# u-blox D9 PMP 1.04

## u-blox D9 correction data receiver

Interface description


**Abstract**
This document describes the interface (version 24.00) of the NEO-D9S, the first mass-market L-band GNSS correction module.

**www.u-blox.com**

UBX-21040023 - R01
C1-Public

# Document information

| | | |
|---|---|---|
| **Title** | **u-blox D9 PMP 1.04** | |
| **Subtitle** | u-blox D9 correction data receiver | |
| **Document type** | Interface description | |
| **Document number** | UBX-21040023 | |
| **Revision and date** | R01 | 21-Oct-2021 |
| **Disclosure restriction** | C1-Public | |

# Contents

# 1 General information

## 1.1 Document overview

This document describes the interface of the u-blox D9 correction data receiver. The interface consists of the following parts:

- UBX protocol
- Configuration interface

See also Related documents.

Some of the features described here may not be available in your product, and some may require specific configurations to be enabled. See the data sheet of your specific product for availability and the integration manual for instructions for enabling the features.

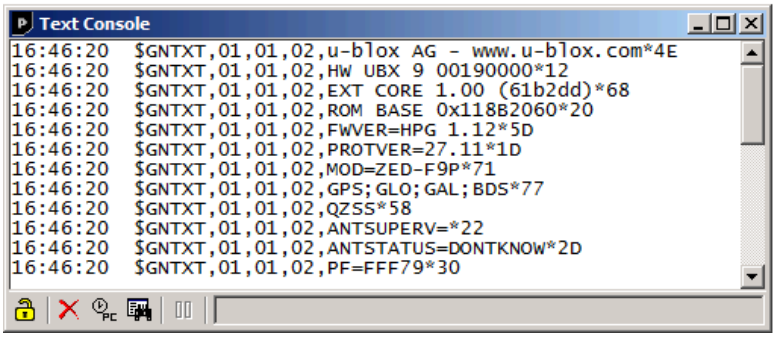
Previous versions of u-blox receiver documentation combined general receiver description and interface specification. In the current documentation the receiver description is included in the integration manual.

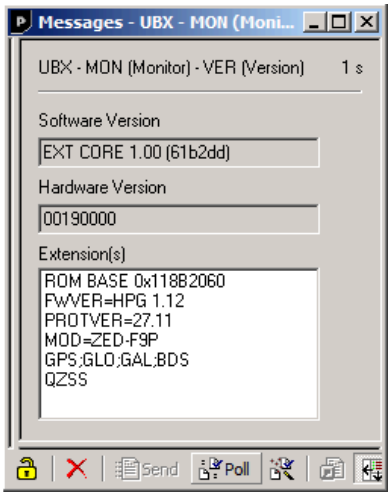
## 1.2 Firmware and protocol versions

u-blox generation 9 receivers execute firmware from internal ROM or from internal code-RAM. If the firmware image is stored in a flash it is loaded into the code-RAM before execution. It is also possible to store the firmware image in the host system. The firmware is then loaded into the code-RAM from the host processor. (Loading the firmware from the host processor is not supported in all products.) If there is no external firmware image, then the firmware is executed from the ROM.

The location and the version of the boot loader and the currently running firmware can be found in the boot screen and in the UBX-MON-VER message. If the firmware has been loaded from a connected flash or from the host processor, it is indicated by text "EXT". When the receiver is started, the boot screen is output automatically in UBX-INF-NOTICE or NMEA-Standard-TXT messages if configured using CFG-INFMSG. The UBX-MON-VER message can be polled using the UBX polling mechanism.

The following u-center screenshots show an example of a u-blox receiver running firmware loaded from flash:



The following information is available (✓) from the boot screen (**B**) and the UBX-MON-VER message (**M**):

