receiver has determined the time with sufficent accuracy and reliability.

Recommendations:

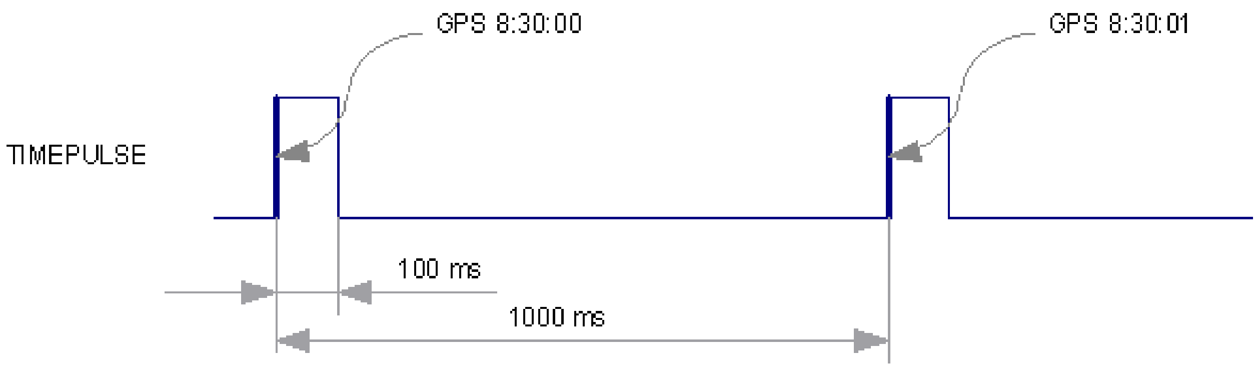
- When using the timepulse for a timing application it is recommended to calibrate the

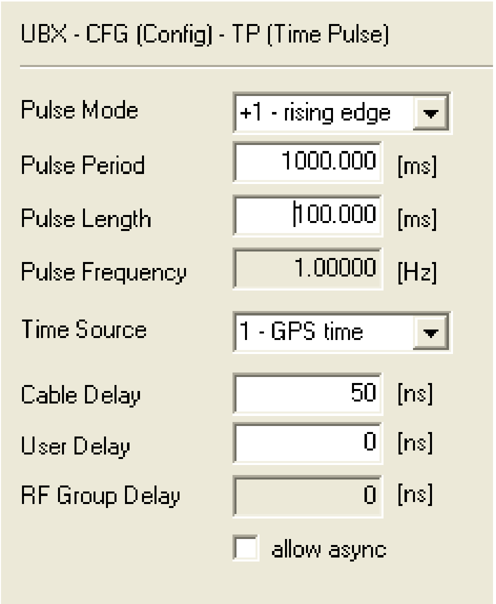
RF signal delay against a reference-timing source.

- In order to get the best timing accuracy with the antenna, a fixed *accurate* position is needed. Once the receiver is in timing mode, the dynamic model does not influence the timing accuracy.

Example:

The example shows the 1PPS timepulse signal generated according the specific parameters of the CFG-TP message.





Receiver Configuration

Configuration Concept

u-blox 5 positioning technology is fully configurable with UBX protocol configuration messages (message class UBX-CFG). The configuration used by the u-blox 5 GPS core during normal operation is termed "Current Configuration". The Current Configuration can be changed during normal operation by sending any UBX-CFG-XXX message to the receiver over an I/O port. The receiver will change its Current Configuration immediately after receiving the configuration message. The u-blox 5 GPS core always uses only the Current Configuration.

Unless the Current Configuration is made permanent by using UBX-CFG-CFG as described below, the Current Configuration will be lost in case of (see message CFG-RST)

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

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

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

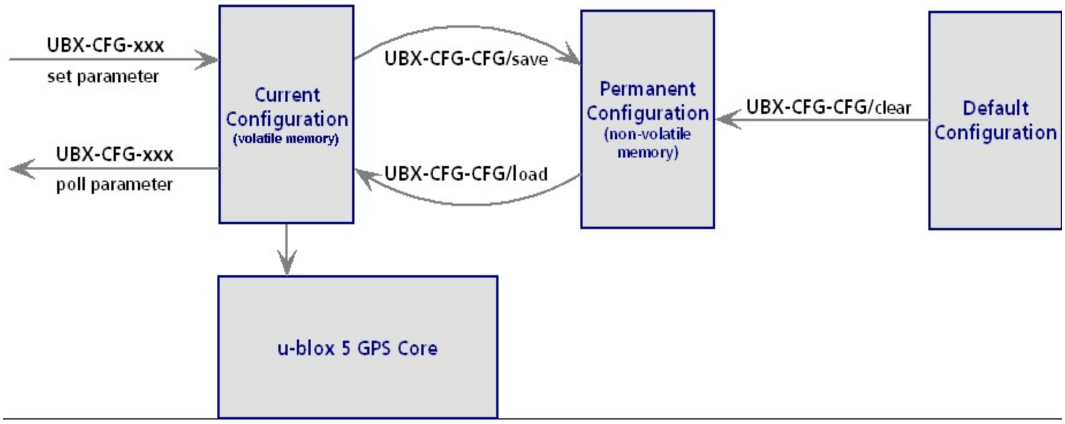
The Permanent Configuration can be restored to the receiver's Default Configuration by sending a UBX-CFG-CFG message with an appropriate **clearMask** (UBX-CFG-CFG/clear) to the receiver.

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

The mentioned masks (saveMask, loadMask, clearMask) are 4 byte bit fields . Every bit represents one configuration sub-section. These sub-sections are defined in section "Organization of the Configuration

Sections"). All three masks are part of every UBX-CFG-CFG message. Save, load and clear commands can be combined in the same message. Order of execution is save, load, clear.

The following diagram illustrates the process:



Organization of the Configuration Sections

The configuration is divided into several sub-sections. Each of these sub-sections corresponds to one or several UBX-CFG-XXX messages. The sub-section numbers in the following table correspond to the bit position in the masks mentioned above.

Configuration sub-sections

sub-section	*CFG messages*	*Description*
0	UBX-CFG-PRT UBX-CFG-USB	Port and USB settings
1	UBX-CFG-MSG	Message settings (enable/disable, update rate)
2	UBX-CFG-INF	Information output settings (Errors, Warnings, Notice, Test etc.)
3	UBX-CFG-NAV5 UBX-CFG-DAT UBX-CFG-RATE UBX-CFG-SBAS UBX-CFG-NMEA UBX-CFG-TMODE	Navigation Parameter, Receiver Datum, Measurement and Navigation Rate setting, Timemode settings, SBAS settings, NMEA protocol settings
4	UBX-CFG-TP	Timepulse Settings
5	N/A	Reserved for future low power modes
6-9	N/A	Reserved for EKF (Dead Reckoning) Receivers
10	UBX-CFG-ANT	Antenna configuration
11-31	N/A	Reserved

Permanent Configuration Storage Media

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

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

* External EEPROM (Electrically Erasable Programmable Read-Only Memory), where available via DDC (I2C compatible).

Receiver Default Configuration

Permanent Configurations can be reset to Default Configurations through a UBX-CFG-CFG/clear message. The receiver's Default Configuration is determined at system startup. The Default Configuration depends on various information such as system clock frequency and others. The receiver searches for this information in various places (memories and configuration pins). Refer to the receiver's data sheet for details.

Power modi for search engine

The receiver determines how and if to search for satellites depending on power configuration (low-level config), number of satellites tracked and if a valid position could be calculated.

Max. Performance mode

In max. performance mode, the receiver searches for all satellites which are currently not tracked on a channel and not invisible (as far as information from satellite pre-positioning is available). If no information is available, the unknown and known-visible satellites are be searched continuously.

Eco mode

In eco mode, if no valid fix could be calculated before, the receiver searches for all satellites with the search engine as then no assumptions about visibility can be made. After a fix could be calculated, the receiver no more uses the search engine to search for satellites without pre-positioning information. Pre-positioning information is available for satellites if orbits for this special SV, and position and time are known at the receiver. If a confirmed position and time are determined and a sufficient number (more or equal to 4) of satellites are tracked, the search engine is completely powered off.

Remark that even if the search engine is powered off, satellites can be found and tracked due to pre-positioning information (slightly slower) or without information at all (significantly slower).

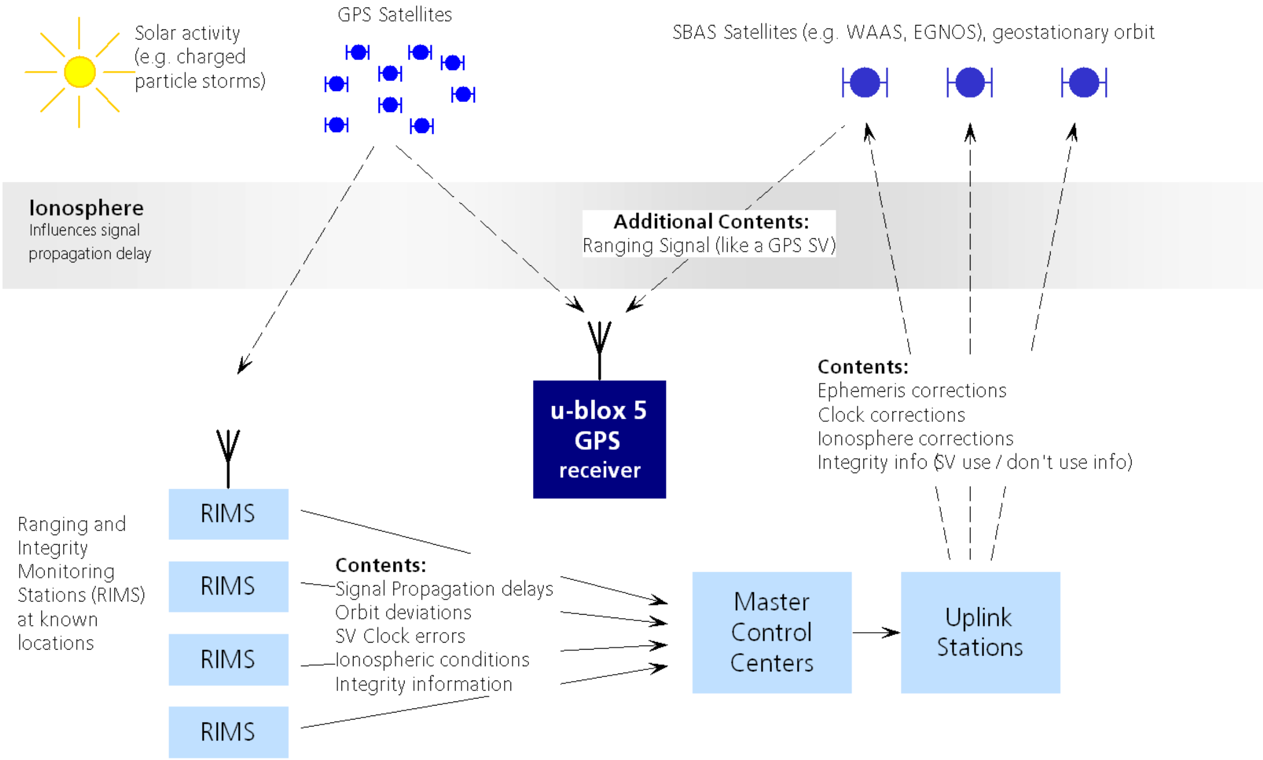
Additionally to these strategic changes, the search engine does not use all resources available in the search engine, saving computational load and therefore reducing power consumption, but increasing mean time to find the satellites.

SBAS Configuration Settings Description

SBAS (Satellite Based Augmentation Systems)

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

SBAS Principle



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

- WAAS (Wide Area Augmentation System) for North America has been in operation since 2003.
- MSAS (Multi-Functional Satellite Augmentation System) for Asia has been in operation since 2007.
- EGNOS (European Geostationary Navigation Overlay Service) is in test mode ESTB (EGNOS satellite test bed). EGNOS has passed the ORR (Operational Readiness Review) in Q2/2005. Full operation of EGNOS is planned for 2008.
- GAGAN (GPS Aided Geo Augmented Navigation), developed by the Indian government is in test mode and expected to be operational by 2010.

Other systems are planned for Canada (CSAS), Africa (EGNOS) and South America.

SBAS support allows u-blox 5 technology to take full advantage of the augmentation systems that are currently available (WAAS, EGNOS, MSAS), as well as those being tested and planned (such as GAGAN).

With SBAS enabled the user benefits from additional satellites for ranging (navigation). u-blox 5 technology uses the available SBAS Satellites for navigation just like GPS satellites, if the SBAS satellites offer this service.

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

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

Another benefit is the use of GPS integrity information. In this way SBAS Control stations can 'disable' usage of GPS satellites in case of major GPS satellite problems within a 6 second alarm time. If integrity monitoring is enabled, u-blox 5 GPS technology will only use satellites, for which integrity information is available.

For more information on SBAS and associated services please refer to

- RTCA/DO-229C (MOPS). Available from www.rtca.org
- gps.faa.gov for information on WAAS and the NSTB

- www.esa.int for information on EGNOS and the ESTB
- www.essp.be for information about European Satellite Services Provider EEIG is the EGNOS operations manager.
- www.kasc.go.jp for information on MSAS

SBAS GEO PRN Numbers

GEO identification	Stationed over	GPS PRN	SBAS Provider
Inmarsat AOR-E	Eastern Africa	120	EGNOS
Inmarsat AOR-W	Western Africa	122	WAAS
ESA Artemis	Africa (Congo)	124	EGNOS
Inmarsat IND-W	Africa (Congo)	126	EGNOS
Insat-NAV	(tbd)	127	GAGAN
Insat-NAV	(tbd)	128	GAGAN
MTSAT-1R (or MTSAT-2)	Pacific	129	MSAS
Inmarsat IOR	Indian Ocean	131	EGNOS
Inmarsat POR	Pacific	134	WAAS
PanAmSat Galaxy XV	133° West	135	WAAS
MTSAT-2 (or MTSAT-1R)	(tbd)	137	MSAS
Telesat Anik F1R	107° West	138	WAAS

SBAS Features

This u-blox 5 SBAS implementation is, in accordance with standard RTCA/DO-229C, a class Beta-1 equipment. All timeouts etc. are chosen for the En Route Case. Do not use this equipment under any circumstances for safety of life applications!

u-blox 5 is capable of receiving multiple SBAS satellites in parallel, even from different SBAS systems (WAAS, EGNOS, MSAS, etc.). They can be tracked and used for navigation simultaneously. At least three SBAS satellites can be tracked in parallel. Every SBAS satellite tracked utilizes one vacant GPS receiver tracking channel. Only the number of receiver channels limits the total number of satellites used. Each SBAS satellite, which broadcasts ephemeris or almanac information, can be used for navigation, just like a normal GPS satellite.

For receiving correction data, the u-blox 5 GPS receiver automatically chooses the best SBAS satellite as its primary source. It will select only one since the information received from other SBAS GEOs is redundant and/or could be inconsistent. The selection strategy is determined by the proximity of the GEOs, the services offered by the GEO, the configuration of the receiver (Testmode allowed/disallowed, Integrity enabled/disabled) and the signal link quality to the GEO.

In case corrections are available from the chosen GEO and used in the navigation calculation, the DGPS flag is set in the receiver's output protocol messages (see `NAV-SOL`, `NAV-STATUS`, `NAV-SVINFO`, `NMEA Position Fix Flags description`).

The most important SBAS feature for accuracy improvement is Ionosphere correction. The measured data from RIMS stations of a region are combined to a TEC (Total Electron Content) Map. This map is transferred to the GPS devices via the GEOs to allow a correction of the ionosphere error on each received satellite.

Supported SBAS messages

Message Type	Message Content	Used from GEO
0(0/2)	Test Mode	All
1	PRN Mask Assignment	Primary
2, 3, 4, 5	Fast Corrections	Primary
6	Integrity	Primary

Supported SBAS messages continued

Message Type	*Message Content*	*Used from GEO*
7	Fast Correction Degradation	Primary
9	GEO Navigation (Ephemeris)	All
10	Degradation	Primary
12	Time Offset	Primary
17	GEO Almanacs	All
18	Ionosphere Grid Point Assignment	Primary
24	Mixed Fast / Long term Corrections	Primary
25	Long term Corrections	Primary
26	Ionosphere Delays	Primary

As each GEO services a specific region, the correction signal is only useful within that region. Therefore, mission planning is crucial to determine the best possible configuration. The different stages (Testmode vs. Operational) of the various SBAS systems further complicate this task. The following examples show possible scenarios:

Example 1: SBAS Receiver in North America

At the time of writing, the WAAS system is in operational stage, whereas the EGNOS system is still in test mode (ESTB). Therefore, and especially in the eastern parts of the US, care must be taken in order not to have EGNOS satellites taking preference over WAAS satellites. This can be achieved by disallowing Test Mode use (this inhibits EGNOS satellites from being used as a correction data source), but keeping the PRN Mask to have all SBAS GEOs enabled (which allows EGNOS GEOs to be used for navigation).

Example 2: SBAS Receiver in Europe

At the time of writing, the EGNOS system is still in test mode. To try out EGNOS operation, Testmode usage must be enabled. Since the WAAS GEO #122 can be received in the western parts of Europe, but since this GEO does not carry correction data for the European continent, the GEOs from all but the EGNOS system should be disallowed, using the PRN Mask. It is important to understand that while EGNOS is in test mode, anything can happen to the EGNOS signals, such as sudden interruption of service or broadcast of invalid or inconsistent data.

The u-blox 5 GPS receiver always makes use of the best available SBAS correction data.

SBAS Configuration

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

SBAS Configuration parameters

Parameter	*Description*
Mode - SBAS Subsystem	Enables or disables the SBAS subsystem
Mode - Allow test mode usage	Allow / Disallow SBAS usage from satellites in Test Mode (Message 0)
Services/Usage - Ranging	Use the SBAS satellites for navigation
Services/Usage - Apply SBAS correction data	Combined enable/disable switch for Fast-, Long-Term and Ionosphere Corrections
Services/Usage - Apply integrity information	Use integrity data

SBAS Configuration parameters continued

Parameter	*Description*
Number of tracking channels	Sets how many channels are reserved for SBAS tracking (if that many SBAS signals were acquired). E.g., if this is set to three and five SBAS SVs are acquired, only three of them will prioritized over available GPS signals.
PRN Mask	Allows to selectively enable/disable SBAS satellite. With this parameter, for example, one can restrict SBAS usage to WAAS-only

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

NMEA Protocol Configuration

The NMEA protocol on u-blox receivers can be configured to the need of customer applications using CFG-NMEA. As default all invalid positions out of the defined accuracy range are not reported.

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

NMEA filtering flags

Parameter	*Description*
Position filtering	If disabled, invalid or old position output is being communicated, but the valid flag indicates that the data is not current.
Masked position filtering	If disabled, Masked position data is still being output, but the valid flag will indicate that the defined accuracy range has been exceeded.
Time filtering	If disabled, the receiver's best knowledge of time is output, even though it might be wrong.
Date filtering	If disabled, the receiver's best knowledge of date is output, even though it might be wrong.
SBAS filtering	If enabled, SBAS satellites are reported according to the NMEA standard.
Track filtering	If disabled, an unfiltered course over ground (COG) output is being output.

NMEA flags

Parameter	*Description*
Compatibility Mode	Some NMEA applications only work with a fixed number of digits behind the decimal comma. Therefore u-blox receivers offer a compatibility mode to communicate with the most popular map applications.
Consideration Mode	u-blox receivers use a sophisticated signal quality detection scheme, in order to produce the best possible position output. This algorithm considers all SV measurements, and eventually decides 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 being communicated as being used for the position determination. If Consideration Mode is disabled, only those satellites are marked as being used, which after the consideration step remained in the position output.

Time Mode Configuration

Introduction

Time Mode is a special stationary GPS 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 as well as for single-SV solutions.

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 a Survey-in. Errors in the fixed position will translate into time errors depending on the satellite constellation. Using the TDOP value (see `UBX-NAV-DOP`) and assuming a symmetrical 3D position error , the expected time error can be estimated as

```
time error = tdop * position error
```

As a rule of thumb the position should be known better than 1m for a time accuracy on the order of nanoseconds. If only microseconds accuracy is required, a position accuracy of roughly 300m is sufficient.