| B | M | Example | Information |
|---|---|---|---|
| ✓ | | `u-blox AG - www.u-blox.com` | Start of the boot screen. |
| ✓ | | `HW UBX 9 00190000` | Hardware version of the u-blox receiver. |
| | ✓ | `00190000` | |
| ✓ | ✓ | `EXT CORE 1.00 (61b2dd)` | Base (CORE) firmware version and revision number, loaded from external memory (EXT). |
| | | `EXT LAP 1.00 (12a3bc)` | Product firmware version and revision number, loaded from external memory (EXT). Available only in some firmware versions. See below for a list of product acronyms. |
| ✓ | ✓ | `ROM BASE 0x118B2060` | Revision number of the underlying boot loader firmware in ROM. |
| ✓ | ✓ | `FWVER=HPG 1.12` | Product firmware version number, where:<br>• `SPG` = Standard precision GNSS product<br>• `HPG` = High precision GNSS product<br>• `ADR` = Automotive dead reckoning product<br>• `TIM` = Time sync product<br>• `LAP` = Lane accurate positioning product<br>• `HPS` = High precision sensor fusion product<br>• `DBS` = Dual band standard precision<br>• `MDR` = Multi-mode dead reckoning product<br>• `PMP` = L-Band Inmarsat point-to-multipoint receiver<br>• `QZS` = QZSS L6 centimeter level augmentation service (CLAS) message receiver |
| ✓ | ✓ | `PROTVER=34.00` | Supported protocol version. |
| ✓ | ✓ | `MOD=ZED-F9P` | Module name (if available). |
| ✓ | ✓ | `GPS;GLO;GAL;BDS` | List of supported major GNSS (see GNSS identifiers). |
| ✓ | ✓ | `SBAS;QZSS` | List of supported augmentation systems (see GNSS identifiers). |
| ✓ | | `ANTSUPERV=AC SD PDoS SR` | Configuration of the antenna supervisor (if available), where:<br>• `AC` = Active antenna control enabled<br>• `SD` = Short circuit detection enabled<br>• `OD` = Open circuit detection enabled<br>• `PDoS` = Short circuit power down logic enabled<br>• `SR` = Automatic recovery from short state enabled |
| ✓ | | `PF=FFF79` | Product configuration. |

☞ The "FWVER" product firmware version indicates which firmware is currently running. This is referred to as "firmware version" in this and other documents.

☞ The revision numbers should only be used to identify a known firmware version. They are not necessarily numeric nor are they guaranteed to increase with newer firmware versions.

☞ Similarly, firmware version numbers can have additional non-numeric information appended, such as in "5.00B03".

☞ Not every entry is output by all u-blox receivers. The availability of some of the information depends on the product, the firmware location and the firmware version.

The product firmware version and the base firmware version relate to the protocol version:

| Product firmware version | Base firmware version | Protocol version |
|---|---|---|
| PMP 1.04 | EXT CORE 1.00 (cd6322) | 24.00 |

## 1.3 Receiver configuration

u-blox positioning receivers are fully configurable with UBX protocol messages. The configuration used by the receiver during normal operation is called the "current configuration". The current configuration can be changed during normal operation by sending UBX-CFG-VALSET messages over any I/O port. The receiver will change its current configuration immediately after receiving a configuration message. The receiver will always use the current configuration only.

The current configuration is loaded from permanent configuration hard-coded in the receiver firmware (the defaults) and from non-volatile memory (user configuration) on startup of the receiver. Changes made to the current configuration at run-time will be lost when there is a power cycle, a hardware reset or a (complete) controlled software reset (see Configuration reset behavior).

See Configuration interface for a detailed description of the receiver configuration system, the explanation of the configuration concept and its principles and interfaces.

☞ The configuration interface has changed from earlier u-blox positioning receivers. There is some backwards compatibility provided in UBX-CFG configuration messages. Users are strongly advised to only use the Configuration interface. See also Legacy UBX message fields reference.

☞ See the integration manual for a basic receiver configuration most commonly used.

## 1.4 Naming

Message names are written in full with the parts of the name separated by hyphens ("-"). The full message name consists of the protocol name (e.g., *UBX)*, the class name (e.g. *NAV*) and the message name (e.g. *PVT*). For example the receiver software version information message is referred to as *UBX-MON-VER*. Similarly, the *NMEA-Standard-GGA* is the NMEA standard message (sentence) with the global positioning fix data.

References to fields of the message add the field name separated by a dot ("."), e.g. *UBX-MON-VER.swVersion*.

Some messages use a fourth level of naming, called the message version. One example is the *UBX-MGA-GPS* message for GPS assistance data, which exists in versions for ephemerides (*UBX-MGA-GPS-EPH*) and almanacs (*UBX-MGA-GPS-ALM*).

Names of configuration items are of the form *CFG-GROUP-ITEM*. For example, *CFG-NAVSPG-DYNMODEL* refers to the navigation dynamic platform model the receiver uses. Constants add a fourth level to the item name, such as *CFG-NAVSPG-DYNMODEL-AUTOMOT* for the automotive platform model. In the context of describing an item's value, only the last part of the constant name can be used (e.g. "set *CFG-NAVSPG-DYNMODEL* to *PORT* for portable applications").

## 1.5 GNSS, satellite and signal identifiers

### 1.5.1 Overview

The UBX protocol messages use two different numbering schemes. Some messages use a one-byte (type U1) field for the satellite identifier (normally named `svid`). This uses numbering similar to the "extended" NMEA scheme and is merely an extension of the scheme in use for previous generations of u-blox receivers.

With the ever increasing numbers of GNSS satellites, this scheme has been phased out in recent u-blox positioning receivers (as numbers greater than 255 would have become necessary). Consequently, newer messages use a more sophisticated, flexible and future-proof approach. This

involves having a separate `gnssId` field to identify which GNSS the satellite is part of and a simple `svId` (SV for space vehicle) field that indicates which number the satellite is in that system. In nearly all cases, this means that the *svId* is the natural number associated with the satellite. For example the GLONASS SV4 is identified as *gnssId 6, svId 4*, while the GPS SV4 is *gnssId 0, svId 4*.

Signal identifiers are used where different signals from a GNSS satellite need to be distinguished (e.g. in the UBX-NAV-SIG message). A separate `sigId` field is used. These identifiers are only valid when combined with a GNSS identifier (`gnssId` field).

Note that the following sections are a generic overview for different u-blox positioning receivers. A particular product may not support all of the described GNSS identifiers, satellite numbers, signal identifiers or combinations thereof.

### 1.5.2 GNSS identifiers

The following table lists each GNSS along with the GNSS identifiers (UBX protocol), the system IDs , and abbreviations used in this document:

| GNSS | Abbreviations | | UBX gnssId | NMEA system ID | | |
|---|---|---|---|---|---|---|
| | | | | 2.3 - 4.0 | 4.10 | 4.11 |
| GPS | GPS | G | 0 | 1 | 1 | 1 |
| SBAS | SBAS | S | 1 | 1 | 1 | 1 |
| Galileo | GAL | E | 2 | n/a | 3 | 3 |
| BeiDou | BDS | B | 3 | n/a | (4)[1] | 4 |
| IMES | IMES | I | 4 | n/a | n/a | n/a |
| QZSS | QZSS | Q | 5 | n/a | (1)[1] | 5 |
| GLONASS | GLO | R | 6 | 2 | 2 | 2 |
| NavIC | NavIC | N | 7 | n/a | n/a | 6 |

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

### 1.5.3 Satellite identifiers

A summary of all the satellite numbering schemes used in the UBX protocol is provided in the following table.

| | | UBX Protocol | | NMEA Protocol 2.3 - 4.0 | | NMEA Protocol 4.10 | | NMEA Protocol 4.11 | |
|---|---|---|---|---|---|---|---|---|---|
| GNSS | SV Range | gnssId:svId | single svid | (strict) | (extended) | (strict) | (extended) | (strict) | (extended) |
| GPS | G1-G32 | 0:1-32 | 1-32 | 1-32 | 1-32 | 1-32 | 1-32 | 1-32 | 1-32 |
| SBAS | S120-S158 | 1:120-158 | 120-158 | 33-64 | 33-64, 152-158 | 33-64 | 33-64, 152-158 | 33-64 | 33-64, 152-158 |
| Galileo | E1-E36 | 2:1-36 | 211-246 | - | 301-336 | 1-36 | 1-36 | 1-36 | 1-36 |
| BeiDou | B1-B5 | 3:1-5 | 159-163 | - | 401-405 | 1-5 | 1-5 | 1-5 | 1-5 |
| | B6-B37 | 3:6-37 | 33-64 | - | 406-437 | 6-37 | 6-37 | 6-37 | 6-37 |
| | B38-B63 | 3:38-63 | n/a | - | 438-463 | 38-63 | 38-63 | 38-63 | 38-63 |
| IMES | I1-I10 | 4:1-10 | 173-182 | n/a | 173-182 | n/a | 173-182 | n/a | 173-182 |
| QZSS | Q1-Q10 | 5:1-10 | 193-202 | n/a | 193-202 | n/a | 193-202 | 1-10 | 1-10 |
| GLONASS | R1-R32, R? | 6:1-32, 6:255 | 65-96, 255 | 65-96, null | 65-96, null | 65-96, null | 65-96, null | 65-96, null | 65-96, null |