Survey-in

Survey-in is the procedure of determining a stationary receiver's position prior to using *Time Mode* by averaging. The current implementation builds a weighted mean of all valid 3D position solutions. Two stop criteria can 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** forces the calculated position to be of at least the given accuracy. As the position error translates into a time error when using *Time Mode* (see above), one should carefully evaluate the time accuracy requirements and the choose an appropriate position accuracy requirement.

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.

Navigation Configuration Settings Description

Platform settings

u-blox 5 positioning technology supports different dynamic platform models to adjust the navigation engine to the expected environment. These platform settings can be changed dynamically without doing a power cycle or reset. It allows a better interpretation of the measurements and hence provides a more accurate position output. Setting the receiver to an unsuitable platform model for the application environment may reduce the receiver performance and position accuracy significantly.

Dynamic Platform Model

Platform	Description
Portable	Default setting. Applications with low accelerations, as any portable devices. Suitable for most situations.
Stationary	Used in timing applications (antenna must be stationary) or other stationary applications. Velocity is constrained to 0 m/s. Zero dynamics assumed.
Pedestrian	Applications with low accelerations and low speed, as a pedestrian would move. Assuming low accelerations.
Automotive	Used for applications that can be compared with the dynamics of a passenger car. Assuming low vertical acceleration.
At sea	Recommended for applications at sea, with zero vertical velocity. Assuming zero vertical velocity.
Airborne <1g	Used for applications that have to handle a higher dynamic range than a car and higher vertical accelerations. No 2D position fixes supported.
Airborne <2g	Recommended for typical airborne environment. No 2D position fixes supported.
Airborne <4g	Only recommended for an extreme dynamic environment. No 2D position fixes supported.

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

Navigation Input Filters

The navigation input filters mask the input data of the navigation engine.

These settings are already optimized. It is not recommended that changes to any parameters be made unless advised by u-blox support engineers.

Navigation Input Filter parameters

Parameter	Description
fixMode	By default, the receiver calculates a 3D position fix if possible but reverts to a 2D position if necessary (**Auto 2D/3D**). It is possible to force the receiver to permanently calculate 2D (**2D only**) or 3D (**3D only**) positions.
fixedAlt and fixedAltVar	The fixed altitude is used if fixMode is set to 2D only. A variance greater than zero must be supplied as well.
minElev	Minimum elevation of a satellite above the horizon in order to be used in the navigation solution. Low elevation satellites may provide degraded accuracy, because of the long signal path through the atmosphere.
drLimit	Dead Reckoning limit: The time during which the receiver provides an extrapolated solution. After the DR timeout has expired, no GPS solution is provided at all.

Navigation Output Filters

The navigation output filters adjust the valid flag of the relevant NMEA and UBX output messages. Users of the UBX protocol have additional access to messages containing an accuracy indicator, along with the position, time and velocity solutions.

- The **pDop** and **pAcc** values: The PDOP and Position Accuracy Mask are used to determine if a position solution is marked valid in the NMEA sentences or if the UBX PosLimit flag is set. A solution is considered valid, when both PDOP and Accuracy lie below the respective limits.

* The **tDop** and **tAcc** values: The TDOP and Time Accuracy Mask are used to determine when a time pulse should be allowed. The time pulse is disabled if either TDOP or the time accuracy exceeds its respective limit. See also the TIM-TP message description.

Static Hold

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

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

Degraded Navigation

Degraded navigation describes all navigation modes, which use less than 4 satellites.

2D Navigation

If the receiver only has 3 satellites to calculate a position, the navigation algorithm uses a constant altitude to make up for the missing fourth satellite. When losing a satellite after a successful 3D fix (min. 4 SV available), the altitude is kept constant to the last known altitude. This is called a 2D fix.

The u-blox 5 positioning technology does not calculate any solution with a number of SVs less than 3. Only u-blox 5 Timing Receivers can calculate timing solution with only one SV when stationary.

Dead Reckoning, Extrapolating Positioning

The implemented extrapolation algorithm kicks in as soon as the receiver no longer achieves a position fix with a sufficient position accuracy or DOP value (see section Navigation Output Filters). It keeps a fix track (heading is equal to the last calculated heading) until the Dead Reckoning Timeout is reached. The position is extrapolated but it's indicated as "NoFix" (except for NMEA V2.1).

For sensor based Dead Reckoning GPS solutions, u-blox offers Dead Reckoning enabled GPS modules. They allow high accuracy position solutions for automotive applications at places with poor or no GPS coverage. This technology relies on additional inputs like a turn rate sensor (gyro) or a speed sensor (odometer or wheel tick).

Receiver Status Monitoring

Messages in this class are used to report the status of the non-GPS-specific parts of the embedded computer system.

The main purposes are

* Stack- and CPU load (Antaris 4, only)
* Hard- and Software Versions, using MON-VER

- Status of the Communications Input/Output system
- Status of various Hardware Sections with MON-HW

Input/Output system

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

If too much data is being configured for a certain port's bandwidth (e.g. all UBX messages shall be output on a UART port with a baud rate of 9600), the buffer will fill up. Once the buffer's space is exceeded, the receiver will deactivate messages automatically.

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

A `target` in the context of the I/O system is a I/O protocol. The following table shows the target numbers used

Target Number assignment

Target #	*Electrical Interface*
0	DDC (I2C compatible)
1	UART 1
2	UART 2
3	USB
4	SPI
5	reserved

Protocol Number assignment

Protocol #	*Protocol Name*
0	UBX Protocol
1	NMEA Protocol
2	RTCM Protocol (not supported on u-blox 5)
3	RAW Protocol (not supported on u-blox 5)
4..7	Reserved for future use

Aiding

Introduction

The UBX Message Class AID provides all mechanisms for providing Assiste GPS Data to u-blox GPS receivers, including AssistNow Online and AssistNow Offline.

Aiding Data

Following aiding data can be submitted to the receiver:

- **Position** Position information can be submitted to the receiver using the `UBX-AID-INI` message. Both, ECEF X/Y/Z and latitude/longitude/height formats are supported.
- **Time** The time can either be supplied as an inexact value via the standard communication interfaces, suffering from latency depending on the baud rate, or using hardware time synchronization where an accurate time pulse is connected to an external interrupt. Both methods are supported in the `UBX-AID-INI` message.
- **Frequency** It is possible to supply hardware frequency aiding by connecting a continuous signal to an external interrupt using the `UBX-AID-INI` message.
- **Orbit data** Orbit data can be submitted using `UBX-AID-ALM` and `UBX-AID-EPH`.
- **Additional information** `UBX-AID-HUI` can be used to supply health information, UTC parameters and ionospheric data to the receiver.

Aiding Sequence

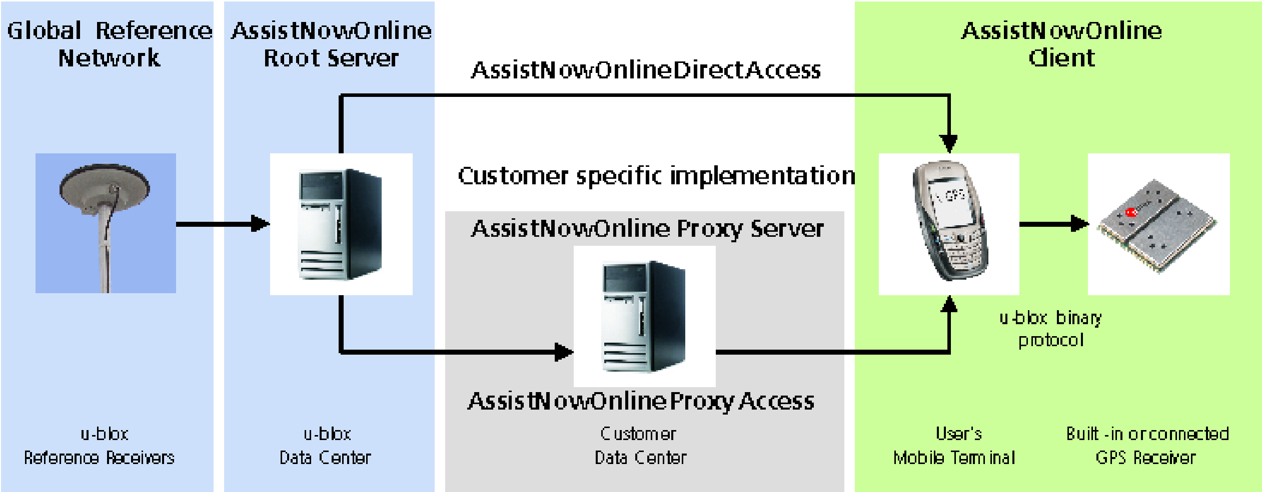
A typical aiding sequence would comprise following steps:

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

AssistNow Online

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

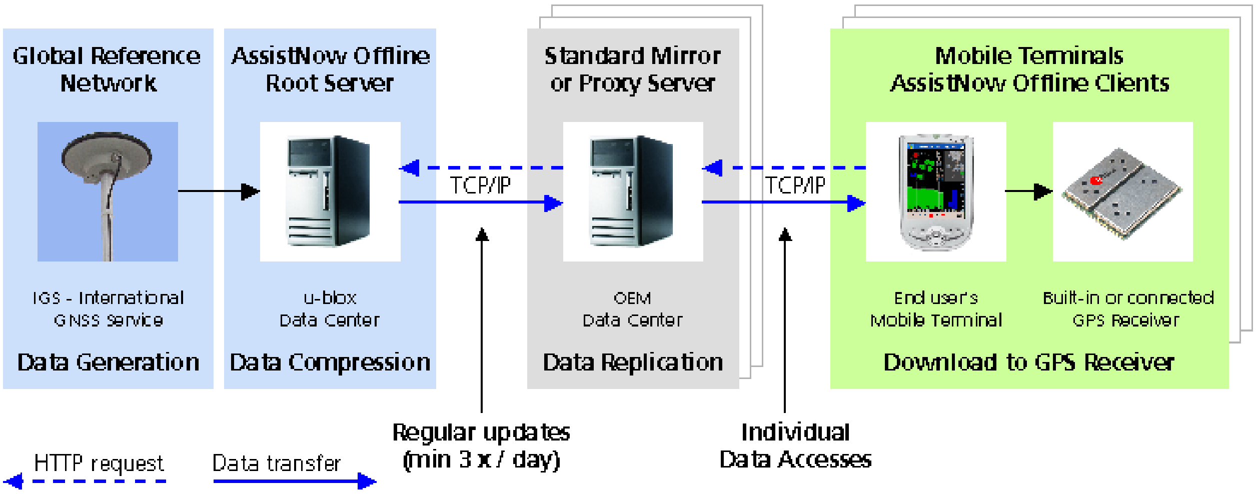
AssistNow Online makes use of User Plane communication and open standards such as TCP/IP. Therefore, it works on all standard mobile communication networks that support Internet access, including GPRS, UMTS and Wireless LAN. No special arrangements need to be made with mobile network operators to enable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



Messaging wise, AssistNow Online consists of Aiding data which deliver Position and Time UBX-AID-INI, Ephemerides UBX-AID-EPH, Almanac UBX-AID-ALM and Health/UTC/Iono information UBX-AID-HUI

AssistNow Offline

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



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

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

The simplest set-up is for GPS receivers including an internal Flash Memory where ALP data can be stored. In this case, the UBX-AID-ALP message is used.

When the GPS receiver does not contain a Flash Memory, the ALP file must be stored to the host CPU. The GPS receiver can the request data from the host when needed. This arrangement is implemented using the UBX-AID-ALPSRV message.

In both cases, status reporting on ALP data currently available to the GPS receiver can be taken from message AID-ALP_STAT

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

Please note that this functionality is only supported on u-blox 5 Firmware 4.0 and above.

Host-based AlmanacPlus Overview

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

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

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

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

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

Overview of the three versions of AID-ALPSRV messages

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

Message specifics

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

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

Range checks

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

Changing ALP files

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

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

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

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

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

Sample Code

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

Please note that this functionality is only supported on u-blox 5 Firmware 4.0 and above and with special versions of Antaris 4 receivers.

Flash-based AlmanacPlus Overview

Flash-based AlmanacPlus functionality means that AlmanacPlus data is stored in the program flash memory connected to the u-blox 5 chip. The task of a server is simply to download the data from an Internet server or other sources, and then deliver the full file piece by piece to the GPS receiver. This is different to the method described in UBX-AID-ALPSRV where the file would remain within the host and the GPS receiver would request chunks from that file when needed.

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

Download Procedure

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

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

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

Please note that

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

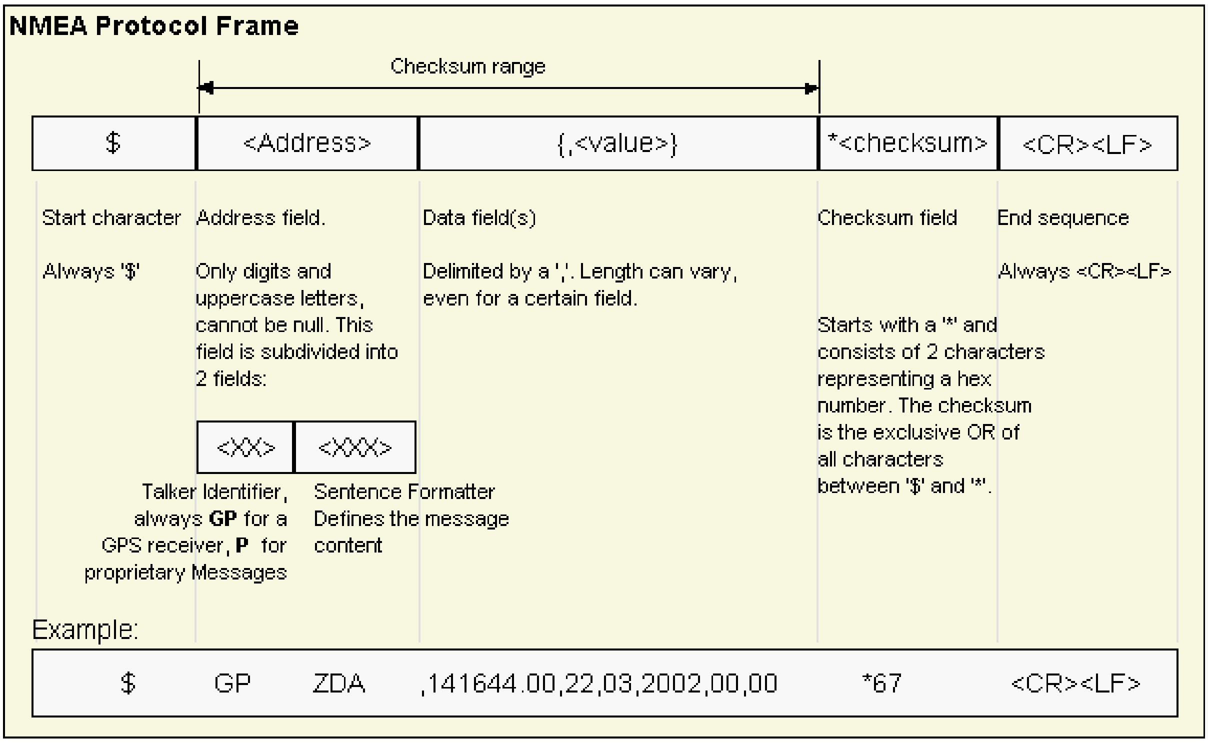
Overview of the different versions of AID-ALP messages

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

NMEA Protocol

Protocol Overview

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



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

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

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

Position Fix Flags in NMEA Mode

The following list shows how u-blox implements the NMEA protocol, and the conditions determining how flags are set in version 2.3 and above.

NMEA Message: Field	*No position fix (at power-up, after losing satellite lock)*	*Valid position fix, but user limits exceeded*	*Dead reckoning (linear extrapolation)*	*EKF (only on DR receivers)*	*2D position fix*	*3D position fix*	*combined GPS/EKF position fix (only on DR receivers)*
GLL, RMC: Status	V	V	V	A	A	A	A
	A=Data VALID, V=Data Invalid (Navigation Receiver Warning)						
GGA: Quality Indicator	0	0	6	6	1 / 2	1 / 2	1 / 2
	0=Fix not available/invalid, 1=GPS SPS Mode, Fix valid, 2=Differential GPS, SPS Mode, Fix Valid, 6=Estimated/Dead Reckoning						
GSA: Nav Mode	1	1	2	2	2	3	3
	1=Fix Not available, 2=2D Fix, 3=3D Fix						
GLL, RMC, VTG: Mode Indicator	N	N	E	E	A / D	A / D	A / D
	N=No Fix, A=Autonomous GNSS Fix, D=Differential GNSS Fix, E=Estimated/Dead Reckoning Fix						
UBX GPSFixOK	0	0	0	1	1	1	1
UBX GPSFix	0	>1	1	1	2	3	4

The following list shows how u-blox implements the NMEA protocol, and the conditions determining how flags are set in version 2.2 and below.

NMEA Message: Field	*No position fix (at power-up, after losing satellite lock*	*Valid position fix, but user limits exceeded*	*Dead reckoning (linear extrapolation)*	*EKF (only on DR receivers)*	*2D position fix*	*3D position fix*	*combined GPS/EKF position fix (only on DR receivers)*
GLL, RMC: Status	V	V	A	A	A	A	A
	A=Data VALID, V=Data Invalid (Navigation Receiver Warning)						
GGA: Quality Indicator	0	0	1	1	1 / 2	1 / 2	1 / 2
	0=Fix not available/invalid, 1=GPS SPS Mode, Fix valid, 2=Differential GPS, SPS Mode, Fix Valid						
GSA: Nav Mode	1	1	2	2	2	3	3
	1=Fix Not available, 2=2D Fix, 3=3D Fix						
GLL, RMC, VTG: Mode Indicator. This field is not output by this NMEA version.							
UBX GPSFixOK	0	0	0	1	1	1	1
UBX GPSFix	0	>1	1	1	2	3	4

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
```

In Antaris firmware versions older than 3.0, the receiver did output invalid data and marked it with the 'Invalid/Valid' Flags. If required, this function can still be enabled in later firmware versions, using the UBX protocol message CFG-NMEA.

NMEA Messages Overview

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

Page	*Mnemonic*	*Cls/ID*	*Description*
	NMEA Proprietary Messages		**Proprietary Messages**
52	**UBX,00**	0xF1 0x00	Lat/Long Position Data
54	**UBX,03**	0xF1 0x03	Satellite Status
56	**UBX,04**	0xF1 0x04	Time of Day and Clock Information
58	**UBX,40**	0xF1 0x40	Set NMEA message output rate
59	**UBX,41**	0xF1 0x41	Set Protocols and Baudrate
57	**UBX**	0xF1 0x40	Poll a PUBX message
	NMEA Standard Messages		**Standard Messages**
49	**DTM**	0xF0 0x0A	Datum Reference
48	**GBS**	0xF0 0x09	GNSS Satellite Fault Detection
38	**GGA**	0xF0 0x00	Global positioning system fix data
40	**GLL**	0xF0 0x01	Latitude and longitude, with time of position fix and status
50	**GPQ**	0xF0 0x40	Poll message
45	**GRS**	0xF0 0x06	GNSS Range Residuals
41	**GSA**	0xF0 0x02	GPS DOP and Active Satellites
46	**GST**	0xF0 0x07	GNSS Pseudo Range Error Statistics
42	**GSV**	0xF0 0x03	GPS Satellites in View
43	**RMC**	0xF0 0x04	Recommended Minimum data
51	**TXT**	0xF0 0x41	Text Transmission
44	**VTG**	0xF0 0x05	Course over ground and Ground speed
47	**ZDA**	0xF0 0x08	Time and Date

Standard Messages

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

GGA

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

Message Structure:

```
$GPGGA,hhmmss.ss,Latitude,N,Longitude,E,FS,NoSV,HDOP,msl,m,Altref,m,DiffAge,DiffStation*cs<CR><LF>
```

Example:

```
$GPGGA,092725.00,4717.11399,N,00833.91590,E,1,8,1.01,499.6,M,48.0,M,,0*5B
```

Field No.	*Example*	*Format*	*Name*	*Unit*	*Description*
0	$GPGGA	string	`$GPGGA`	-	Message ID, GGA protocol header
1	092725.00	hhmmss.sss	`hhmmss.ss`	-	UTC Time, Current time
2	4717.11399	ddmm.mmmm	`Latitude`	-	Latitude, Degrees + minutes, see Format description
3	N	character	`N`	-	N/S Indicator, N=north or S=south
4	00833.91590	dddmm.mmmm	`Longitude`	-	Longitude, Degrees + minutes, see Format description
5	E	character	`E`	-	E/W indicator, E=east or W=west
6	1	digit	`FS`	-	Position Fix Status Indicator, See Table below and Position Fix Flags description
7	8	numeric	`NoSV`	-	Satellites Used, Range 0 to 12
8	1.01	numeric	`HDOP`	-	HDOP, Horizontal Dilution of Precision
9	499.6	numeric	`msl`	m	MSL Altitude
10	M	character	`uMsl`	-	Units, Meters (fixed field)
11	48.0	numeric	`Altref`	m	Geoid Separation
12	M	character	`uSep`	-	Units, Meters (fixed field)
13	-	numeric	`DiffAge`	s	Age of Differential Corrections, Blank (Null) fields when DGPS is not used
14	0	numeric	`DiffStation`	-	Diff. Reference Station ID
15	*5B	hexadecimal	`cs`	-	Checksum
16	-	character	`<CR><LF>`	-	Carriage Return and Line Feed

Table Fix Status

Fix Status	*Description, see also Position Fix Flags description*
0	No Fix / Invalid
1	Standard GPS (2D/3D)
2	Differential GPS
6	Estimated (DR) Fix

GLL

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

Message Structure:

```
$GPGLL,Latitude,N,Longitude,E,hhmmss.ss,Valid,Mode*cs<CR><LF>
```

Example:

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

Field No.	Example	Format	Name	Unit	Description
0	$GPGLL	string	`$GPGLL`	-	Message ID, GLL protocol header
1	4717.11364	ddmm.mmmm	`Latitude`	-	Latitude, Degrees + minutes, see Format description
2	N	character	`N`	-	N/S Indicator, hemisphere N=north or S=south
3	00833.91565	dddmm.mmmm	`Longitude`	-	Longitude, Degrees + minutes, see Format description
4	E	character	`E`	-	E/W indicator, E=east or W=west
5	092321.00	hhmmss.sss	`hhmmss.ss`	-	UTC Time, Current time
6	A	character	`Valid`	-	V = Data invalid or receiver warning, A = Data valid. See Position Fix Flags description
Start of optional block					
7	A	character	`Mode`	-	Positioning Mode, see Position Fix Flags description
End of optional block					
7	*60	hexadecimal	`cs`	-	Checksum
8	-	character	`<CR><LF>`	-	Carriage Return and Line Feed

GSA

Message	**GSA**		
Description	**GPS DOP and Active Satellites**		
Type	Output Message		
Comment	• If less than 12 SVs are used for navigation, the remaining fields are left empty. If more than 12 SVs are used for navigation, only the IDs of the first 12 are output. • The SV Numbers (Fields 'Sv') are in the range of 1 to 32 for GPS satellites, and 33 to 64 for SBAS satellites (33 = SBAS PRN 120, 34 = SBAS PRN 121, and so on)		
	ID for CFG-MSG	*Number of fields*	
Message Info	0xF0 0x02	20	

Message Structure:

```
$GPGSA,Smode,FS{,sv},PDOP,HDOP,VDOP*cs<CR><LF>
```

Example:

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

Field No.	*Example*	*Format*	*Name*	*Unit*	*Description*
0	$GPGSA	string	`$GPGSA`	-	Message ID, GSA protocol header
1	A	character	`Smode`	-	Smode, see first table below
2	3	digit	`FS`	-	Fix status, see second table below and Position Fix Flags description
Start of repeated block (12 times)					
3 + 1*N	29	numeric	`sv`	-	Satellite number
End of repeated block					
15	1.94	numeric	`PDOP`	-	Position dilution of precision
16	1.18	numeric	`HDOP`	-	Horizontal dilution of precision
17	1.54	numeric	`VDOP`	-	Vertical dilution of precision
18	*0D	hexadecimal	`cs`	-	Checksum
19	-	character	`<CR><LF>`	-	Carriage Return and Line Feed

Table Smode

Smode	*Description*
M	Manual - forced to operate in 2D or 3D mode
A	Allowed to automatically switch 2D/3D mode

Table Fix Status

Fix Status	*Description, see also Position Fix Flags description*
1	Fix not available
2	2D Fix
3	3D Fix

GSV

Message	**GSV**		
Description	**GPS Satellites in View**		
Type	Output Message		
Comment	The number of satellites in view, together with each PRN (SV ID), elevation and azimuth, and C/No (Signal/Noise Ratio) value. Only four satellite details are transmitted in one message. There are up to 4 messages used as indicated in the first field NoMsg.		
	ID for CFG-MSG	Number of fields	
Message Info	0xF0 0x03	7..16	

Message Structure:

```
$GPGSV,NoMsg,MsgNo,NoSv,{,sv,elv,az,cno}*cs<CR><LF>
```

Example:

```
$GPGSV,3,1,10,23,38,230,44,29,71,156,47,07,29,116,41,08,09,081,36*7F
$GPGSV,3,2,10,10,07,189,,05,05,220,,09,34,274,42,18,25,309,44*72
$GPGSV,3,3,10,26,82,187,47,28,43,056,46*77
```

Field No.	Example	Format	Name	Unit	Description
0	$GPGSV	string	`$GPGSV`	-	Message ID, GSV protocol header
1	3	digit	`NoMsg`	-	Number of messages, total number of GPGSV messages being output
2	1	digit	`MsgNo`	-	Number of this message
3	10	numeric	`NoSv`	-	Satellites in View
Start of repeated block (1..4 times)					
4 + 4*N	23	numeric	`sv`	-	Satellite ID
5 + 4*N	38	numeric	`elv`	degrees	Elevation, range 0..90
6 + 4*N	230	numeric	`az`	degrees	Azimuth, range 0..359
7 + 4*N	44	numeric	`cno`	dBHz	C/N0, range 0..99, null when not tracking
End of repeated block					
5..16	*7F	hexadecimal	`cs`	-	Checksum
6..16	-	character	`<CR><LF>`	-	Carriage Return and Line Feed

RMC

Message	**RMC**		
Description	**Recommended Minimum data**		
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 GPS/Transit system data.		
	ID for CFG-MSG	*Number of fields*	
Message Info	0xF0 0x04	15	

Message Structure:

```
$GPRMC,hhmmss,status,latitude,N,longitude,E,spd,cog,ddmmyy,mv,mvE,mode*cs<CR><LF>
```

Example:

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

Field No.	Example	Format	Name	Unit	Description
0	$GPRMC	string	`$GPRMC`	-	Message ID, RMC protocol header
1	083559.00	hhmmss.sss	`hhmmss.ss`	-	UTC Time, Time of position fix
2	A	character	`Status`	-	Status, V = Navigation receiver warning, A = Data valid, see Position Fix Flags description
3	4717.11437	ddmm.mmmm	`Latitude`	-	Latitude, Degrees + minutes, see Format description
4	N	character	`N`	-	N/S Indicator, hemisphere N=north or S=south
5	00833.91522	dddmm.mmmm	`Longitude`	-	Longitude, Degrees + minutes, see Format description
6	E	character	`E`	-	E/W indicator, E=east or W=west
7	0.004	numeric	`Spd`	knots	Speed over ground
8	77.52	numeric	`Cog`	degrees	Course over ground
9	091202	ddmmyy	`date`	-	Date in day, month, year format
10	-	numeric	`mv`	degrees	Magnetic variation value, not being output by receiver
11	-	character	`mvE`	-	Magnetic variation E/W indicator, not being output by receiver
12	-	character	`mode`	-	Mode Indicator, see Position Fix Flags description
13	*57	hexadecimal	`cs`	-	Checksum
14	-	character	`<CR><LF>`	-	Carriage Return and Line Feed

VTG

Message	**VTG**		
Description	**Course over ground and Ground speed**		
Type	Output Message		
Comment	Velocity is given as Course over Ground (COG) and Speed over Ground (SOG).		
	ID for CFG-MSG	*Number of fields*	
Message Info	0xF0 0x05	12	

Message Structure:

```
$GPVTG,cogt,T,cogm,M,sog,N,kph,K,mode*cs<CR><LF>
```

Example:

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

Field No.	*Example*	*Format*	*Name*	*Unit*	*Description*
0	$GPVTG	string	`$GPVTG`	-	Message ID, VTG protocol header
1	77.52	numeric	`cogt`	degrees	Course over ground (true)
2	T	character	`T`	-	Fixed field: true
3	-	numeric	`cogm`	degrees	Course over ground (magnetic), not output
4	M	character	`M`	-	Fixed field: magnetic
5	0.004	numeric	`sog`	knots	Speed over ground
6	N	character	`N`	-	Fixed field: knots
7	0.008	numeric	`kph`	km/h	Speed over ground
8	K	character	`K`	-	Fixed field: kilometers per hour
9	A	character	`mode`	-	Mode Indicator, see Position Fix Flags description
10	*06	hexadecimal	`cs`	-	Checksum
11	-	character	`<CR><LF>`	-	Carriage Return and Line Feed

GRS

Message	**GRS**
Description	**GNSS Range Residuals**
Type	Output Message
Comment	**This messages relates to associated GGA and GSA messages.** If less than 12 SVs are available, the remaining fields are output empty. If more than 12 SVs are used, only the residuals of the first 12 SVs are output, in order to remain consistent with the NMEA standard.

	ID for CFG-MSG	*Number of fields*	
Message Info	0xF0 0x06	17	

Message Structure:

```
$GPGRS,hhmmss.ss, mode {,residual}*cs<CR><LF>
```

Example:

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

Field No.	*Example*	*Format*	*Name*	*Unit*	*Description*
0	$GPGRS	string	`$GPGRS`	-	Message ID, GRS protocol header
1	082632.00	hhmmss.sss	`hhmmss.ss`	-	UTC Time, Time of associated position fix
2	1	digit	`mode`	-	Mode (see table below), u-blox receivers will always output Mode 1 residuals
Start of repeated block (12 times)					
3 + 1*N	0.54	numeric	`residual`	m	Range residuals for SVs used in navigation. The SV order matches the order from the GSA sentence.
End of repeated block					
15	*70	hexadecimal	`cs`	-	Checksum
16	-	character	`<CR><LF>`	-	Carriage Return and Line Feed

Table Mode

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

GST

Message	**GST**		
Description	**GNSS Pseudo Range Error Statistics**		
Type	Output Message		
Comment	-		
	ID for CFG-MSG	*Number of fields*	
Message Info	0xF0 0x07	11	

Message Structure:

```
$GPGST,hhmmss.ss,range_rms,std_major,std_minor,hdg,std_lat,std_long,std_alt*cs<CR><LF>
```

Example:

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

Field No.	*Example*	*Format*	*Name*	*Unit*	*Description*
0	$GPGST	string	`$GPGST`	-	Message ID, GST protocol header
1	082356.00	hhmmss.sss	`hhmmss.ss`	-	UTC Time, Time of associated position fix
2	1.8	numeric	`range_rms`	m	RMS value of the standard deviation of the ranges
3	-	numeric	`std_major`	m	Standard deviation of semi-major axis, not supported (empty)
4	-	numeric	`std_minor`	m	Standard deviation of semi-minor axis, not supported (empty)
5	-	numeric	`hdg`	degrees	Orientation of semi-major axis, not supported (empty)
6	1.7	numeric	`std_lat`	m	Standard deviation of latitude, error in meters
7	1.3	numeric	`std_long`	m	Standard deviation of longitude, error in meters
8	2.2	numeric	`std_alt`	m	Standard deviation of altitude, error in meters
9	*7E	hexadecimal	`cs`	-	Checksum
10	-	character	`<CR><LF>`	-	Carriage Return and Line Feed

ZDA

Message	**ZDA**		
Description	**Time and Date**		
Type	Output Message		
Comment	-		
	ID for CFG-MSG	*Number of fields*	
Message Info	0xF0 0x08	9	

Message Structure:

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

Example:

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

Field No.	*Example*	*Format*	*Name*	*Unit*	*Description*
0	$GPZDA	string	`$GPZDA`	-	Message ID, ZDA protocol header
1	082710.00	hhmmss.sss	`hhmmss.ss`	-	UTC Time
2	16	dd	`day`	day	UTC time: day, 01..31
3	09	mm	`month`	month	UTC time: month, 01..12
4	2002	yyyy	`year`	year	UTC time: 4 digit year
5	00	-xx	`ltzh`	-	Local zone hours, not supported (fixed to 00)
6	00	zz	`ltzn`	-	Local zone minutes, not supported (fixed to 00)
7	*64	hexadecimal	`cs`	-	Checksum
8	-	character	`<CR><LF>`	-	Carriage Return and Line Feed

GBS

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

Message Structure:

```
$GPGBS,hhmmss.ss,errlat,errlon,erralt,svid,prob,bias,stddev*cs<CR><LF>
```

Example:

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

Field No.	*Example*	*Format*	*Name*	*Unit*	*Description*
0	$GPGBS	string	`$GPGBS`	-	Message ID, GBS protocol header
1	235503.00	hhmmss.sss	`hhmmss.ss`	-	UTC Time, Time to which this RAIM sentence belongs
2	1.6	numeric	`errlat`	m	Expected error in latitude
3	1.4	numeric	`errlon`	m	Expected error in longitude
4	3.2	numeric	`erralt`	m	Expected error in altitude
5	03	numeric	`svid`	-	Satellite ID of most likely failed satellite
6	-	numeric	`prob`	-	Probability of missed detection, no supported (empty)
7	-21.4	numeric	`bias`	m	Estimate on most likely failed satellite (a priori residual)
8	3.8	numeric	`stddev`	m	Standard deviation of estimated bias
9	*40	hexadecimal	`cs`	-	Checksum
10	-	character	`<CR><LF>`	-	Carriage Return and Line Feed

DTM

Message	**DTM**		
Description	**Datum Reference**		
Type	Output Message		
Comment	This message gives the difference between the currently selected Datum, and the reference Datum. If the currently configured Datum is not WGS84 or WGS72, then the field **LLL** will be set to 999, and the field **LSD** is set to a variable-lenght string, representing the Name of the Datum. The list of supported datums can be found in CFG-DAT. The reference Datum can not be changed and is always set to WGS84.		
	ID for CFG-MSG	*Number of fields*	
Message Info	0xF0 0x0A	11	

Message Structure:

```
$GPDTM,LLL,LSD,lat,N/S,lon,E/W,alt,RRR*cs<CR><LF>
```

Example:

```
$GPDTM,W84,,0.0,N,0.0,E,0.0,W84*6F
$GPDTM,W72,,0.00,S,0.01,W,-2.8,W84*4F
$GPDTM,999,CH95,0.08,N,0.07,E,-47.7,W84*1C
```

Field No.	*Example*	*Format*	*Name*	*Unit*	*Description*
0	$GPDTM	string	`$GPDTM`	-	Message ID, DTM protocol header
1	W72	string	`LLL`	-	Local Datum Code, W84 = WGS84, W72 = WGS72, 999 = user defined
2	-	string	`LSD`	-	Local Datum Subdivision Code, This field outputs the currently selected Datum as a string (see also note above).
3	0.08	numeric	`lat`	minutes	Offset in Latitude
4	S	character	`NS`	-	North/South indicator
5	0.07	numeric	`lon`	minutes	Offset in Longitude
6	E	character	`EW`	-	East/West indicator
7	-2.8	numeric	`alt`	m	Offset in altitude
8	W84	string	`RRR`	-	Reference Datum Code, W84 = WGS 84. This is the only supported Reference datum.
9	*67	hexadecimal	`cs`	-	Checksum
10	-	character	`<CR><LF>`	-	Carriage Return and Line Feed

GPQ

Message	**GPQ**	
Description	**Poll message**	
Type	Input Message	
Comment	Polls a standard NMEA message.	
	ID for CFG-MSG	*Number of fields*
Message Info	0xF0 0x40	4

Message Structure:

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

Example:

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

Field No.	*Example*	*Format*	*Name*	*Unit*	*Description*
0	$EIGPQ	string	`$xxGPQ`	-	Message ID, GPQ protocol header, xx = talker identifier
1	RMC	string	`sid`	-	Sentence identifier
2	*3A	hexadecimal	`cs`	-	Checksum
3	-	character	`<CR><LF>`	-	Carriage Return and Line Feed

TXT

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

Message Structure:

```
$GPTXT,xx,yy,zz,ascii data*cs&lt;CR&gt;&lt;LF&gt;
```

Example:

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

Field No.	*Example*	*Format*	*Name*	*Unit*	*Description*
0	$GPTXT	string	`$GPTXT`	-	Message ID, TXT protocol header
1	01	numeric	`xx`	-	Total number of messages in this transmission, 01..99
2	01	numeric	`yy`	-	Message number in this transmission, range 01..xx
3	02	numeric	`zz`	-	Text identifier, u-blox GPS receivers specify the severity of the message with this number. - 00 = ERROR - 01 = WARNING - 02 = NOTICE - 07 = USER
4	www.u-blox.com	string	`string`	-	Any ASCII text
5	*67	hexadecimal	`cs`	-	Checksum
6	-	character	`<CR><LF>`	-	Carriage Return and Line Feed

Proprietary Messages

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

UBX,00

Message	**UBX,00**		
Description	**Lat/Long Position Data**		
Type	Output Message		
Comment	**The output of this message is dependent on the currently selected datum (Default: WGS84)** This message contains position solution data. The datum selection may be changed using the message CFG-DAT.		
	ID for CFG-MSG	*Number of fields*	
Message Info	0xF1 0x00	23	

Message Structure:

```
$PUBX,00,hhmmss.ss,Latitude,N,Longitude,E,AltRef,NavStat,Hacc,Vacc,SOG,COG,Vvel,ageC,HDOP,VDOP,TDOP
,GU,RU,DR,*cs<CR><LF>
```

Example:

```
$PUBX,00,081350.00,4717.113210,N,00833.915187,E,546.589,G3,2.1,2.0,0.007,77.52,0.007,,0.92,1.19,0.7
7,9,0,0*5F
```

Field No.	*Example*	*Format*	*Name*	*Unit*	*Description*
0	$PUBX	string	`$PUBX`	-	Message ID, UBX protocol header, proprietary sentence
1	00	numeric	`ID`	-	Propietary message identifier: 00
2	081350.00	hhmmss.sss	`hhmmss.ss`	-	UTC Time, Current time
3	4717.113210	ddmm.mmmm	`Latitude`	-	Latitude, Degrees + minutes, see Format description
4	N	character	`N`	-	N/S Indicator, N=north or S=south
5	00833.915187	dddmm.mmmm	`Longitude`	-	Longitude, Degrees + minutes, see Format description
6	E	character	`E`	-	E/W indicator, E=east or W=west
7	546.589	numeric	`AltRef`	m	Altitude above user datum ellipsoid.
8	G3	string	`NavStat`	-	Navigation Status, See Table below
9	2.1	numeric	`Hacc`	m	Horizontal accuracy estimate.
10	2.0	numeric	`Vacc`	m	Vertical accuracy estimate.
11	0.007	numeric	`SOG`	km/h	Speed over ground
12	77.52	numeric	`COG`	degrees	Course over ground
13	0.007	numeric	`Vvel`	m/s	Vertical velocity, positive=downwards
14	-	numeric	`ageC`	s	Age of most recent DGPS corrections, empty = none available
15	0.92	numeric	`HDOP`	-	HDOP, Horizontal Dilution of Precision

UBX,00 continued

Field No.	*Example*	*Format*	*Name*	*Unit*	*Description*
16	1.19	numeric	VDOP	-	VDOP, Vertical Dilution of Precision
17	0.77	numeric	TDOP	-	TDOP, Time Dilution of Precision
18	9	numeric	GU	-	Number of GPS satellites used in the navigation solution
19	0	numeric	RU	-	Number of GLONASS satellites used in the navigation solution
20	0	numeric	DR	-	DR used
21	*5B	hexadecimal	cs	-	Checksum
22	-	character	<CR><LF>	-	Carriage Return and Line Feed

Table Navigation Status

Navigation Status	*Description*
NF	No Fix
DR	Predictive Dead Reckoning Solution
G2	Stand alone 2D solution
G3	Stand alone 3D solution
D2	Differential 2D solution
D3	Differential 3D solution

UBX,03

Message	**UBX,03**		
Description	**Satellite Status**		
Type	Output Message		
Comment	The PUBX,03 message contains satellite status information.		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF1 0x03	5 + 6*GT	

Message Structure:

```
$PUBX,03,GT{,SVID,s,AZM,EL,SN,LK},*cs<CR><LF>
```

Example:

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

Field No.	*Example*	*Format*	*Name*	*Unit*	*Description*
0	$PUBX	string	`$PUBX`	-	Message ID, UBX protocol header, proprietary sentence
1	03	numeric	`ID`	-	Propietary message identifier: 03
2	11	numeric	`GT`	-	Number of GPS satellites tracked
Start of repeated block (GT times)					
3 + 6*N	23	numeric	`SVID`	-	Satellite PRN number
4 + 6*N	-	character	`s`	-	Satellite status, see table below
5 + 6*N	-	numeric	`AZM`	degrees	Satellite azimuth, range 000..359
6 + 6*N	-	numeric	`EL`	degrees	Satellite elevation, range 00..90
7 + 6*N	45	numeric	`SN`	dBHz	Signal to noise ratio, range 00..55
8 + 6*N	010	numeric	`LK`	s	Satellite carrier lock time, range 00..255 0 = code lock only 255 = lock for 255 seconds or more
End of repeated block					
3 + 6*GT	*0D	hexadecimal	`cs`	-	Checksum
4 + 6*GT	-	character	`<CR><LF>`	-	Carriage Return and Line Feed

Table Satellite Status

Satellite Status	*Description*
-	Not used
U	Used in solution
e	Available for navigation, but no ephemeris

UBX,04

Message	**UBX,04**		
Description	**Time of Day and Clock Information**		
Type	Output Message		
Comment	-		
	ID for CFG-MSG	Number of fields	
Message Info	0xF1 0x04	12	

Message Structure:

```
$PUBX,04,hhmmss.ss,ddmmyy,UTC_TOW,week,reserved,Clk_B,Clk_D,PG,*cs<CR><LF>
```

Example:

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

Field No.	Example	Format	Name	Unit	Description
0	$PUBX	string	$PUBX	-	Message ID, UBX protocol header, proprietary sentence
1	04	numeric	ID	-	Propietary message identifier: 04
2	073731.00	hhmmss.sss	hhmmss.ss	-	UTC Time, Current time in hour, minutes, seconds
3	091202	ddmmyy	ddmmyy	-	UTC Date, day, month, year format
4	113851.00	numeric	UTC_TOW	s	UTC Time of Week
5	1196	numeric	week	-	GPS week numer, continues beyond 1023
6	113851.00	numeric	reserved	-	reserved, for future use
7	1930035	numeric	Clk_B	ns	Receiver clock bias
8	-2660.664	numeric	Clk_D	ns/s	Receiver clock drift
9	43	numeric	PG	ns	Timepulse Granularity, The quantization error of the Timepulse pin
10	*3C	hexadecimal	cs	-	Checksum
11	-	character	<CR><LF>	-	Carriage Return and Line Feed

UBX

Message	**UBX**		
Description	**Poll a PUBX message**		
Type	Input Message		
Comment	A PUBX is message is polled by sending the PUBX message without any data fields.		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF1 0x40	4	

Message Structure:

```
$PUBX,xx*cs<CR><LF>
```

Example:

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

Field No.	*Example*	*Format*	*Name*	*Unit*	*Description*
0	$PUBX	string	`$PUBX`	-	Message ID, UBX protocol header, proprietary sentence
1	04	numeric	`MsgID`	-	Requested PUBX message identifier
2	*37	hexadecimal	`cs`	-	Checksum
3	-	character	`<CR><LF>`	-	Carriage Return and Line Feed

UBX,40

Message	**UBX,40**	
Description	**Set NMEA message output rate**	
Type	Set Message	
Comment	Set/Get message rate configuration (s) to/from the receiver. • Send rate is relative to the event a message is registered on. For example, if the rate of a navigation message is set to 2, the message is sent every second navigation solution.	
	ID for CFG-MSG	*Number of fields*
Message Info	0xF1 0x40	11

Message Structure:

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

Example:

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

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

UBX,41

Message	**UBX,41**		
Description	**Set Protocols and Baudrate**		
Type	Set Message		
Comment	-		
Message Info	*ID for CFG-MSG*	*Number of fields*	
	0xF1 0x41	9	

Message Structure:

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

Example:

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

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

UBX Protocol

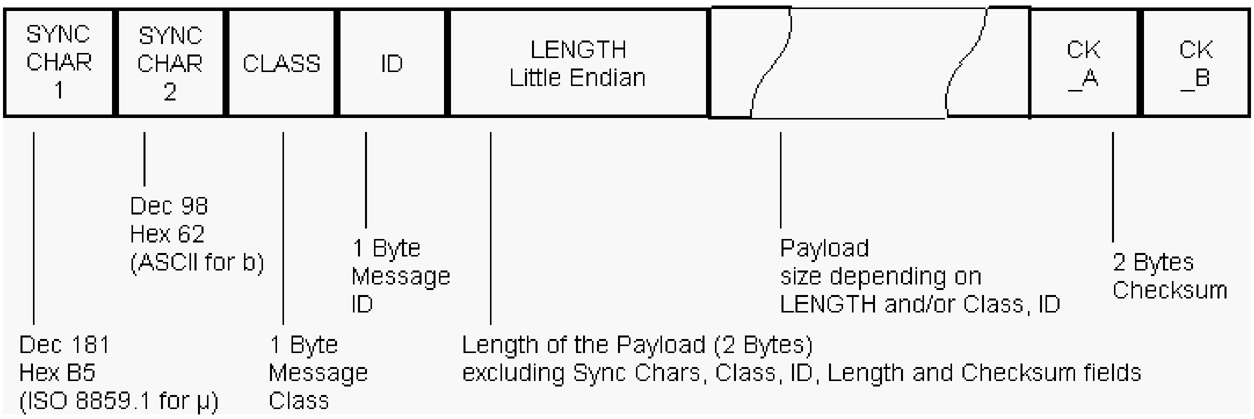
UBX Protocol Key Features

u-blox GPS receivers use a u-blox proprietary protocol to transmit GPS data to a host computer using asynchronous RS232 ports. 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)

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.

UBX Class IDs

A Class is a grouping of messages which are related to each other. The following table gives the short names, description and Class ID Definitions.

Name	*Class*	*Description*
NAV	0x01	Navigation Results: Position, Speed, Time, Acc, Heading, DOP, SVs used
RXM	0x02	Receiver Manager Messages: Satellite Status, RTC Status
INF	0x04	Information Messages: Printf-Style Messages, with IDs such as Error, Warning, Notice
ACK	0x05	Ack/Nack Messages: as replies to CFG Input Messages
CFG	0x06	Configuration Input Messages: Set Dynamic Model, Set DOP Mask, Set Baud Rate, etc.
MON	0x0A	Monitoring Messages: Comunication Status, CPU Load, Stack Usage, Task Status

UBX Class IDs continued

Name	Class	Description
AID	0x0B	AssistNow Aiding Messages: Ephemeris, Almanac, other A-GPS data input
TIM	0x0D	Timing Messages: Timepulse Output, Timemark Results

All remaining class IDs are reserved.

UBX Payload Definition Rules

Structure Packing

Values are placed in an order that structure packing is not a problem. This means that 2Byte values shall start on offsets which are a multiple of 2, 4-byte values shall start at a multiple of 4, and so on. This can easily be achieved by placing the largest values first in the Message payload (e.g. R8), and ending with the smallest (i.e. one-byters such as U1) values.

Message Naming

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

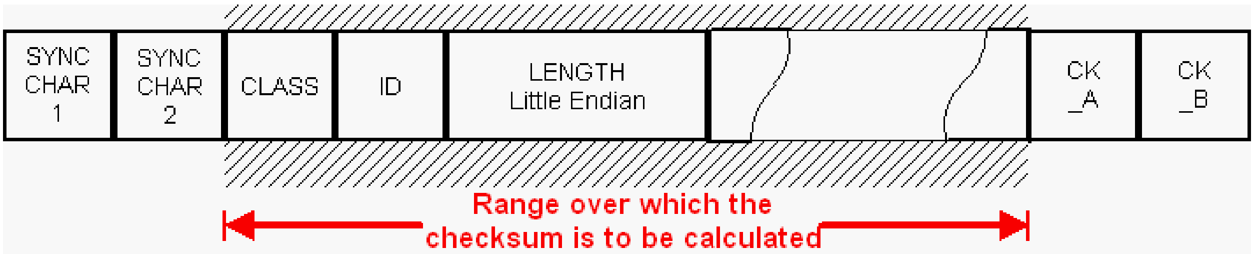
Number Formats

All multi-byte values are ordered in Little Endian format, unless otherwise indicated.
All floating point values are transmitted in IEEE754 single or double precision. A technical description of the IEEE754 format can be found in the AnswerBook from the ADS1.x toolkit.
The following table gives information about the various values:

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

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.

UBX Message Flow

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

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.
There is no ACK/NAK mechanism for message poll requests outside Class CFG.

Polling Mechanism

All messages that are output by the receiver in a periodic manner (i.e. Messages in Classes MON, NAV and RXM) can also be polled.
There is not a single specific message which polls any other message. The UBX protocol was designed such, that when sending a message with no payload (or just a single parameter which identifies the poll request) the message is polled.

UBX Messages Overview

Page	Mnemonic	Cls/ID	Length	Type	Description
UBX Class ACK				**Ack/Nack Messages**	
82	**ACK-ACK**	0x05 0x01	2	Answer	Message Acknowledged
82	**ACK-NAK**	0x05 0x00	2	Answer	Message Not-Acknowledged
UBX Class AID				**AssistNow Aiding Messages**	
121	**AID-ALM**	0x0B 0x30	0	Poll Request	Poll GPS Aiding Almanach Data
122	**AID-ALM**	0x0B 0x30	1	Poll Request	Poll GPS Aiding Almanach Data for a SV
122	**AID-ALM**	0x0B 0x30	(8) or (40)	Input/Output Message	GPS Aiding Almanach Input/Output Message
124	**AID-ALPSRV**	0x0B 0x32	16	Output Message	ALP client requests AlmanacPlus data from server
125	**AID-ALPSRV**	0x0B 0x32	16 + 1*dataSize	Input Message	ALP server sends AlmanacPlus data to client
125	**AID-ALPSRV**	0x0B 0x32	8 + 2*size	Output Message	ALP client sends AlmanacPlus data to server.
126	**AID-ALP**	0x0B 0x50	0 + 2*Variable	Input message	ALP file data transfer to the receiver
127	**AID-ALP**	0x0B 0x50	1	Input message	Mark end of data transfer
127	**AID-ALP**	0x0B 0x50	1	Output message	Acknowledges a data transfer
128	**AID-ALP**	0x0B 0x50	1	Output message	Indicate problems with a data transfer
128	**AID-ALP**	0x0B 0x50	24	Periodic/Polled	Poll the AlmanacPlus status
121	**AID-DATA**	0x0B 0x10	0	Poll	Polls all GPS Initial Aiding Data
123	**AID-EPH**	0x0B 0x31	0	Poll Request	Poll GPS Aiding Ephemeris Data
123	**AID-EPH**	0x0B 0x31	1	Poll Request	Poll GPS Aiding Ephemeris Data for a SV
123	**AID-EPH**	0x0B 0x31	(8) or (104)	Input/Output Message	GPS Aiding Ephemeris Input/Output Message
119	**AID-HUI**	0x0B 0x02	0	Poll Request	Poll GPS Health, UTC and ionosphere parameters
120	**AID-HUI**	0x0B 0x02	72	Input/Output Message	GPS Health, UTC and ionosphere parameters
117	**AID-INI**	0x0B 0x01	0	Poll Request	Poll GPS Initial Aiding Data
118	**AID-INI**	0x0B 0x01	48	Polled	Aiding position, time, frequency, clock drift
117	**AID-REQ**	0x0B 0x00	0	Virtual	Sends a poll (AID-DATA) for all GPS Aiding Data
UBX Class CFG				**Configuration Input Messages**	
101	**CFG-ANT**	0x06 0x13	0	Poll Request	Poll Antenna Control Settings
101	**CFG-ANT**	0x06 0x13	4	Get/Set	Get/Set Antenna Control Settings
99	**CFG-CFG**	0x06 0x09	(12) or (13)	Command	Clear, Save and Load configurations
95	**CFG-DAT**	0x06 0x06	0	Poll Request	Poll Datum Setting
95	**CFG-DAT**	0x06 0x06	2	Set	Set Standard Datum
95	**CFG-DAT**	0x06 0x06	44	Set	Set User-defined Datum
96	**CFG-DAT**	0x06 0x06	52	Get	Get currently selected Datum
92	**CFG-INF**	0x06 0x02	1	Poll Request	Poll INF message configuration for one protocol
93	**CFG-INF**	0x06 0x02	0 + 8*Num	Set/Get	Information message configuration
91	**CFG-MSG**	0x06 0x01	2	Poll Request	Poll a message configuration
91	**CFG-MSG**	0x06 0x01	8	Set/Get	Set Message Rate(s)
92	**CFG-MSG**	0x06 0x01	3	Set/Get	Set Message Rate

UBX Messages Overview continued

Page	*Mnemonic*	*Cls/ID*	*Length*	*Type*	*Description*
109	**CFG-NAV5**	0x06 0x24	0	Poll Request	Poll Navigation Engine Settings
110	**CFG-NAV5**	0x06 0x24	36	Get/Set	Get/Set Navigation Engine Settings
108	**CFG-NAVX5**	0x06 0x23	0	Poll Request	Poll Navigation Engine Expert Settings
108	**CFG-NAVX5**	0x06 0x23	40	Get/Set	Get/Set Navigation Engine Expert Settings
104	**CFG-NMEA**	0x06 0x17	0	Poll Request	Poll the NMEA protocol configuration
104	**CFG-NMEA**	0x06 0x17	4	Set/Get	Set/Get the NMEA protocol configuration
83	**CFG-PRT**	0x06 0x00	0	Poll Request	Polls the configuration of the used I/O Port
83	**CFG-PRT**	0x06 0x00	1	Poll Request	Polls the configuration for one I/O Port
84	**CFG-PRT**	0x06 0x00	20	Get/Set	Get/Set Port Configuration for UART
85	**CFG-PRT**	0x06 0x00	20	Get/Set	Get/Set Port Configuration for USB Port
86	**CFG-PRT**	0x06 0x00	20	Get/Set	Get/Set Port Configuration for SPI Port
88	**CFG-PRT**	0x06 0x00	20	Get/Set	Get/Set Port Configuration for DDC Port
89	**CFG-PRT**	0x06 0x00	20	Get/Set	Get/Set Port Configuration for SPI Port
98	**CFG-RATE**	0x06 0x08	0	Poll Request	Poll Navigation/Measurement Rate Settings
98	**CFG-RATE**	0x06 0x08	6	Get/Set	Navigation/Measurement Rate Settings
94	**CFG-RST**	0x06 0x04	4	Command	Reset Receiver / Clear Backup Data Structures
100	**CFG-RXM**	0x06 0x11	2	Set/Get	RXM configuration
102	**CFG-SBAS**	0x06 0x16	8	Command	SBAS Configuration
107	**CFG-TMODE**	0x06 0x1D	0	Poll Request	Poll Time Mode Settings
107	**CFG-TMODE**	0x06 0x1D	28	Get/Set	Time Mode Settings
97	**CFG-TP**	0x06 0x07	0	Poll Request	Poll TimePulse Parameters
97	**CFG-TP**	0x06 0x07	20	Get/Set	Get/Set TimePulse Parameters
105	**CFG-USB**	0x06 0x1B	0	Poll Request	Poll a USB configuration
106	**CFG-USB**	0x06 0x1B	108	Get/Set	Get/Set USB Configuration
UBX Class INF				**Information Messages**	
81	**INF-DEBUG**	0x04 0x04	0 + 1*variable		ASCII String output, indicating debug output
79	**INF-ERROR**	0x04 0x00	0 + 1*variable		ASCII String output, indicating an error
80	**INF-NOTICE**	0x04 0x02	0 + 1*variable		ASCII String output, with informational contents
80	**INF-TEST**	0x04 0x03	0 + 1*variable		ASCII String output, indicating test output
79	**INF-WARNING**	0x04 0x01	0 + 1*variable		ASCII String output, indicating a warning
UBX Class MON				**Monitoring Messages**	
115	**MON-HW**	0x0A 0x09	68	Periodic/Polled	Hardware Status
112	**MON-IO**	0x0A 0x02	0 + 20*NPRT	Periodic/Polled	I/O Subsystem Status
113	**MON-MSGPP**	0x0A 0x06	120	Periodic/Polled	Message Parse and Process Status
114	**MON-RXBUF**	0x0A 0x07	24	Periodic/Polled	Receiver Buffer Status
114	**MON-TXBUF**	0x0A 0x08	28	Periodic/Polled	Transmitter Buffer Status
113	**MON-VER**	0x0A 0x04	40 + 30*Num	Answer to Poll	Receiver/Software Version
UBX Class NAV				**Navigation Results**	

UBX Messages Overview continued

Page	*Mnemonic*	*Cls/ID*	*Length*	*Type*	*Description*
73	**NAV-CLOCK**	0x01 0x22	20	Periodic/Polled	Clock Solution
68	**NAV-DOP**	0x01 0x04	18	Periodic/Polled	Dilution of precision
66	**NAV-POSECEF**	0x01 0x01	20	Periodic/Polled	Position Solution in ECEF
66	**NAV-POSLLH**	0x01 0x02	28	Periodic/Polled	Geodetic Position Solution
75	**NAV-SBAS**	0x01 0x32	12 + 12*cnt	Periodic/Polled	SBAS Status Data
69	**NAV-SOL**	0x01 0x06	52	Periodic/Polled	Navigation Solution Information
67	**NAV-STATUS**	0x01 0x03	16	Periodic/Polled	Receiver Navigation Status
73	**NAV-SVINFO**	0x01 0x30	8 + 12*numCh	Periodic/Polled	Space Vehicle Information
71	**NAV-TIMEGPS**	0x01 0x20	16	Periodic/Polled	GPS Time Solution
72	**NAV-TIMEUTC**	0x01 0x21	20	Periodic/Polled	UTC Time Solution
70	**NAV-VELECEF**	0x01 0x11	20	Periodic/Polled	Velocity Solution in ECEF
71	**NAV-VELNED**	0x01 0x12	36	Periodic/Polled	Velocity Solution in NED
UBX Class RXM				**Receiver Manager Messages**	
77	**RXM-SVSI**	0x02 0x20	8 + 6*numSV	Periodic/Polled	SV Status Info
UBX Class TIM				**Timing Messages**	
131	**TIM-SVIN**	0x0D 0x04	28	Periodic/Polled	Survey-in data
130	**TIM-TM2**	0x0D 0x03	28	Periodic/Polled	Time mark data
129	**TIM-TP**	0x0D 0x01	16	Periodic/Polled	Timepulse Timedata

NAV (0x01)

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

NAV-POSECEF (0x01 0x01)

Position Solution in ECEF

Message	**NAV-POSECEF**				
Description	**Position Solution in ECEF**				
Type	Periodic/Polled				
Comment	-				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x01 0x01	20	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS Millisecond Time of Week
4	I4	-	`ecefX`	cm	ECEF X coordinate
8	I4	-	`ecefY`	cm	ECEF Y coordinate
12	I4	-	`ecefZ`	cm	ECEF Z coordinate
16	U4	-	`pAcc`	cm	Position Accuracy Estimate

NAV-POSLLH (0x01 0x02)

Geodetic Position Solution

Message	**NAV-POSLLH**				
Description	**Geodetic Position Solution**				
Type	Periodic/Polled				
Comment	This message outputs the Geodetic position in the currently selected Ellipsoid. The default is the WGS84 Ellipsoid, but can be changed with the message CFG-DAT.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x01 0x02	28	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS Millisecond Time of Week
4	I4	1e-7	`lon`	deg	Longitude
8	I4	1e-7	`lat`	deg	Latitude
12	I4	-	`height`	mm	Height above Ellipsoid
16	I4	-	`hMSL`	mm	Height above mean sea level
20	U4	-	`hAcc`	mm	Horizontal Accuracy Estimate
24	U4	-	`vAcc`	mm	Vertical Accuracy Estimate

NAV-STATUS (0x01 0x03)

Receiver Navigation Status

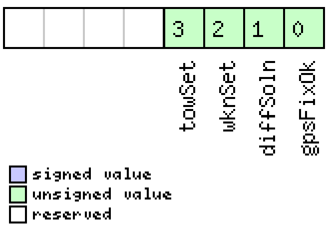
Message	**NAV-STATUS**				
Description	**Receiver Navigation Status**				
Type	Periodic/Polled				
Comment	-				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x01 0x03	16	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS Millisecond Time of Week
4	U1	-	`gpsFix`	-	GPSfix Type - 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	-	`diffStat`	-	Differential Status (see graphic below)
7	U1	-	`res`	-	Reserved
8	U4	-	`ttff`	-	Time to first fix (millisecond time tag)
12	U4	-	`msss`	-	Milliseconds since Startup / Reset

Bitfield flags

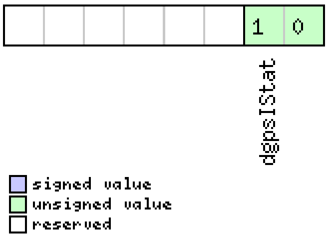
This Graphic explains the bits of `flags`



Name	*Description*
`gpsFixOk`	within DOP and ACC Masks
`diffSoln`	1 if DGPS used
`wknSet`	1 if Week Number valid
`towSet`	1 if Time of Week valid

Bitfield diffStat

This Graphic explains the bits of `diffStat`



Name	Description
`dgpsIStat`	DGPS Input Status 00: none 01: PR+PRR Correction 10: PR+PRR+CP Correction 11: High accuracy PR+PRR+CP Correction

NAV-DOP (0x01 0x04)

Dilution of precision

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

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS Millisecond Time of Week
4	U2	0.01	`gDOP`	-	Geometric DOP
6	U2	0.01	`pDOP`	-	Position DOP
8	U2	0.01	`tDOP`	-	Time DOP
10	U2	0.01	`vDOP`	-	Vertical DOP
12	U2	0.01	`hDOP`	-	Horizontal DOP
14	U2	0.01	`nDOP`	-	Northing DOP
16	U2	0.01	`eDOP`	-	Easting DOP

NAV-SOL (0x01 0x06)

Navigation Solution Information

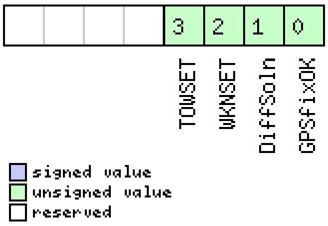
Message	**NAV-SOL**				
Description	**Navigation Solution Information**				
Type	Periodic/Polled				
Comment	This message combines Position, velocity and time solution in ECEF, including accuracy figures				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x01 0x06	52	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS Millisecond Time of Week
4	I4	-	`fTOW`	ns	Fractional Nanoseconds remainder of rounded ms above, range -500000 .. 500000
8	I2	-	`week`	-	GPS week (GPS time)
10	U1	-	`gpsFix`	-	GPSfix Type, range 0..4 0x00 = No Fix 0x01 = Dead Reckoning only 0x02 = 2D-Fix 0x03 = 3D-Fix 0x04 = GPS + dead reckoning combined 0x05 = Time only fix 0x06..0xff: reserved
11	X1	-	`flags`	-	Fix Status Flags (see graphic below)
12	I4	-	`ecefX`	cm	ECEF X coordinate
16	I4	-	`ecefY`	cm	ECEF Y coordinate
20	I4	-	`ecefZ`	cm	ECEF Z coordinate
24	U4	-	`pAcc`	cm	3D Position Accuracy Estimate
28	I4	-	`ecefVX`	cm/s	ECEF X velocity
32	I4	-	`ecefVY`	cm/s	ECEF Y velocity
36	I4	-	`ecefVZ`	cm/s	ECEF Z velocity
40	U4	-	`sAcc`	cm/s	Speed Accuracy Estimate
44	U2	0.01	`pDOP`	-	Position DOP
46	U1	-	`res1`	-	reserved
47	U1	-	`numSV`	-	Number of SVs used in Nav Solution
48	U4	-	`res2`	-	reserved

Bitfield flags

This Graphic explains the bits of `flags`



Name	Description
`GPSfixOK`	i.e within DOP & ACC Masks
`DiffSoln`	1 if DGPS used
`WKNSET`	1 if Week Number valid
`TOWSET`	1 if Time of Week valid

NAV-VELECEF (0x01 0x11)

Velocity Solution in ECEF

Message	**NAV-VELECEF**				
Description	**Velocity Solution in ECEF**				
Type	Periodic/Polled				
Comment	-				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x01 0x11	20	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS Millisecond Time of Week
4	I4	-	`ecefVX`	cm/s	ECEF X velocity
8	I4	-	`ecefVY`	cm/s	ECEF Y velocity
12	I4	-	`ecefVZ`	cm/s	ECEF Z velocity
16	U4	-	`sAcc`	cm/s	Speed Accuracy Estimate

NAV-VELNED (0x01 0x12)

Velocity Solution in NED

Message	**NAV-VELNED**				
Description	**Velocity Solution in NED**				
Type	Periodic/Polled				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x12	36	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS Millisecond Time of Week
4	I4	-	`velN`	cm/s	NED north velocity
8	I4	-	`velE`	cm/s	NED east velocity
12	I4	-	`velD`	cm/s	NED down velocity
16	U4	-	`speed`	cm/s	Speed (3-D)
20	U4	-	`gSpeed`	cm/s	Ground Speed (2-D)
24	I4	1e-5	`heading`	deg	Heading 2-D
28	U4	-	`sAcc`	cm/s	Speed Accuracy Estimate
32	U4	1e-5	`cAcc`	deg	Course / Heading Accuracy Estimate

NAV-TIMEGPS (0x01 0x20)

GPS Time Solution

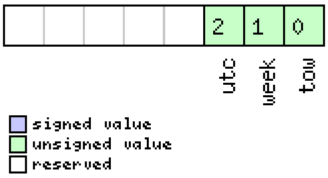
Message	**NAV-TIMEGPS**				
Description	**GPS Time Solution**				
Type	Periodic/Polled				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x20	16	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS Millisecond time of Week
4	I4	-	`fTOW`	ns	Fractional Nanoseconds remainder of rounded ms above, range -500000 .. 500000
8	I2	-	`week`	-	GPS week (GPS time)
10	I1	-	`leapS`	s	Leap Seconds (GPS-UTC)
11	X1	-	`valid`	-	Validity Flags (see graphic below)
12	U4	-	`tAcc`	ns	Time Accuracy Estimate

Bitfield valid

This Graphic explains the bits of `valid`



Name	*Description*
`tow`	1=Valid Time of Week
`week`	1=Valid Week Number
`utc`	1=Valid Leap Seconds, i.e. Leap Seconds already known

NAV-TIMEUTC (0x01 0x21)

UTC Time Solution

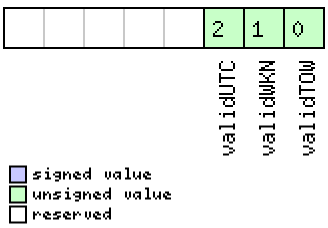
Message	**NAV-TIMEUTC**				
Description	**UTC Time Solution**				
Type	Periodic/Polled				
Comment	-				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x01 0x21	20	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS Millisecond Time of Week
4	U4	-	`tAcc`	ns	Time Accuracy Estimate
8	I4	-	`nano`	ns	Nanoseconds of second, range -500000000 .. 500000000 (UTC)
12	U2	-	`year`	y	Year, range 1999..2099 (UTC)
14	U1	-	`month`	month	Month, range 1..12 (UTC)
15	U1	-	`day`	d	Day of Month, range 1..31 (UTC)
16	U1	-	`hour`	h	Hour of Day, range 0..23 (UTC)
17	U1	-	`min`	min	Minute of Hour, range 0..59 (UTC)
18	U1	-	`sec`	s	Seconds of Minute, range 0..59 (UTC)
19	X1	-	`valid`	-	Validity Flags (see graphic below)

Bitfield valid

This Graphic explains the bits of `valid`



Name	*Description*
`validTOW`	1 = Valid Time of Week

Bitfield valid Description continued

Name	Description
validWKN	1 = Valid Week Number
validUTC	1 = Valid UTC (Leap Seconds already known)

NAV-CLOCK (0x01 0x22)

Clock Solution

Message	**NAV-CLOCK**				
Description	**Clock Solution**				
Type	Periodic/Polled				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x22	20	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	iTOW	ms	GPS Millisecond Time of week
4	I4	-	clkB	ns	Clock bias in nanoseconds
8	I4	-	clkD	ns/s	Clock drift in nanoseconds per second
12	U4	-	tAcc	ns	Time Accuracy Estimate
16	U4	-	fAcc	ps/s	Frequency Accuracy Estimate

NAV-SVINFO (0x01 0x30)

Space Vehicle Information

Message	**NAV-SVINFO**				
Description	**Space Vehicle Information**				
Type	Periodic/Polled				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x01 0x30	8 + 12*numCh	*see below*	CK_A CK_B

Payload Contents:

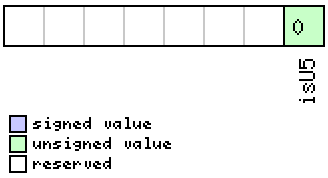
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	iTOW	ms	GPS Millisecond time of week
4	U1	-	numCh	-	Number of channels