[1] While not defined by NMEA 4.10, u-blox receivers in this mode will use system ID 4 for BeiDou and, if extended satellite numbering is enabled, system ID 1 for QZSS.

| | | UBX Protocol | | NMEA Protocol 2.3 - 4.0 | | NMEA Protocol 4.10 | | NMEA Protocol 4.11 | |
|---|---|---|---|---|---|---|---|---|---|
| GNSS | SV Range | gnssId:svId | single svid | (strict) | (extended) | (strict) | (extended) | (strict) | (extended) |
| NavIC | N1-N7 | 7:1-7 | 247-253 | n/a | n/a | n/a | n/a | n/a | n/a |

Note that GLONASS satellites can be tracked before they have been identified. In UBX messages such unknown satellites will be reported with svId 255. In NMEA messages they will be null (empty) fields. Product-related documentation and u-center will use R? to label unidentified GLONASS satellites.

### 1.5.4 Signal identifiers

A summary of all the signal identification schemes used in the UBX protocol is provided in the following table. (Only a subset of the signals is supported by each product.)

| | UBX Protocol | | NMEA Protocol 4.10[5] | | NMEA Protocol 4.11[5] | |
|---|---|---|---|---|---|---|
| Signal | gnssId | sigId | System ID | Signal ID | System ID | Signal ID |
| GPS L1C/A[2] | 0 | 0 | 1 | 1 | 1 | 1 |
| GPS L2 CL | 0 | 3 | 1 | 6 | 1 | 6 |
| GPS L2 CM | 0 | 4 | 1 | 5 | 1 | 5 |
| GPS L5 I | 0 | 6 | 1 | 7 | 1 | 7 |
| GPS L5 Q | 0 | 7 | 1 | 8 | 1 | 8 |
| SBAS L1C/A[2] | 1 | 0 | 1 | 1 | 1 | 1 |
| Galileo E1 C[2] | 2 | 0 | 3 | 7 | 3 | 7 |
| Galileo E1 B[2] | 2 | 1 | 3 | 7 | 3 | 7 |
| Galileo E5 aI | 2 | 3 | 3 | 1 | 3 | 1 |
| Galileo E5 aQ | 2 | 4 | 3 | 1 | 3 | 1 |
| Galileo E5 bI | 2 | 5 | 3 | 2 | 3 | 2 |
| Galileo E5 bQ | 2 | 6 | 3 | 2 | 3 | 2 |
| BeiDou B1I D1[2] | 3 | 0 | (4)[3] | (1)[4] | 4 | 1 |
| BeiDou B1I D2[2] | 3 | 1 | (4)[3] | (1)[4] | 4 | 1 |
| BeiDou B2I D1 | 3 | 2 | (4)[3] | (3)[4] | 4 | B |
| BeiDou B2I D2 | 3 | 3 | (4)[3] | (3)[4] | 4 | B |
| BeiDou B1C | 3 | 5 | (4)[3] | N/A | 4 | 3 |
| BeiDou B2a | 3 | 7 | (4)[3] | N/A | 4 | 5 |
| QZSS L1C/A[2] | 5 | 0 | (1)[3] | (1)[4] | 5 | 1 |
| QZSS L1S | 5 | 1 | (1)[3] | (4)[4] | 5 | 4 |
| QZSS L2 CM | 5 | 4 | (1)[3] | (5)[4] | 5 | 5 |
| QZSS L2 CL | 5 | 5 | (1)[3] | (6)[4] | 5 | 6 |
| QZSS L5 I | 5 | 8 | (1)[3] | N/A | 5 | 7 |

[2] UBX messages that do not have an explicit `sigId` field contain information about the subset of signals marked.

[3] While not defined by NMEA 4.10, u-blox receivers in this mode will use system ID 4 for BeiDou and, if extended satellite numbering is enabled, system ID 1 for QZSS.

[4] BeiDou and QZSS signal ID are not defined in the NMEA protocol version 4.10. Values shown in the table are only valid for u-blox products and, for QZSS signal ID, if extended satellite numbering is enabled.

[5] NMEA System ID and Signal ID are in hexadecimal format.

| Signal | UBX Protocol | | NMEA Protocol 4.10[5] | | NMEA Protocol 4.11[5] | |
|---|---|---|---|---|---|---|
| | gnssId | sigId | System ID | Signal ID | System ID | Signal ID |
| QZSS L5 Q | 5 | 9 | (1)[3] | N/A | 5 | 8 |
| GLONASS L1 OF[2] | 6 | 0 | 2 | 1 | 2 | 1 |
| GLONASS L2 OF | 6 | 2 | 2 | 3 | 2 | 3 |
| NavIC L5 A | 7 | 0 | N/A | N/A | 6 | 1 |

## 1.6 Message types

The following message types are defined:

| Message type | Description |
|---|---|
| Input | Messages that are input to the receiver and never output. E.g. UBX-MGA-GPS-EPH. |
| Output | Messages that are output by the receiver in no particular interval and never input. E.g. UBX-ACK-ACK. |
| Input/output | Messages that can be output by or input to the receiver. E.g. UBX-MGA-DBD-DATA0. |
| Periodic | Messages that are output in regular intervals but cannot be polled. E.g. UBX-NAV-EOE. |
| Periodic/polled | Messages that are output in regular intervals and can be polled. E.g. UBX-NAV-PVT. |
| Command | Messages that are a command to the receiver. Similar to type *Input* these are input-only. E.g. UBX-CFG-RST. |
| Get | Output-only configuration or command messages. E.g. UBX-CFG-DAT. |
| Set | Input-only configuration or command messages. E.g. UBX-CFG-VALDEL. |
| Get/set | Input/output configuration or command messages. E.g. UBX-CFG-NAVX5. |
| Polled | Non-periodic messages that can only be polled. E.g. UBX-MON-VER. |
| Poll request | Poll request. E.g. UBX-MGA-DBD-POLL. |

# 2 UBX protocol

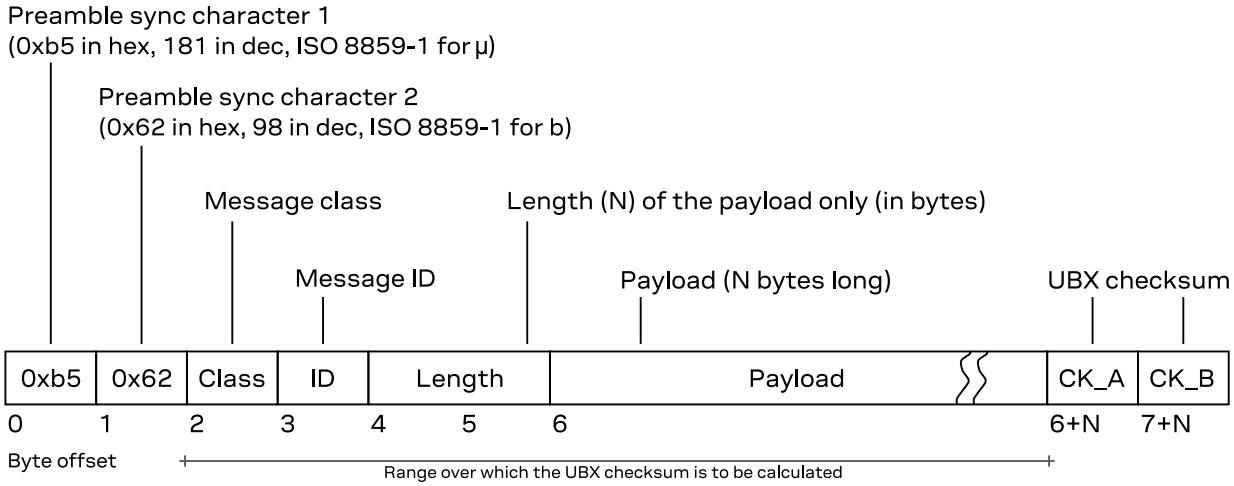
## 2.1 UBX protocol key features

u-blox receivers support a u-blox-proprietary protocol to communicate with a host computer. This protocol has the following key features:

- Compact – uses 8-bit binary data
- Checksum protected – uses a low-overhead checksum algorithm
- Modular – uses a two-stage message identifier (Class and Message ID)

## 2.2 UBX frame structure

The structure of a basic UBX frame is shown in the following diagram.