5	X1	-	globalFlags	-	Bitmask (see graphic below)
6	U2	-	res2	-	Reserved
Start of repeated block (numCh times)					
8 + 12*N	U1	-	chn	-	Channel number
9 + 12*N	U1	-	svid	-	Satellite ID
10 + 12*N	X1	-	flags	-	Bitmask (see graphic below)

NAV-SVINFO continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
11 + 12*N	X1	-	`quality`	-	Bitfield (see graphic below)
12 + 12*N	U1	-	`cno`	dbHz	Carrier to Noise Ratio (Signal Strength)
13 + 12*N	I1	-	`elev`	deg	Elevation in integer degrees
14 + 12*N	I2	-	`azim`	deg	Azimuth in integer degrees
16 + 12*N	I4	-	`prRes`	cm	Pseudo range residual in centimetres
End of repeated block					

Bitfield globalFlags

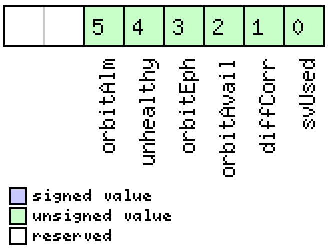
This Graphic explains the bits of `globalFlags`



Name	*Description*
`isU5`	u-blox 5 generation receiver

Bitfield flags

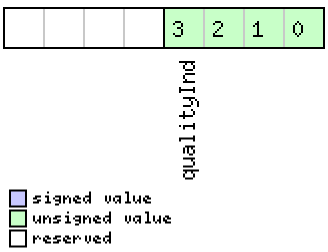
This Graphic explains the bits of `flags`



Name	*Description*
`svUsed`	SV is used for navigation
`diffCorr`	Differential correction data is available for this SV
`orbitAvail`	Orbit information is available for this SV (Ephemeris or Almanach)
`orbitEph`	Orbit information is Ephemeris
`unhealthy`	SV is unhealthy / shall not be used
`orbitAlm`	Orbit information is Almanac Plus

Bitfield quality

This Graphic explains the bits of `quality`



Name	*Description*

Bitfield quality Description continued

Name	*Description*
`qualityInd`	Signal Quality indicator (range 0..7). The following list shows the meaning of the different QI values: 0: This channel is idle 1: Channel is searching 2: Signal aquired 3: Signal detected but unusable 4: Code Lock on Signal 5, 6: Code and Carrier locked 7: Code and Carrier locked, receiving 50bps data

NAV-SBAS (0x01 0x32)

SBAS Status Data

Message	**NAV-SBAS**				
Description	**SBAS Status Data**				
Type	Periodic/Polled				
Comment	This message outputs the status of the SBAS sub system				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x01 0x32	12 + 12*cnt	*see below*	CK_A CK_B

Payload Contents:

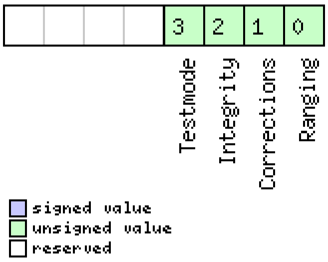
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`iTOW`	ms	GPS Millisecond time of week
4	U1	-	`geo`	-	PRN Number of the GEO where correction and integrity data is used from
5	U1	-	`mode`	-	SBAS Mode 0 Disabled 1 Enabled Integrity 3 Enabled Testmode
6	I1	-	`sys`	-	SBAS System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

NAV-SBAS continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
16 + 12*N	U1	-	`svService`	-	Services available same as SERVICE
17 + 12*N	U1	-	`res0`	-	Reserved
18 + 12*N	I2	-	`prc`	cm	Pseudo Range correction in [cm]
20 + 12*N	I2	-	`res1`	-	Reserved
22 + 12*N	I2	-	`ic`	cm	Ionosphere correction in [cm]
End of repeated block					

Bitfield service

This Graphic explains the bits of `service`



RXM (0x02)

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

RXM-SVSI (0x02 0x20)

SV Status Info

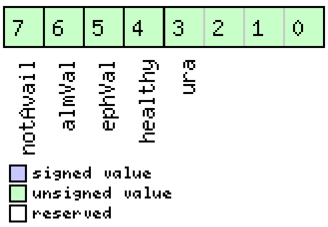
Message	**RXM-SVSI**				
Description	**SV Status Info**				
Type	Periodic/Polled				
Comment	Status of the receiver manager knowledge about GPS Orbit Validity				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x02 0x20	8 + 6*numSV	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	I4	-	`iTOW`	ms	Measurement integer millisecond GPS time of week
4	I2	-	`week`	weeks	Measurement GPS week number.
6	U1	-	`numVis`	-	Number of visible satellites
7	U1	-	`numSV`	-	Number of per-SV data blocks following
Start of repeated block (numSV times)					
8 + 6*N	U1	-	`svid`	-	Satellite ID
9 + 6*N	X1	-	`svFlag`	-	Information Flags (see graphic below)
10 + 6*N	I2	-	`azim`	-	Azimuth
12 + 6*N	I1	-	`elev`	-	Elevation
13 + 6*N	X1	-	`age`	-	Age of Almanach and Ephemeris: (see graphic below)
End of repeated block					

Bitfield svFlag

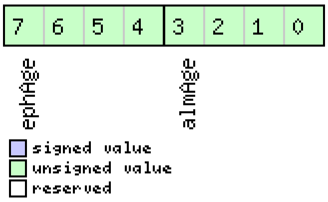
This Graphic explains the bits of `svFlag`



Name	*Description*
`ura`	Figure of Merit (URA) range 0..15
`healthy`	SV healthy flag
`ephVal`	Ephemeris valid
`almVal`	Almanach valid
`notAvail`	SV not available

Bitfield age

This Graphic explains the bits of `age`



Name	*Description*
`almAge`	Age of ALM in days offset by 4 i.e. the reference time may be in the future: ageOfAlm = (age & 0x0f) - 4
`ephAge`	Age of EPH in hours offset by 4. i.e. the reference time may be in the future: ageOfEph = ((age & 0xf0) >> 4) - 4

INF (0x04)

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

INF-ERROR (0x04 0x00)

ASCII String output, indicating an error

Message	**INF-ERROR**				
Description	**ASCII String output, indicating an error**				
Type					
Comment	This message has a variable length payload, representing an ASCII string.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x04 0x00	0 + 1*variable	*see below*	CK_A CK_B

Payload Contents:					
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
Start of repeated block (variable times)					
N*1	CH	-	char	-	ASCII Character
End of repeated block					

INF-WARNING (0x04 0x01)

ASCII String output, indicating a warning

Message	**INF-WARNING**				
Description	**ASCII String output, indicating a warning**				
Type					
Comment	This message has a variable length payload, representing an ASCII string.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x04 0x01	0 + 1*variable	*see below*	CK_A CK_B

Payload Contents:					
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
Start of repeated block (variable times)					
N*1	CH	-	char	-	ASCII Character
End of repeated block					

INF-NOTICE (0x04 0x02)

ASCII String output, with informational contents

Message	**INF-NOTICE**						
Description	**ASCII String output, with informational contents**						
Type							
Comment	This message has a variable length payload, representing an ASCII string.						
Message Structure	*Header*	*ID*	*Length (Bytes)*			*Payload*	*Checksum*
	0xB5 0x62	0x04 0x02	0 + 1*variable			see below	CK_A CK_B
Payload Contents:							
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*		
Start of repeated block (variable times)							
N*1	CH	-	char	-	ASCII Character		
End of repeated block							

INF-TEST (0x04 0x03)

ASCII String output, indicating test output

Message	**INF-TEST**						
Description	**ASCII String output, indicating test output**						
Type							
Comment	This message has a variable length payload, representing an ASCII string.						
Message Structure	*Header*	*ID*	*Length (Bytes)*			*Payload*	*Checksum*
	0xB5 0x62	0x04 0x03	0 + 1*variable			see below	CK_A CK_B
Payload Contents:							
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*		
Start of repeated block (variable times)							
N*1	CH	-	char	-	ASCII Character		
End of repeated block							

INF-DEBUG (0x04 0x04)

ASCII String output, indicating debug output

Message	**INF-DEBUG**				
Description	**ASCII String output, indicating debug output**				
Type					
Comment	This message has a variable length payload, representing an ASCII string.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x04 0x04	0 + 1*variable	*see below*	CK_A CK_B

Payload Contents:

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

ACK (0x05)

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

ACK-NAK (0x05 0x00)

Message Not-Acknowledged

Message	**ACK-NAK**				
Description	**Message Not-Acknowledged**				
Type	Answer				
Comment	Output upon processing of an input message				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x05 0x00	2	*see below*	CK_A CK_B

Payload Contents:

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

ACK-ACK (0x05 0x01)

Message Acknowledged

Message	**ACK-ACK**				
Description	**Message Acknowledged**				
Type	Answer				
Comment	Output upon processing of an input message				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x05 0x01	2	*see below*	CK_A CK_B

Payload Contents:

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

CFG (0x06)

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

CFG-PRT (0x06 0x00)

Polls the configuration of the used I/O Port

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

Polls the configuration for one I/O Port

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

Payload Contents:

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

Get/Set Port Configuration for UART

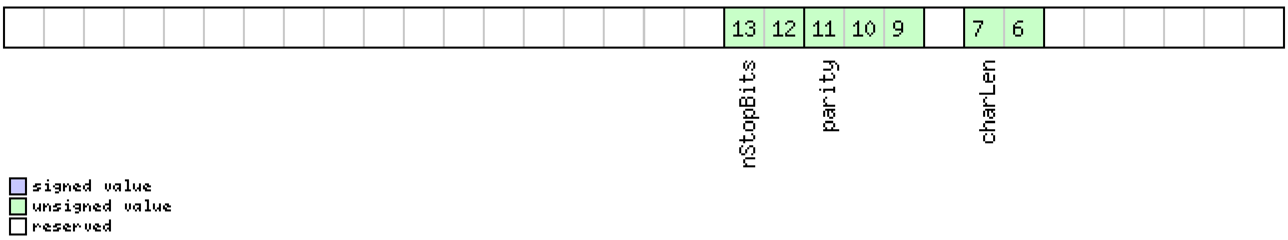
Message	**CFG-PRT**				
Description	**Get/Set Port Configuration for UART**				
Type	Get/Set				
Comment	Several configurations can be concatenated to one input message. In this case the payload length can be a multiple of the normal length (see the other versions of CFG-PRT). Output messages from the module contain only one configuration unit.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	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 (= 1 or 2 for UART ports)
1	U1	-	`res0`	-	Reserved
2	U2	-	`res1`	-	Reserved
4	X4	-	`mode`	-	A bit mask describing the UART mode (see graphic below)
8	U4	-	`baudRate`	Bits/s	Baudrate in bits/second
12	X2	-	`inProtoMask`	-	A mask describing which input protocols are active. Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port. (see graphic below)
14	X2	-	`outProtoMask`	-	A mask describing which output protocols are active. Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port. (see graphic below)
16	X2	-	`flags`	-	Reserved, set to 0
18	U2	-	`pad`	-	Reserved, set to 0

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

Bitfield mode Description continued

Name	*Description*
`parity`	000 Even Parity 001 Odd Parity 10X No Parity X1X Reserved
`nStopBits`	Number of Stop Bits 00 1 Stop Bit 01 1.5 Stop Bit 10 2 Stop Bit 11 0.5 Stop Bit

Bitfield inProtoMask

This Graphic explains the bits of `inProtoMask`

Bitfield outProtoMask

This Graphic explains the bits of `outProtoMask`

Get/Set Port Configuration for USB Port

Message	**CFG-PRT**				
Description	**Get/Set Port Configuration for USB Port**				
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.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	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	-	`res0`	-	Reserved
2	U2	-	`res1`	-	Reserved
4	U4	-	`res2`	-	Reserved
8	U4	-	`res3`	-	Reserved

CFG-PRT continued

Byte Offset	Number Format	Scaling	Name	Unit	Description
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`	-	Reserved, set to 0
18	U2	-	`pad`	-	Reserved, set to 0

Bitfield inProtoMask

This Graphic explains the bits of `inProtoMask`

Bitfield outProtoMask

This Graphic explains the bits of `outProtoMask`

Get/Set Port Configuration for SPI Port

Message	**CFG-PRT**				
Description	**Get/Set Port Configuration for SPI Port**				
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.				
	Header	ID	Length (Bytes)	Payload	Checksum
Message Structure	0xB5 0x62	0x06 0x00	20	see below	CK_A CK_B

Payload Contents:

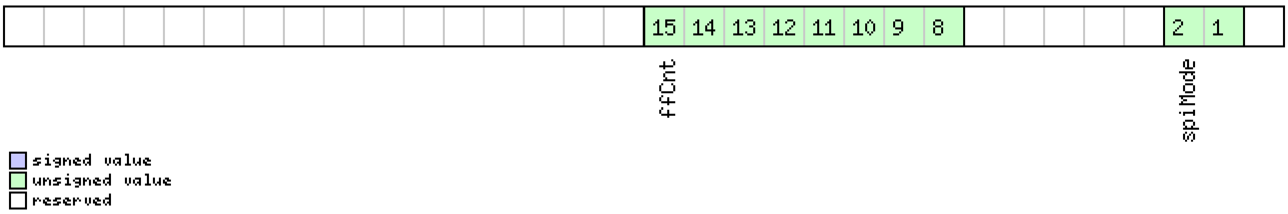
Byte Offset	Number Format	Scaling	Name	Unit	Description
0	U1	-	`portID`	-	Port Identifier Number (= 4 for SPI port)
1	U1	-	`res0`	-	Reserved

CFG-PRT continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
2	U2	-	`res1`	-	Reserved
4	X4	-	`mode`	-	SPI Mode Flags (see graphic below)
8	U4	-	`res2`	-	Reserved
12	X2	-	`inProtoMask`	-	A mask describing which input protocols are active. Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port. (see graphic below)
14	X2	-	`outProtoMask`	-	A mask describing which output protocols are active. Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port. (see graphic below)
16	X2	-	`flags`	-	Reserved, set to 0
18	U2	-	`pad`	-	Reserved, set to 0

Bitfield mode

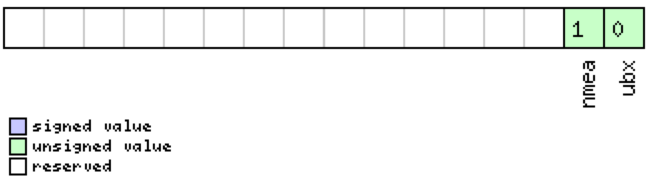
This Graphic explains the bits of `mode`



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
`ffCnt`	Number of bytes containing 0xFF to receive before switching off reception. Range: 0(mechanism off)-255

Bitfield inProtoMask

This Graphic explains the bits of `inProtoMask`



Bitfield outProtoMask

This Graphic explains the bits of `outProtoMask`

Get/Set Port Configuration for DDC Port

Message	**CFG-PRT**
Description	**Get/Set Port Configuration for DDC Port**
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.

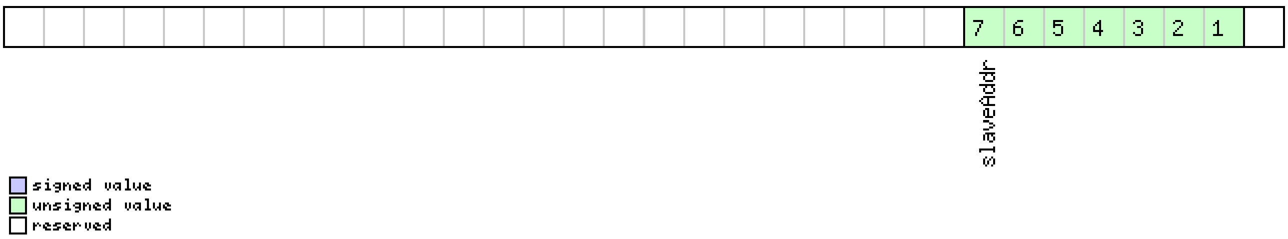
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x06 0x00	20	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`portID`	-	Port Identifier Number (= 0 for DDC port)
1	U1	-	`res0`	-	Reserved
2	U2	-	`res1`	-	Reserved
4	X4	-	`mode`	-	DDC Mode Flags (see graphic below)
8	U4	-	`res2`	-	Reserved
12	X2	-	`inProtoMask`	-	A mask describing which input protocols are active. Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port. (see graphic below)
14	X2	-	`outProtoMask`	-	A mask describing which output protocols are active. Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port. (see graphic below)
16	X2	-	`flags`	-	Reserved, set to 0
18	U2	-	`pad`	-	Reserved, set to 0

Bitfield mode

This Graphic explains the bits of `mode`



Bitfield mode Description continued

Name	*Description*
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`

Get/Set Port Configuration for SPI Port

Message	**CFG-PRT**				
Description	**Get/Set Port Configuration for SPI Port**				
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.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x06 0x00	20	*see below*	CK_A CK_B

Payload Contents:

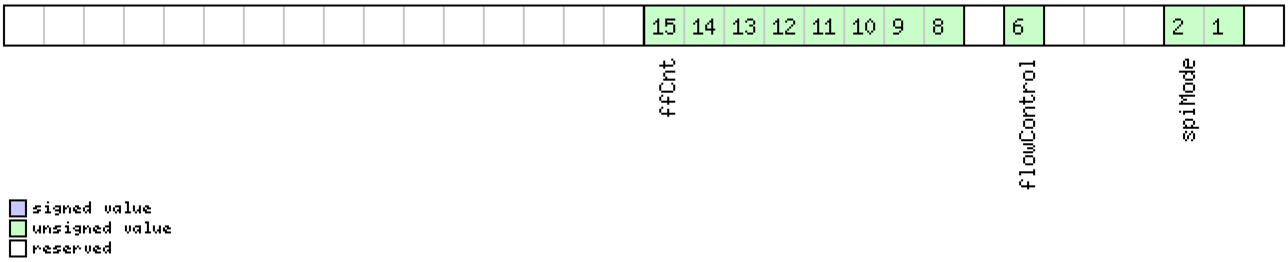
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`portID`	-	Port Identifier Number (= 4 for SPI port)
1	U1	-	`res0`	-	Reserved
2	U2	-	`res1`	-	Reserved
4	X4	-	`mode`	-	SPI Mode Flags (see graphic below)
8	U4	-	`res2`	-	Reserved
12	X2	-	`inProtoMask`	-	A mask describing which input 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)

CFG-PRT continued

Byte Offset	Number Format	Scaling	Name	Unit	Description
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`	-	Reserved, set to 0
18	U2	-	`pad`	-	Reserved, set to 0

Bitfield mode

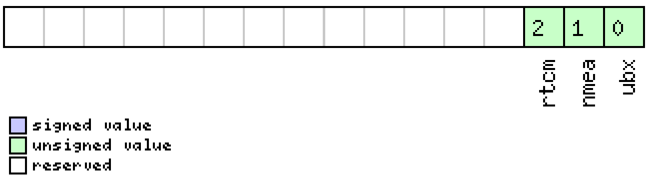
This Graphic explains the bits of `mode`



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`	0 Flow control disabled 1 Flow control enabled (9-bit mode)
`ffCnt`	Number of bytes containing 0xFF to receive before switching off reception. Range: 0(mechanism off)-255

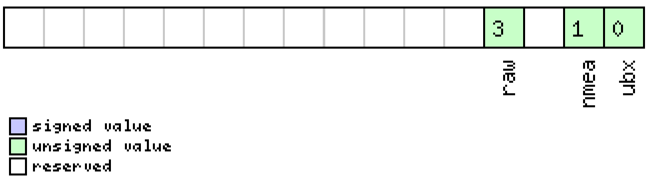
Bitfield inProtoMask

This Graphic explains the bits of `inProtoMask`



Bitfield outProtoMask

This Graphic explains the bits of `outProtoMask`



CFG-MSG (0x06 0x01)

Poll a message configuration

Message	**CFG-MSG**				
Description	**Poll a message configuration**				
Type	Poll Request				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x01	2	*see below*	CK_A CK_B

Payload Contents:

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

Set Message Rate(s)

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

Payload Contents:

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

Set Message Rate

Message	**CFG-MSG**					
Description	**Set Message Rate**					
Type	Set/Get					
Comment	Set message rate configuration for the current target. See also section How to change between protocols.					
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*	
	0xB5 0x62	0x06 0x01	3	*see below*	CK_A CK_B	
Payload Contents:						
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*	
0	U1	-	`class`	-	Message Class	
1	U1	-	`msgID`	-	Message Identifier	
2	U1	-	`rate`	-	Send rate on current Target	

CFG-INF (0x06 0x02)

Poll INF message configuration for one protocol

Message	**CFG-INF**					
Description	**Poll INF message configuration for one protocol**					
Type	Poll Request					
Comment	-					
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*	
	0xB5 0x62	0x06 0x02	1	*see below*	CK_A CK_B	
Payload Contents:						
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*	
0	U1	-	`protocolID`	-	Protocol Identifier, identifying the output protocol for this Poll Request. The following are valid Protocol Identifiers: - 0: UBX Protocol - 1: NMEA Protocol - 2-255: Reserved	

Information message configuration

Message	**CFG-INF**
Description	**Information message configuration**
Type	Set/Get
Comment	The value of INFMSG_mask<x> below are that each bit represents one of the INF class messages (Bit 0 for ERROR, Bit 1 for WARNING and so on.). For a complete list, please see the Message Class INF. Several configurations can be concatenated to one input message. In this case the payload length can be a multiple of the normal length. Output messages from the module contain only one configuration unit. Please note that I/O Targets 0, 1 and 2 correspond to serial ports 0, 1 and 2. I/O target 3 is reserved for future use.

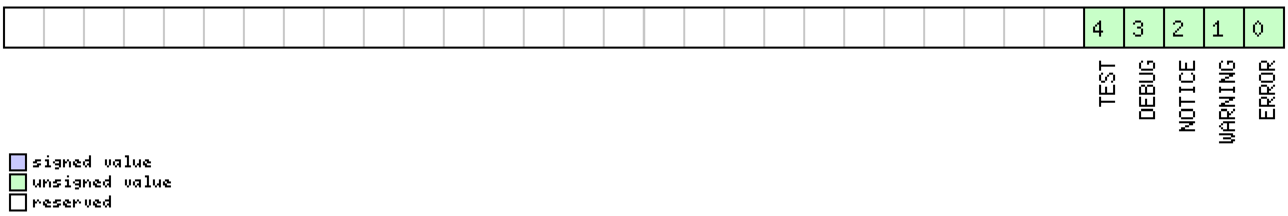
	Header	ID	Length (Bytes)	Payload	Checksum
Message Structure	0xB5 0x62	0x06 0x02	0 + 8*Num	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
Start of repeated block (Num times)					
N*8	U1	-	`protocolID`	-	Protocol Identifier, identifying for which protocol the configuration is set/get. The following are valid Protocol Identifiers: - 0: UBX Protocol - 1: NMEA Protocol - 2-255: Reserved
1 + 8*N	U1	-	`res0`	-	Reserved
2 + 8*N	U2	-	`res1`	-	Reserved
4 + 8*N	X1[4]	-	`infMsgMask`	-	A bit mask, saying which information messages are enabled on each I/O target (see graphic below)
End of repeated block					

Bitfield infMsgMask

This Graphic explains the bits of `infMsgMask`



CFG-RST (0x06 0x04)

Reset Receiver / Clear Backup Data Structures

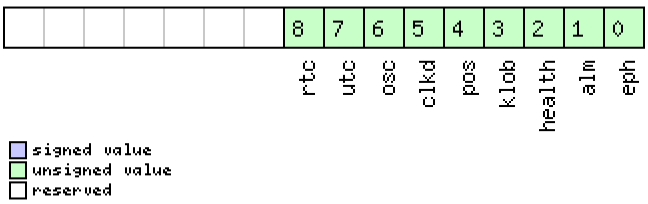
Message	**CFG-RST**				
Description	**Reset Receiver / Clear Backup Data Structures**				
Type	Command				
Comment	-				
Message Structure	Header	ID	Length (Bytes)	Payload	Checksum
	0xB5 0x62	0x06 0x04	4	see below	CK_A CK_B

Payload Contents:

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

Bitfield navBbrMask

This Graphic explains the bits of `navBbrMask`



Name	Description
`eph`	Ephemeris
`alm`	Almanach
`health`	Health
`klob`	Klobuchard
`pos`	Position
`clkd`	Clock Drift
`osc`	Oscilator Parameter
`utc`	UTC Correction Parameters
`rtc`	RTC

CFG-DAT (0x06 0x06)

Poll Datum Setting

Message	**CFG-DAT**				
Description	**Poll Datum Setting**				
Type	Poll Request				
Comment	Upon sending of this message, the receiver returns CFG-DAT as defined below				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x06	0	*see below*	CK_A CK_B
No payload					

Set Standard Datum

Message	**CFG-DAT**				
Description	**Set Standard Datum**				
Type	Set				
Comment	See section Geodetic Datums for a list of supported Datums				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x06	2	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U2	-	`datumNum`	-	Datum Number

Set User-defined Datum

Message	**CFG-DAT**				
Description	**Set User-defined Datum**				
Type	Set				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x06	44	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	R8	-	`majA`	m	Semi-major Axis (accepted range = 6,300,000.0 to 6,500,000.0 metres).
8	R8	-	`flat`	-	1.0 / Flattening (accepted range is 0.0 to 500.0).
16	R4	-	`dX`	m	X Axis shift at the origin (accepted range is +/- 5000.0 metres).
20	R4	-	`dY`	m	Y Axis shift at the origin (accepted range is +/- 5000.0 metres).