- Every *frame* starts with a 2-byte *preamble* consisting of two synchronization characters: `0xb5` and `0x62`.
- A 1-byte *message class* field follows. A class is a group of messages that are related to each other.
- A 1-byte *message ID* field defines the message that is to follow.
- A 2-byte *length* field follows. The length is defined as being that of the payload only. It does not include the preamble, message class, message ID, length, or UBX checksum fields. The number format of the length field is an unsigned little-endian 16-bit integer (a "U2" in UBX data types).
- The *payload* field contains a variable number (= *length*) of bytes.
- The two 1-byte *CK_A* and *CK_B* fields hold a 16-bit checksum whose calculation is defined in UBX checksum section. This concludes the frame.

## 2.3 UBX payload definition rules

This section contains the rules and guidelines for UBX message payloads. See also UBX message example.

### 2.3.1 UBX structure packing

Values are placed in such an order that structure packing is not a problem. This means that two-byte values shall start on offsets that are a multiple of two; four-byte values shall start at a multiple of four; and so on.

### 2.3.2 UBX reserved elements

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

For fields in a bitfield the same rules apply. Note that bits not described are automatically reserved and are not explicitly stated (see UBX message example).

### 2.3.3 UBX undefined values

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

### 2.3.4 UBX conditional values

Some UBX messages use validity flag fields to indicate whether the values of some value fields are valid. For example the UBX-NAV-PVT message has the `validDate` and `validTime` fields that indicate whether the date (`year`, `month` and `day` fields), and, respectively, the time (`hour`, `min` and `sec` fields) are valid. This means that these value fields will only contain meaningful data if the corresponding flag field is set (has the value 1).

### 2.3.5 UBX data types

The following data types (number formats) are defined.

| Name | Type | Size (Bytes) | Range | Resolution |
|---|---|---|---|---|
| U1 | unsigned 8-bit integer | 1 | $0...2^8-1$ | 1 |
| I1 | signed 8-bit integer, two's complement | 1 | $-2^7...2^7-1$ | 1 |
| X1 | 8-bit bitfield | 1 | n/a | n/a |
| U2 | unsigned little-endian 16-bit integer | 2 | $0...2^{16}-1$ | 1 |
| I2 | signed little-endian 16-bit integer, two's complement | 2 | $-2^{15}...2^{15}-1$ | 1 |
| X2 | 16-bit little-endian bitfield | 2 | n/a | n/a |
| U4 | unsigned little-endian 32-bit integer | 4 | $0...2^{32}-1$ | 1 |
| I4 | signed little-endian 32-bit integer, two's complement | 4 | $-2^{31}...2^{31}-1$ | 1 |
| X4 | 32-bit little-endian bitfield | 4 | n/a | n/a |

| Name | Type | Size (Bytes) | Range | Resolution |
|---|---|---|---|---|
| R4 | IEEE 754 single (32-bit) precision | 4 | $-2^{127}...2^{127}$ | ~ value·$2^{-24}$ |
| R8 | IEEE 754 double (64-bit) precision | 8 | $-2^{1023}...2^{1023}$ | ~ value·$2^{-53}$ |
| CH | ASCII / ISO 8859-1 char (8-bit) | 1 | n/a | n/a |
| $U_{:n}$ | unsigned bitfield value of *n* bits width | var. | variable | variable |
| $I_{:n}$ | signed (two's complement) bitfield value of *n* bits width | var. | variable | variable |
| $S_{:n}$ | signed bitfield value of *n* bits width, in sign (most significant bit) and magnitude (remaining bits) notation | var. | variable | variable |

### 2.3.6 UBX fields scale and unit

Fields in UBX messages can have a unit defined. Whenever possible, SI units and symbols are used (e.g. "m" for meters, "s" for seconds). For civil (UTC) time representation units of years (y), months (month), days (d), hours (h), minutes (min) and seconds (s) are used.

Fields in UBX messages can have a scale factor defined. Unity (factor 1) is assumed if no scale is specified. For integer type fields this is often combined with a unit. When a scale is combined with a unit, the scale represents the smallest storage unit. For example, if meters (m) are expressed (stored) in centimeters the scale would be 0.01 (or 1e-2). This is equivalent of specifying a unit of centimeters (cm) and no scale.

### 2.3.7 UBX repeated fields

There are two types of repetitions in UBX messages. The first type specifies that a single field is repeated a constant number of times. This repetition is defined in the type of the field. For example, the UBX message example can specify a field `data` of type *U1[5]*. In this case the `data` field should be interpreted as an array of five U1 values.