CFG-DAT continued

Byte Offset	Number Format	Scaling	Name	Unit	Description
24	R4	-	dZ	m	Z Axis shift at the origin (accepted range is +/- 5000.0 metres).
28	R4	-	rotX	s	Rotation about the X Axis (accepted range is +/- 20.0 milli-arc seconds).
32	R4	-	rotY	s	Rotation about the Y Axis (accepted range is +/- 20.0 milli-arc seconds).
36	R4	-	rotZ	s	Rotation about the Z Axis (accepted range is +/- 20.0 milli-arc seconds).
40	R4	-	scale	ppm	Scale change (accepted range is 0.0 to 50.0 parts per million).

Get currently selected Datum

Message	**CFG-DAT**				
Description	**Get currently selected Datum**				
Type	Get				
Comment	The Parameter datumName is only valid, if datumNum is not equal to -1. In case datumNum is -1,the receiver is configured for a custom datum. The parameters from majA to scale are valid for both custom or standard datum formats.				
	Header	ID	Length (Bytes)	Payload	Checksum
Message Structure	0xB5 0x62	0x06 0x06	52	see below	CK_A CK_B

Payload Contents:

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

CFG-TP (0x06 0x07)

Poll TimePulse Parameters

Message	**CFG-TP**				
Description	**Poll TimePulse Parameters**				
Type	Poll Request				
Comment	Sending this (empty / no-payload) message to the receiver results in the receiver returning a message of type CFG-TP with a payload as defined below				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x07	0	*see below*	CK_A CK_B
No payload					

Get/Set TimePulse Parameters

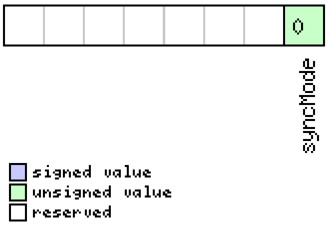
Message	**CFG-TP**				
Description	**Get/Set TimePulse Parameters**				
Type	Get/Set				
Comment	-				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x06 0x07	20	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`interval`	us	Time interval for time pulse
4	U4	-	`length`	us	Length of time pulse
8	I1	-	`status`	-	Time pulse config setting +1 = positive 0 = off -1 = negative
9	U1	-	`timeRef`	-	Alignment to reference time: 0 = UTC time, 1 = GPS time 2 = Local time
10	U1	-	`flags`	-	Bitmask (see graphic below)
11	U1	-	`res`	-	Reserved
12	I2	-	`antennaCableDelay`	ns	Antenna Cable Delay
14	I2	-	`rfGroupDelay`	ns	Receiver RF Group Delay
16	I4	-	`userDelay`	ns	User Time Function Delay (positive delay results in earlier pulse)

Bitfield flags

This Graphic explains the bits of `flags`



Name	Description
`syncMode`	0=Time pulse always synchronized and only available if time is valid 1=Time pulse allowed to be asynchronized and available even when time is not valid

CFG-RATE (0x06 0x08)

Poll Navigation/Measurement Rate Settings

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

Navigation/Measurement Rate Settings

Message	**CFG-RATE**				
Description	**Navigation/Measurement Rate Settings**				
Type	Get/Set				
Comment	The u-blox positioning technology supports navigation update rates higher or lower than 1 update per second. The calculation of the navigation solution will always be aligned to the top of a second. • The update rate has a direct influence on the power consumption. The more fixes that are required, the more CPU power and communication resources are required. • For most applications a 1 Hz update rate would be sufficient.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x06 0x08	6	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U2	-	`measRate`	ms	Measurement Rate, GPS measurements are taken every measRate milliseconds
2	U2	-	`navRate`	cycles	Navigation Rate, in number of measurement cycles. On u-blox 5, this parameter cannot be changed, and is always equals 1.

CFG-RATE continued

Byte Offset	Number Format	Scaling	Name	Unit	Description
4	U2	-	timeRef	-	Alignment to reference time: 0 = UTC time, 1 = GPS time

CFG-CFG (0x06 0x09)

Clear, Save and Load configurations

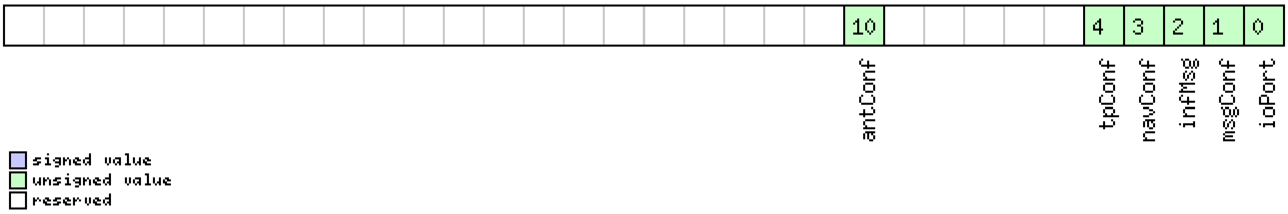
Message	CFG-CFG				
Description	**Clear, Save and Load configurations**				
Type	Command				
Comment	See the Receiver Configuration chapter for a detailed description on how Receiver Configuration should be used.The three masks are made up of individual bits, each bit indicating the sub-section of all configurations on which the corresponding action shall be carried out. Please note that commands can be combined. The sequence of execution is Clear, Save, Load				
	Header	ID	Length (Bytes)	Payload	Checksum
Message Structure	0xB5 0x62	0x06 0x09	(12) or (13)	see below	CK_A CK_B

Payload Contents:

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

Bitfield clearMask

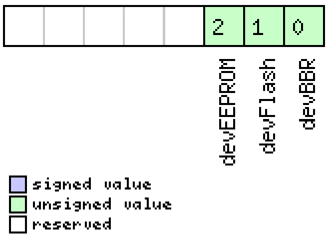
This Graphic explains the bits of `clearMask`



Name	Description
`ioPort`	I/O Port Assignements, Protocols and Baud Rates (See messages UBX-CFG-PRT and UBX-CFG-USB)
`msgConf`	Message Configuration (See message UBX-CFG-MSG)
`infMsg`	INF Message Configuration (See UBX-CFG-INF)
`navConf`	NAV Configuration (See UBX-CFG-DAT, UBX-CFG-NAV5, UBX-CFG-RATE, UBX-CFG-SBAS, UBX-CFG-NMEA, UBX-CFG-TMODE)
`tpConf`	Timepulse Configuration (See UBX-CFG-TP)
`antConf`	Used for Receiver Model-specific settings (e.g. UBX-CFG-ANT)

Bitfield deviceMask

This Graphic explains the bits of `deviceMask`



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

CFG-RXM (0x06 0x11)

RXM configuration

Message	**CFG-RXM**				
Description	**RXM configuration**				
Type	Set/Get				
Comment	This message is support with firmware 4.01 or later.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x06 0x11	2	see below	CK_A CK_B

Payload Contents:

Byte Offset	Number Format	Scaling	Name	Unit	Description
0	U1	-	`reserved`	-	reserved

CFG-RXM continued

Byte Offset	Number Format	Scaling	Name	Unit	Description
1	U1	-	`lpMode`	-	Low Power Mode 0: Max. performance mode 1-3: reserved 4: Eco mode 5-255: reserved

CFG-ANT (0x06 0x13)

Poll Antenna Control Settings

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

Get/Set Antenna Control Settings

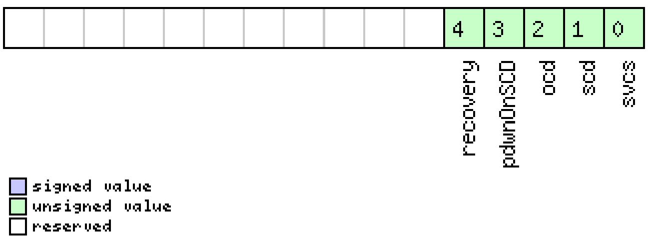
Message	CFG-ANT				
Description	**Get/Set Antenna Control Settings**				
Type	Get/Set				
Comment	-				
	Header	ID	Length (Bytes)	Payload	Checksum
Message Structure	0xB5 0x62	0x06 0x13	4	see below	CK_A CK_B

Payload Contents:

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

Bitfield flags

This Graphic explains the bits of `flags`



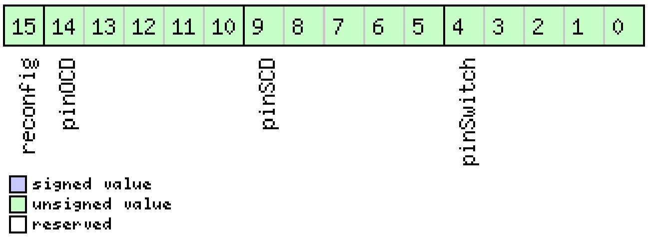
Name	Description

Bitfield flags Description continued

Name	Description
svcs	Enable Antenna Supply Voltage Control Signal
scd	Enable Short Circuit Detection
ocd	Enable Open Circuit Detection
pdwnOnSCD	Power Down Antenna supply if Short Circuit is detected. (only in combination with Bit 1)
recovery	Enable automatic recovery from short state

Bitfield pins

This Graphic explains the bits of `pins`



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

CFG-SBAS (0x06 0x16)

SBAS Configuration

Message	**CFG-SBAS**				
Description	**SBAS Configuration**				
Type	Command				
Comment	This message configures the SBAS receiver subsystem (i.e. WAAS, EGNOS, MSAS).See the SBAS Configuration Settings Description for a detailed description of how these settings affect receiver operation.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x06 0x16	8	*see below*	CK_A CK_B

Payload Contents:

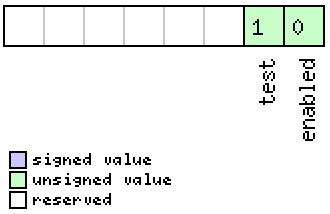
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	X1	-	mode	-	SBAS Mode (see graphic below)
1	X1	-	usage	-	SBAS Usage (see graphic below)
2	U1	-	maxSBAS	-	Maximum Number of SBAS prioritized tracking channels (valid range: 0 - 3) to use
3	X1	-	scanmode2	-	Continuation of scanmode bitmask below (see graphic below)

CFG-SBAS continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
4	X4	-	`scanmode1`	-	Which SBAS PRN numbers to search for (Bitmask) If all Bits are set to zero, auto-scan (i.e. all valid PRNs) are searched. Every bit corresponds to a PRN number (see graphic below)

Bitfield mode

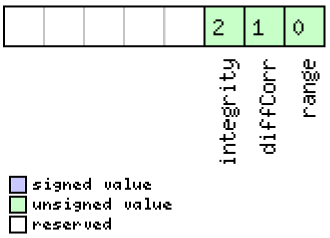
This Graphic explains the bits of `mode`



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

Bitfield usage

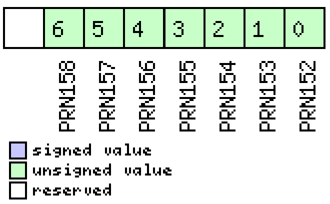
This Graphic explains the bits of `usage`



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

Bitfield scanmode2

This Graphic explains the bits of `scanmode2`



Bitfield scanmode1

This Graphic explains the bits of `scanmode1`

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRN151	PRN150	PRN149	PRN148	PRN147	PRN146	PRN145	PRN144	PRN143	PRN142	PRN141	PRN140	PRN139	PRN138	PRN137	PRN136	PRN135	PRN134	PRN133	PRN132	PRN131	PRN130	PRN129	PRN128	PRN127	PRN126	PRN125	PRN124	PRN123	PRN122	PRN121	PRN120

signed value
unsigned value
reserved

CFG-NMEA (0x06 0x17)

Poll the NMEA protocol configuration

Message	**CFG-NMEA**				
Description	**Poll the NMEA protocol configuration**				
Type	Poll Request				
Comment	-				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x06 0x17	0	*see below*	CK_A CK_B
No payload					

Set/Get the NMEA protocol configuration

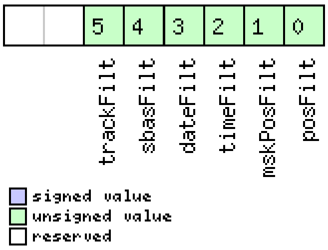
Message	**CFG-NMEA**				
Description	**Set/Get the NMEA protocol configuration**				
Type	Set/Get				
Comment	Set/Get the NMEA protocol configuration. See section NMEA Protocol Configuration for a detailed description of the configuration effects on NMEA output.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	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	-	`version`	-	0x23 = NMEA version 2.3 0x21 = NMEA version 2.1
2	U1	-	`numSV`	-	Maximum Number of SVs to report in NMEA protocol. This does not affect the receiver's operation. It only limits the number of SVs reported in NMEA mode (this might be needed with older mapping applications which only support 8- or 12-channel receivers).
3	X1	-	`flags`	-	flags (see graphic below)

Bitfield filter

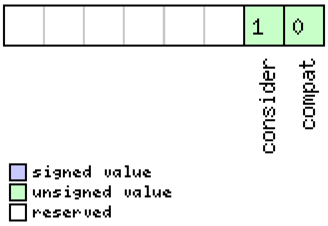
This Graphic explains the bits of `filter`



Name	*Description*
`posFilt`	disable position filtering
`mskPosFilt`	disable masked position filtering
`timeFilt`	disable time filtering
`dateFilt`	disable date filtering
`sbasFilt`	enable SBAS filtering
`trackFilt`	disable track filtering

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.

CFG-USB (0x06 0x1B)

Poll a USB configuration

Message	**CFG-USB**				
Description	**Poll a USB configuration**				
Type	Poll Request				
Comment	-				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x06 0x1B	0	*see below*	CK_A CK_B
No payload					

Get/Set USB Configuration

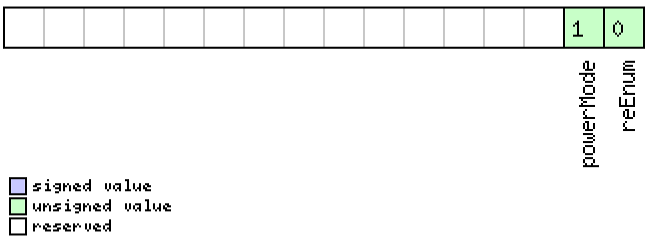
Message	**CFG-USB**				
Description	**Get/Set USB Configuration**				
Type	Get/Set				
Comment	-				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x06 0x1B	108	*see below*	CK_A CK_B

Payload Contents:

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

Bitfield flags

This Graphic explains the bits of `flags`



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

CFG-TMODE (0x06 0x1D)

Poll Time Mode Settings

Message	**CFG-TMODE**				
Description	**Poll Time Mode Settings**				
Type	Poll Request				
Comment	**This message is available only for timing receivers** Sending this (empty / no-payload) message to the receiver results in the receiver returning a message of type CFG-TMODE with a payload as defined below				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x06 0x1D	0	*see below*	CK_A CK_B
No payload					

Time Mode Settings

Message	**CFG-TMODE**				
Description	**Time Mode Settings**				
Type	Get/Set				
Comment	**This message is available only for timing receivers** See the Time Mode Description for details.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x06 0x1D	28	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`timeMode`	-	Time Transfer Mode: 0 Disabled 1 Survey In 2 Fixed Mode (true position information required) 3-255 Reserved
4	I4	-	`fixedPosX`	cm	Fixed Position ECEF X coordinate
8	I4	-	`fixedPosY`	cm	Fixed Position ECEF Y coordinate
12	I4	-	`fixedPosZ`	cm	Fixed Position ECEF Z coordinate
16	U4	-	`fixedPosVar`	mm^2	Fixed position 3D variance
20	U4	-	`svinMinDur`	s	Survey-in minimum duration
24	U4	-	`svinVarLimit`	mm^2	Survey-in position variance limit

CFG-NAVX5 (0x06 0x23)

Poll Navigation Engine Expert Settings

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

Get/Set Navigation Engine Expert Settings

Message	**CFG-NAVX5**				
Description	**Get/Set Navigation Engine Expert Settings**				
Type	Get/Set				
Comment	-				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x06 0x23	40	*see below*	CK_A CK_B

Payload Contents:

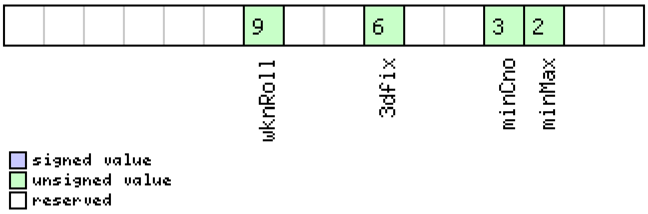
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U2	-	`version`	-	Message version. Current version is 0.
2	X2	-	`mask1`	-	First Parameters Bitmask. Only the flagged parameters will be applied, unused bits must be set to 0. (see graphic below)
4	X4	-	`mask2`	-	Second Parameters Bitmask. Currently unused, must be set to 0.
8	U1	-	`res1`	-	reserved, set to 0
9	U1	-	`res2`	-	reserved, set to 0
10	U1	-	`minSVs`	#SVs	Minimum number of satellites for navigation
11	U1	-	`maxSVs`	#SVs	Maximum number of satellites for navigation
12	U1	-	`minCNO`	dbHz	Minimum satellite signal level for navigation
13	U1	-	`res3`	-	reserved, set to 0
14	U1	-	`iniFix3D`	-	Initial Fix must be 3D flag (0=false/1=true)
15	U1	-	`res4`	-	reserved, set to 0
16	U1	-	`res5`	-	reserved, set to 0
17	U1	-	`res6`	-	reserved, set to 0
18	U2	-	`wknRollover`	-	GPS week rollover number; GPS week numbers will be set correctly from this week up to 1024 weeks after this week. Setting this to 0 reverts to firmware default.
20	U4	-	`res7`	-	reserved, set to 0

CFG-NAVX5 continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
24	U4	-	`res8`	-	reserved, set to 0
28	U4	-	`res9`	-	reserved, set to 0
32	U4	-	`res10`	-	reserved, set to 0
36	U4	-	`res11`	-	reserved, set to 0

Bitfield mask1

This Graphic explains the bits of `mask1`



Name	*Description*
`minMax`	Apply min/max SVs settings
`minCno`	Apply minimum C/N0 setting
`3dfix`	Apply initial 3D fix settings
`wknRoll`	Apply GPS weeknumber rollover settings

CFG-NAV5 (0x06 0x24)

Poll Navigation Engine Settings

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

Get/Set Navigation Engine Settings

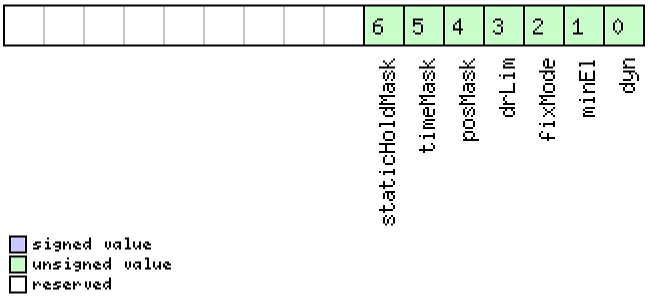
Message	CFG-NAV5				
Description	**Get/Set Navigation Engine Settings**				
Type	Get/Set				
Comment	See the Navigation Configuration Settings Description for a detailed description of how these settings affect receiver operation.				
	Header	ID	Length (Bytes)	Payload	Checksum
Message Structure	0xB5 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
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	Maximum time to perform dead reckoning (linear extrapolation) in case of GPS signal loss
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	-	`res1`	-	reserved, set to 0
24	U4	-	`res2`	-	reserved, set to 0
28	U4	-	`res3`	-	reserved, set to 0
32	U4	-	`res4`	-	reserved, set to 0

Bitfield mask

This Graphic explains the bits of `mask`



Name	Description
dyn	Apply dynamic model settings
minEl	Apply minimum elevation settings
fixMode	Apply fix mode settings
drLim	Apply DR limit settings
posMask	Apply position mask settings
timeMask	Apply time mask settings
staticHoldMask	Apply static hold settings

MON (0x0A)

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

MON-IO (0x0A 0x02)

I/O Subsystem Status

Message	**MON-IO**				
Description	**I/O Subsystem Status**				
Type	Periodic/Polled				
Comment	The size of the message is determined by the NPRT number of ports the receiver supports, i. e. on ANTARIS this is always 4, on u-blox 5 the number of ports is 6.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0A 0x02	0 + 20*NPRT	*see below*	CK_A CK_B

Payload Contents:

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

MON-VER (0x0A 0x04)

Receiver/Software Version

Message	**MON-VER**					
Description	**Receiver/Software Version**					
Type	Answer to Poll					
Comment	-					
	Header	*ID*	*Length (Bytes)*		*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0A 0x04	40 + 30*Num		*see below*	CK_A CK_B

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
Payload Contents:					
0	CH[30]	-	`swVersion`	-	Zero-terminated Software Version String
30	CH[10]	-	`hwVersion`	-	Zero-terminated Hardware Version String
Start of repeated block (Num times)					
40 + 30*N	CH[30]	-	`extension`	-	Installed Extension Package Version
End of repeated block					

MON-MSGPP (0x0A 0x06)

Message Parse and Process Status

Message	**MON-MSGPP**					
Description	**Message Parse and Process Status**					
Type	Periodic/Polled					
Comment	-					
	Header	*ID*	*Length (Bytes)*		*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0A 0x06	120		*see below*	CK_A CK_B

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
Payload Contents:					
0	U2[8]	-	`msg1`	msgs	Number of successfully parsed messages for each protocol on target0
16	U2[8]	-	`msg2`	msgs	Number of successfully parsed messages for each protocol on target1
32	U2[8]	-	`msg3`	msgs	Number of successfully parsed messages for each protocol on target2
48	U2[8]	-	`msg4`	msgs	Number of successfully parsed messages for each protocol on target3
64	U2[8]	-	`msg5`	msgs	Number of successfully parsed messages for each protocol on target4
80	U2[8]	-	`msg6`	msgs	Number of successfully parsed messages for each protocol on target5
96	U4[6]	-	`skipped`	bytes	Number skipped bytes for each target

MON-RXBUF (0x0A 0x07)

Receiver Buffer Status

Message	**MON-RXBUF**				
Description	**Receiver Buffer Status**				
Type	Periodic/Polled				
Comment	-				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0A 0x07	24	*see below*	CK_A CK_B

Payload Contents:

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

MON-TXBUF (0x0A 0x08)

Transmitter Buffer Status

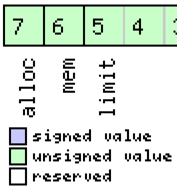
Message	**MON-TXBUF**				
Description	**Transmitter Buffer Status**				
Type	Periodic/Polled				
Comment	-				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0A 0x08	28	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U2[6]	-	`pending`	bytes	Number of bytes pending in transmitter buffer for each target
12	U1[6]	-	`usage`	%	Maximum usage transmitter buffer during the last sysmon period for each target
18	U1[6]	-	`peakUsage`	%	Maximum usage transmitter buffer for each target
24	U1	-	`tUsage`	%	Maximum usage of transmitter buffer during the last sysmon period for all targets
25	U1	-	`tPeakusage`	%	Maximum usage of transmitter buffer for all targets
26	X1	-	`errors`	-	Error bitmask (see graphic below)
27	U1	-	`res`	-	reserved

Bitfield errors

This Graphic explains the bits of `errors`



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

MON-HW (0x0A 0x09)

Hardware Status

Message	**MON-HW**				
Description	**Hardware Status**				
Type	Periodic/Polled				
Comment	Status of different aspect of the hardware, such as Antenna, PIO/Peripheral Pins, Noise Level, Automatic Gain Control (AGC)				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0A 0x09	68	*see below*	CK_A CK_B

Payload Contents:

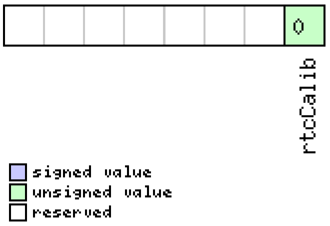
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	X4	-	`pinSel`	-	Mask of Pins Set as Peripheral/PIO