The second type of repetition in messages is referred to as *repeated groups*, which groups one or more fields into a block of payload data. There are several types of repetition:

- The number of repetitions of *variable-by-field group* is indicated by another, earlier field in the same message. The number of repetitions can be zero or more, depending on the value of the referenced field.
- A *constant group* has a constant number of repetitions.
- An *optional group* is repeated zero or one times, depending on the available payload data. That is, the fields are present in the message only if the payload of the message is large enough to cover the whole group of fields.
- The number of repetitions of a *variable-by-size* group is given by the available payload size. The group will repeat until there is not enough payload data left to cover the whole group of fields another time.

Note that only some combinations of repeated groups of fields are possible in a single message. See also UBX payload decoding.

### 2.3.8 UBX payload decoding

UBX message payloads are designed so that the data (fields) can be extracted by a single pass through the payload from start to end. Fixed-size messages are the trivial case where the offset of all fields is unambiguously defined. Variable-size messages have variable number of repetitions of one or multiple groups of fields. For groups where the number of repetitions is given by the value of another field, that field can always be found at a fixed offset in the message payload before the respective group of fields. Groups whose number of repetitions depend on the payload size can only

be the last group of fields in a message and only one such group may exist in a message. See also UBX repeated fields.

## 2.4 UBX checksum

The checksum is calculated over the message, starting and including the class field up until, but excluding, the checksum fields (see the figure UBX frame structure).

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]` is an array of bytes that contains the data over which the checksum is to be calculated.
- The two `CK_A` and `CK_A` 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 the value `0xff` after both operations in the loop.
- After the loop, the two *U1* values contain the checksum, transmitted after the message payload, which concludes the frame.

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

## 2.5 UBX message flow

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

### 2.5.1 UBX acknowledgement

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

Some messages from other classes also use the same acknowledgement mechanism.

### 2.5.2 UBX polling mechanism

All messages that are output by the receiver in a periodic manner (i.e. messages in classes UBX-MON, UBX-NAV and UBX-RXM) and Get/Set type messages, such as the configuration messages in the UBX-CFG class, can also be polled.

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

## 2.6 GNSS, satellite and signal numbering

See GNSS, satellite and signal identifiers for details on how GNSS, satellites and signals are numbered in the UBX protocol.

## 2.7 UBX message example

This is an example of the definition of UBX messages as shown in the following sections.

| | | | | | | |
|---|---|---|---|---|---|---|
| ***Message*** ❶ | **UBX-DEMO-EXAMPLE**<br>**Example demo message** | | | | | |
| *Type* ❷ | Periodic/polled | | | | | |
| *Comment* ❸ | This is a comment that describes the use of the demo example message.<br>There can be references to other sections in the documentation (such as: UBX protocol).<br>☞ Note that there can be important remarks here. | | | | | |
| *Message* ❹ | *Header* | *Class ID* | | *Length (bytes)* | *Payload* | *Checksum* |
| *Structure* | 0xb5 0x62 | 0x01 | 0x07 | 16 + numRepeat*4 | *see below* | CK_A CK_B |

*Payload description:* ❺

| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
|---|---|---|---|---|---|
| 0 | U4 | `aField` | - | - | a field that contains an unsigned integer with no particular scale or unit |
| 4 | I4 | `anotherField` | 1e-2 | m | a field that contains a length in meters (m) with a scale of 1e-2 (= 0.01), i.e. a length in centimeters |
| 8 | X2 | `bitfield` ❻ | - | - | this field contains flags or values smaller than one byte, whose definition follows below (bits not described are reserved) |
| bit 0 | U:1 | `aFieldValid` | - | - | the first bit in `bitfield` indicates whether the `aField` is valid or not (see UBX conditional values) |
| bit 1 | U:1 | `someFlag` | - | - | the second bit is a flag (1 = true, 0 = false) |
| bits 5...2 | U:4 | `aBitFieldValue` | - | - | a 4-bits value (range: 0...15) |
| 10 | U1[5] ❼ | `reserved0` | - | - | a reserved field, whose value shall be ignored (in output messages) or set to 0 (in input messages) |
| 15 | U1 | `numRepeat` | - | - | number of repetitions in the group of fields below |
| *Start of repeated group (`numRepeat` times)* ❽ | | | | | |
| 16 + n*4 | I2 | `someValue` | - | - | a signed value in a repeated group of fields |
| 18 + n*4 | U2 | `anotherValue` | - | - | another value in a repeated group of fields |
| *End of repeated group (`numRepeat` times)* | | | | | |

❶ The first line shows the message name (see Naming). The second line shows a short description of the message.