4	X4	-	`pinBank`	-	Mask of Pins Set as Bank A/B
8	X4	-	`pinDir`	-	Mask of Pins Set as Input/Output
12	X4	-	`pinVal`	-	Mask of Pins Value Low/High
16	U2	-	`noisePerMS`	-	Noise Level as measured by the GPS Core
18	U2	-	`agcCnt`	-	AGC Monitor (counts SIGHI xor SIGLO, range 0 to 8191)
20	U1	-	`aStatus`	-	Status of the Antenna Supervisor State Machine (0=INIT, 1=DONTKNOW, 2=OK, 3=SHORT, 4=OPEN)
21	U1	-	`aPower`	-	Current PowerStatus of Antenna (0=OFF, 1=ON, 2=DONTKNOW)
22	X1	-	`flags`	-	Flags (see graphic below)
23	U1	-	`res1`	-	Reserved
24	X4	-	`usedMask`	-	Mask of Pins that are used by the Virtual Pin Manager
28	U1[25]	-	`VP`	-	Array of Pin Mappings for each of the 25 Physical Pins
53	U1[3]	-	`res2`	-	Reserved
56	X4	-	`pinIrq`	-	Mask of Pins Value using the PIO Irq

MON-HW continued

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
60	X4	-	`pullH`	-	Mask of Pins Value using the PIO Pull High Resistor
64	X4	-	`pullL`	-	Mask of Pins Value using the PIO Pull Low Resistor

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`rtcCalib`	RTC is calibrated

AID (0x0B)

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

AID-REQ (0x0B 0x00)

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

Message	**AID-REQ**				
Description	**Sends a poll (AID-DATA) for all GPS Aiding Data**				
Type	Virtual				
Comment	**AID-REQ is not a message but a placeholder for configuration purposes.** If the virtual AID-REQ is configured to be output (see CFG-MSG), the receiver will output a request for aiding data (AID-DATA) after a start-up if its internally stored data (position, time, ephemeris, almanac) don't allow it to perform a hot start.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0B 0x00	0	*see below*	CK_A CK_B
No payload					

AID-INI (0x0B 0x01)

Poll GPS Initial Aiding Data

Message	**AID-INI**				
Description	**Poll GPS Initial Aiding Data**				
Type	Poll Request				
Comment	**This message has an empty payload!** -				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0B 0x01	0	*see below*	CK_A CK_B
No payload					

Aiding position, time, frequency, clock drift

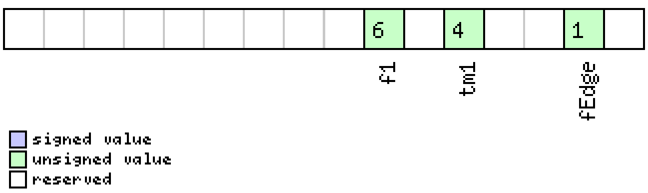
Message	**AID-INI**				
Description	**Aiding position, time, frequency, clock drift**				
Type	Polled				
Comment	This message contains position, time and clock drift information. The position can be input in either the ECEF X/Y/Z coordinate system or as lat/lon/height. The time can either be input as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

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	I4	-	`ecefXOrLat`	cm_or_deg*1e7	WGS84 ECEF X coordinate or latitude, depending on flags below
4	I4	-	`ecefYOrLon`	cm_or_deg*1e7	WGS84 ECEF Y coordinate or longitude, depending on flags below
8	I4	-	`ecefZOrAlt`	cm	WGS84 ECEF Z coordinate or altitude, depending on flags below
12	U4	-	`posAcc`	cm	Position accuracy (stddev)
16	X2	-	`tmCfg`	-	Time mark configuration (see graphic below)
18	U2	-	`wn`	-	Actual week number
20	U4	-	`tow`	ms	Actual time of week
24	I4	-	`towNs`	ns	Sub-millisecond part of time of week
28	U4	-	`tAccMs`	ms	Milliseconds part of time accuracy
32	U4	-	`tAccNs`	ns	Nanoseconds part of time accuracy
36	I4	-	`clkDOrFreq`	ns/s_or_Hz	Clock drift or frequency, depending on flags below
40	U4	-	`clkDAccOrFreqAcc`	ns/s_or_ppm	Accuracy of clock drift or frequency, depending on flags below
44	X4	-	`flags`	-	Bitmask with the following flags (see graphic below)

Bitfield tmCfg

This Graphic explains the bits of `tmCfg`



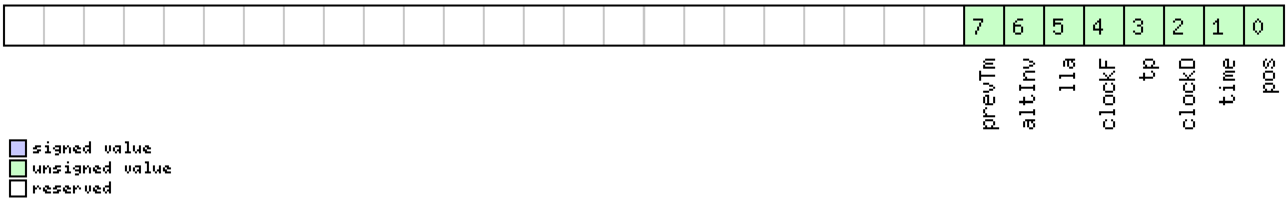
Name	*Description*

Bitfield tmCfg Description continued

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

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`pos`	Position is valid
`time`	Time is valid
`clockD`	Clock drift data contains valid clock drift, must not be set together with clockF
`tp`	Use time pulse
`clockF`	Clock drift data contains valid frequency, must not be set together with clockD
`lla`	Position is given in LAT/LON/ALT (default is ECEF)
`altInv`	Altitude is not valid, in case lla was set
`prevTm`	Use time mark received before AID-INI message (default uses mark received after message)

AID-HUI (0x0B 0x02)

Poll GPS Health, UTC and ionosphere parameters

Message	**AID-HUI**				
Description	**Poll GPS Health, UTC and ionosphere parameters**				
Type	Poll Request				
Comment	**This message has an empty payload!** -				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0B 0x02	0	*see below*	CK_A CK_B
No payload					

GPS Health, UTC and ionosphere parameters

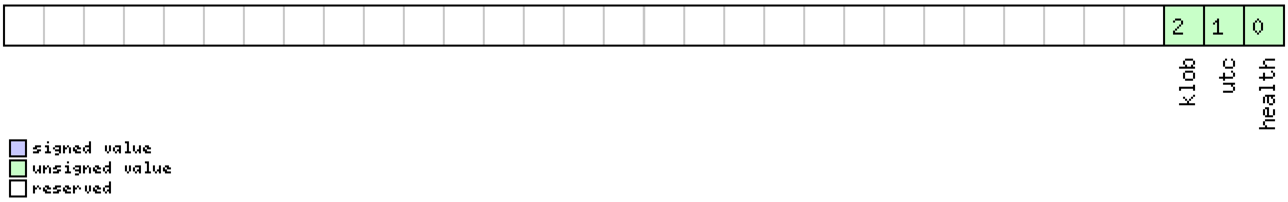
Message	**AID-HUI**				
Description	**GPS Health, UTC and ionosphere parameters**				
Type	Input/Output Message				
Comment	This message contains a health bit mask, UTC time and Klobuchar parameters. For more information on these parameters, please see the ICD-GPS-200 documentation.				
	Header	*ID*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

Payload Contents:

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

Bitfield flags

This Graphic explains the bits of `flags`



Name	Description
health	Healthmask field in this message is valid
utc	UTC parameter fields in this message are valid
klob	Klobuchar parameter fields in this message are valid

AID-DATA (0x0B 0x10)

Polls all GPS Initial Aiding Data

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

AID-ALM (0x0B 0x30)

Poll GPS Aiding Almanach Data

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

Poll GPS Aiding Almanach Data for a SV

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

Payload Contents:

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

GPS Aiding Almanach Input/Output Message

Message	**AID-ALM**				
Description	**GPS Aiding Almanach Input/Output Message**				
Type	Input/Output Message				
Comment	• If the WEEK Value is 0, DWRD0 to DWRD7 are not sent as the almanach is not available for the given SV. • DWORD0 to DWORD7 contain the 8 words following the Hand-Over Word (HOW) from the GPS navigation message, either pages 1 to 24 of sub-frame 5 or pages 2 to 10 of subframe 4. See IS-GPS-200 for a full description of the contents of the Almanac pages. • In DWORD0 to DWORD7, the parity bits have been removed, and the 24 bits of data are located in Bits 0 to 23. Bits 24 to 31 shall be ignored. • Example: Parameter e (Eccentricity) from Almanach Subframe 4/5, Word 3, Bits 69-84 within the subframe can be found in DWRD0, Bits 15-0 whereas Bit 0 is the LSB.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0B 0x30	(8) or (40)	*see below*	CK_A CK_B

Payload Contents:

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

AID-EPH (0x0B 0x31)

Poll GPS Aiding Ephemeris Data

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

Poll GPS Aiding Ephemeris Data for a SV

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

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

GPS Aiding Ephemeris Input/Output Message

Message	**AID-EPH**				
Description	**GPS Aiding Ephemeris Input/Output Message**				
Type	Input/Output Message				
Comment	• SF1D0 to SF3D7 is only sent if ephemeris is available for this SV. If not, the payload may be reduced to 8 Bytes, or all bytes are set to zero, indicating that this SV Number does not have valid ephemeris for the moment. • SF1D0 to SF3D7 contain the 24 words following the Hand-Over Word (HOW) from the GPS navigation message, subframes 1 to 3. See IS-GPS-200 for a full description of the contents of the Subframes. • In SF1D0 to SF3D7, the parity bits have been removed, and the 24 bits of data are located in Bits 0 to 23. Bits 24 to 31 shall be ignored.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0B 0x31	(8) or (104)	*see below*	CK_A CK_B

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

AID-ALPSRV (0x0B 0x32)

ALP client requests AlmanacPlus data from server

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

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

ALP server sends AlmanacPlus data to client

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

Payload Contents:

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

ALP client sends AlmanacPlus data to server.

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

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	`idSize`	bytes	Identifier size
1	U1	-	`type`	-	Set to 0xff to mark that is *not* a data request
2	U2	-	`ofs`	-	Data offset [16bit words]
4	U2	-	`size`	-	Data size [16bit words]

AID-ALPSRV continued

Byte Offset	Number Format	Scaling	Name	Unit	Description
6	U2	-	fileId	-	Corresponding ALP file id
Start of repeated block (size times)					
8 + 2*N	U2	-	data	-	16bit word data to be submitted to the ALP server
End of repeated block					

AID-ALP (0x0B 0x50)

ALP file data transfer to the receiver

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

Payload Contents:

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

Mark end of data transfer

Message	**AID-ALP**					
Description	**Mark end of data transfer**					
Type	Input message					
Comment	This message is used to indicate that all chunks have been transferred, and normal receiver operation can resume. Upon reception of this message, the receiver will verify all chunks received so far, and enable AssistNow Offline and GPS receiver operation if successful. This message could also be sent to cancel an incomplete download.					
	Header	*ID*	*Length (Bytes)*		*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0B 0x50	1		*see below*	CK_A CK_B
Payload Contents:						
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*	
0	U1	-	dummy	-	Value is ignored	

Acknowledges a data transfer

Message	**AID-ALP**					
Description	**Acknowledges a data transfer**					
Type	Output message					
Comment	This message from the receiver acknowledges successful processing of a previously received chunk of data with the "Chunk Transfer" Message. This message will also be sent once a "Stop" message has been received, and the integrity of all chunks received so far has been checked successfully.					
	Header	*ID*	*Length (Bytes)*		*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0B 0x50	1		*see below*	CK_A CK_B
Payload Contents:						
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*	
0	U1	-	ack	-	Set to 0x01	

Indicate problems with a data transfer

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

Payload Contents:

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

Poll the AlmanacPlus status

Message	**AID-ALP**				
Description	**Poll the AlmanacPlus status**				
Type	Periodic/Polled				
Comment	-				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0B 0x50	24	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`predTow`	s	Prediction start time of week
4	U4	-	`predDur`	s	Prediction duration from start of first data set to end of last data set
8	I4	-	`age`	s	Current age of ALP data
12	U2	-	`predWno`	-	Prediction start week number
14	U2	-	`almWno`	-	Truncated week number of reference almanac
16	U4	-	`res1`	-	Reserved for future use
20	U1	-	`svs`	-	Number of satellite data sets contained in the ALP data
21	U1	-	`res2`	-	Reserved for future use
22	U1	-	`res3`	-	Reserved for future use
23	U1	-	`res4`	-	Reserved for future use

TIM (0x0D)

Timing Messages: i.e. Timepulse Output, Timemark Results.
Messages in this class are output by the receiver, giving information on Timepulse and Timemark measurements.

TIM-TP (0x0D 0x01)

Timepulse Timedata

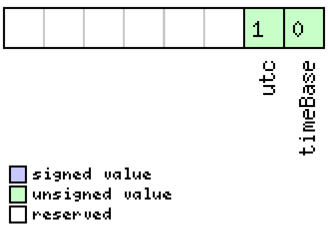
Message	**TIM-TP**				
Description	**Timepulse Timedata**				
Type	Periodic/Polled				
Comment	This message contains information for high precision timing. Note that contents are correct only if the timepulse is set to one pulse per second.				
Message Structure	*Header*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
	0xB5 0x62	0x0D 0x01	16	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`towMS`	ms	Timepulse time of week according to time base
4	U4	2^-32	`towSubMS`	ms	Submillisecond part of TOWMS
8	I4	-	`qErr`	ps	Quantization error of timepulse.
12	U2	-	`week`	weeks	Timepulse week number according to time base
14	X1	-	`flags`	-	bitmask (see graphic below)
15	U1	-	`res`	-	unused

Bitfield flags

This Graphic explains the bits of `flags`



Name	*Description*
`timeBase`	0=Time base is GPS 1=Time base is UTC
`utc`	0=UTC not available 1=UTC available

TIM-TM2 (0x0D 0x03)

Time mark data

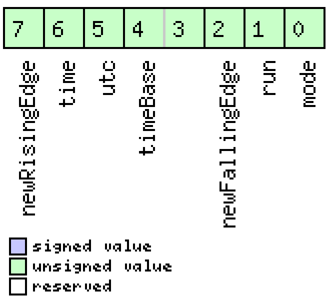
Message	TIM-TM2				
Description	**Time mark data**				
Type	Periodic/Polled				
Comment	This message contains information for high precision time stamping / pulse counting. The delay figures given in CFG-TP are also applied to the time results output in this message.				
	Header	ID	Length (Bytes)	Payload	Checksum
Message Structure	0xB5 0x62	0x0D 0x03	28	see below	CK_A CK_B

Payload Contents:

Byte Offset	Number Format	Scaling	Name	Unit	Description
0	U1	-	`ch`	time	marker channel 0 or 1
1	X1	-	`flags`	-	Bitmask (see graphic below)
2	U2	-	`count`	-	edge counter.
4	U2	-	`wnR`	-	week number of last rising edge
6	U2	-	`wnF`	-	week number of last falling edge
8	U4	-	`towMsR`	ms	tow of rising edge
12	U4	-	`towSubMsR`	ns	millisecond fraction of tow of rising edge in nanoseconds
16	U4	-	`towMsF`	ms	tow of falling edge
20	U4	-	`towSubMsF`	ns	millisecond fraction of tow of falling edge in nanoseconds
24	U4	-	`accEst`	ns	Accuracy estimate

Bitfield flags

This Graphic explains the bits of `flags`



Name	Description
`mode`	0=single 1=running
`run`	0=armed 1=stopped
`newFallingEdge`	new falling edge detected

Bitfield flags Description continued

Name	*Description*
`timeBase`	0=Time base is Receiver Time 1=Time base is GPS 2=Time base is UTC
`utc`	0=UTC not available 1=UTC available
`time`	0=Time is not valid 1=Time is valid (Valid GPS fix)
`newRisingEdge`	new rising edge detected

TIM-SVIN (0x0D 0x04)

Survey-in data

Message	**TIM-SVIN**				
Description	**Survey-in data**				
Type	Periodic/Polled				
Comment	**This message is only supported on timing receivers** This message contains information about survey-in parameters. For details about the Time Mode see section Time Mode Configuration.				
	Header	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0D 0x04	28	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U4	-	`dur`	s	Passed survey-in observation time
4	I4	-	`meanX`	cm	Current survey-in mean position ECEF X coordinate
8	I4	-	`meanY`	cm	Current survey-in mean position ECEF Y coordinate
12	I4	-	`meanZ`	cm	Current survey-in mean position ECEF Z coordinate
16	U4	-	`meanV`	mm^2	Current survey-in mean position 3D variance
20	U4	-	`obs`	-	Observations used during survey-in
24	U1	-	`valid`	-	Survey-in position validity flag
25	U1	-	`active`	-	Survey-in in progress flag
26	U2	-	`reserved`	-	Reserved

Appendix

u-blox 5 Default Settings

The default settings listed in this section apply to u-blox 5 ROM-based receivers with ROM version 4.0. These values assume that the default levels of the configuration pins have been left unchanged. Default settings are dependent on the configuration pin settings, for information regarding these settings, consult the applicable Data Sheet.

Antenna Supervisor Settings (UBX-CFG-ANT)

For parameter and protocol description see section UBX-CFG-ANT.

Antenna Settings

Parameter	*Default Setting*	*Unit*
Enable Control Signal	Enabled	
Enable Short Circuit Detection	Enabled	
Enable Short Circuit Power Down logic	Enabled	
Enable Automatic Short Circuit Recovery logic	Enabled	
Enable Open Circuit Detection	Disabled	

Datum Settings (UBX-CFG-DAT)

For parameter and protocol description see section UBX-CFG-DAT.

Datum Default Settings

Parameter	*Default Setting*	*Unit*
Datum	0 – WGS84	

Navigation Settings (UBX-CFG-NAV5)

For parameter and protocol description see section UBX-CFG-NAV5.

Navigation Default Settings

Parameter	*Default Setting*	*Unit*
Dynamic Platform Model	0 – Portable	
Fix Mode	Auto 2D/3D	#
Fixed Altitude	N/A	m
Fixed Altitude Variance	N/A	m^2
Min SV Elevation	5	deg
DR Timeout	0	s
PDOP Mask	25	-
TDOP Mask	25	-
P Accuracy	100	m
T Accuracy	300	m
Static Hold Threshold	0.00	m/s

Output Rates (UBX-CFG-RATE)

For parameter and protocol description see section UBX-CFG-RATE.

Output Rate Default Settings

Parameter	Default Setting	Unit
Time Source	1 – GPS time	
Measurement Period	1000	ms
Measurement Rate	1	Cycles

SBAS Configuration (UBX-CFG-SBAS)

For parameter and protocol description see section UBX-CFG-SBAS.

SBAS Configuration Default Settings

Parameter	Default Setting	Unit
SBAS Subsystem	Enabled	
Allow test mode usage	Disabled	
Ranging (Use SBAS for navigation)	Enabled	
Apply SBAS Correction Data	Enabled	
Apply integrity information	Disabled	
Number of search channels	3	
PRN Codes	120, 122, 124, 126-127, 129, 131, 134-135, 137-138	

Port Setting (UBX-CFG-PRT)

For parameter and protocol description see section UBX-CFG-PRT.

Port Default Settings

Parameter	Default Setting	Unit
DDC/I2C (Target0)		
Protocol in	0+1 – UBX+NMEA	
Protocol out	0+1 – UBX+NMEA	
USART1 (Target1)		
Protocol in	0+1 – UBX+NMEA	
Protocol out	0+1 – UBX+NMEA	
Baudrate	9600	baud
USART2 (Target2)		
Protocol in	None	
Protocol out	None	
Baudrate	9600	baud
USB (Target3)		
Protocol in	0+1 – UBX+NMEA	
Protocol out	0+1 – UBX+NMEA	
SPI (Target4)		
Protocol in	0+1 – UBX+NMEA	
Protocol out	0+1 – UBX+NMEA	

Port Setting (UBX-CFG-USB)

For parameter and protocol description see section UBX-CFG-USB.

USB default settings

Parameter	*Default Setting*	*Unit*
Power Mode		
Power Mode	Bus powered	
Bus Current required	120	mA

Message Settings (UBX-CFG-MSG)

For parameter and protocol description see section UBX-CFG-MSG.

Enabled output messages

Message	*Type*	*All Targets*
NMEA - GGA	Out	1
NMEA - GLL	Out	1
NMEA - GSA	Out	1
NMEA - GSV	Out	1
NMEA - RMC	Out	1
NMEA - VTG	Out	1

NMEA Protocol Settings (UBX-CFG-NMEA)

For parameter and protocol description see section UBX-CFG-NMEA.

NMEA Protocol Default Settings

Parameter	*Default Setting*	*Unit*
Enable position output even for invalid fixes	Disabled	
Enable position even for masked fixes	Disabled	
Enable time output even for invalid times	Disabled	
Enable time output even for invalid dates	Disabled	
Version	2.3	
Compatibility Mode	Disabled	
Consideration Mode	Enabled	
Number of SV	Unlimited	

INF Messages Settings (UBX–CFG–INF)

For parameter and protocol description see section UBX-CFG-INF.

NMEA default enabled INF msg

Message	*Type*	*All Targets*	*Range/Remark*
INF-Error	Out	1	In NMEA Protocol only (GPTXT)
INF-Warning	Out	1	In NMEA Protocol only (GPTXT)
INF-Notice	Out	1	In NMEA Protocol only (GPTXT)
INF-Test	Out		

NMEA default enabled INF msg continued

Message	*Type*	*All Targets*	*Range/Remark*
INF-Debug	Out		
INF-User	Out	1	In NMEA Protocol only (GPTXT)

Timepulse Settings (UBX–CFG–TP)

For parameter and protocol description see section UBX-CFG-TP.

Timepulse default settings

Parameter	*Default Setting*	*Unit*
Pulse Mode	+1 – rising	
Pulse Period	1000	ms
Pulse Length	100	ms
Time Source	1 – GPS time	
Cable Delay	50	ns
User Delay	0	ns
SyncMode	0 (no time pulse in case of no fix)	