❷ The message type (see Message types).

❸ This section contains comments that describe the message. Often links to other related sections in the documentation or other related messages are found here.

❹ The message structure gives the parameters for the UBX frame structure, notably the message class and message ID values and the payload length. For many messages the payload length is a fixed number (of bytes). Messages that contain repeated blocks of information (fields) have a variable payload (see UBX repeated fields).

❺ The message payload definition is given as a list of fields and their parameters. Each field starts at a specified offset (in bytes) in the payload (see also UBX structure packing), is of a specific type

(see UBX data types), has a unique name (within the message), and a description. Optionally, fields can have a scale and/or a unit (see UBX fields scale and unit).

❻ Bitfields ("X" types) are broken down into smaller parts. Each part can be one or more bits wide. Values that are two or more bits wide can be unsigned or one of two signed value representation (see UBX data types). Note that the ten unused bits 15...6 are not explicitly stated as UBX reserved elements.

❼ Fields can be arrays of values of the same type (see UBX repeated fields).

❽ Groups of fields can be repeated in the payload. The number of repetitions can be given by another field in the message (this example), a constant number, zero or one times (known as "optional group"), or derived from the remaining payload size (labeled as "repeated N times"). See also UBX repeated fields and UBX payload decoding.

## 2.8 UBX messages overview

| *Message* | *Class/ID* | *Description (Type)* |
|---|---|---|
| **UBX-ACK – Acknowledgement and negative acknowledgement messages** | | |
| UBX-ACK-ACK | 0x05 0x01 | • Message acknowledged (Output) |
| UBX-ACK-NAK | 0x05 0x00 | • Message not acknowledged (Output) |
| **UBX-CFG – Configuration and command messages** | | |
| UBX-CFG-PRT | 0x06 0x00 | • Polls the configuration for one I/O port (Poll request)<br>• Port configuration for UART ports (Get/set)<br>• Port configuration for USB port (Get/set)<br>• Port configuration for SPI port (Get/set)<br>• Port configuration for I2C (DDC) port (Get/set) |
| UBX-CFG-RST | 0x06 0x04 | • Reset receiver / Clear backup data structures (Command) |
| UBX-CFG-VALDEL | 0x06 0x8c | • Delete configuration item values (Set)<br>• Delete configuration item values (with transaction) (Set) |
| UBX-CFG-VALGET | 0x06 0x8b | • Get configuration items (Poll request)<br>• Configuration items (Polled) |
| UBX-CFG-VALSET | 0x06 0x8a | • Set configuration item values (Set)<br>• Set configuration item values (with transaction) (Set) |
| **UBX-INF – Information messages** | | |
| UBX-INF-ERROR | 0x04 0x00 | • ASCII output with error contents (Output) |
| UBX-INF-NOTICE | 0x04 0x02 | • ASCII output with informational contents (Output) |
| UBX-INF-WARNING | 0x04 0x01 | • ASCII output with warning contents (Output) |
| **UBX-MON – Monitoring messages** | | |
| UBX-MON-HW2 | 0x0a 0x0b | • Extended hardware status (Periodic/polled) |
| UBX-MON-VER | 0x0a 0x04 | • Poll receiver and software version (Poll request)<br>• Receiver and software version (Polled) |
| **UBX-RXM – Receiver manager messages** | | |
| UBX-RXM-PMP | 0x02 0x72 | • PMP raw data (Periodic) |
| UBX-RXM-PMREQ | 0x02 0x41 | • Power management request (Command) |

# 2.9 UBX-ACK (0x05)

The messages in the UBX-ACK class are used to indicate acknowledgement or rejection (i.e. negative acknowledgement) of input messages, such as UBX-CFG messages.

## 2.9.1 UBX-ACK-ACK (0x05 0x01)

### 2.9.1.1 Message acknowledged

| ***Message*** | **UBX-ACK-ACK**<br>**Message acknowledged** | | | | | |
|---|---|---|---|---|---|---|
| *Type* | Output | | | | | |
| *Comment* | Output upon processing of an input message. A UBX-ACK-ACK is sent as soon as possible but at least within one second. | | | | | |
| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
| | 0xb5 0x62 | 0x05 | 0x01 | 2 | *see below* | CK_A CK_B |

*Payload description:*

| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
|---|---|---|---|---|---|
| 0 | U1 | `clsID` | - | - | Class ID of the Acknowledged Message |
| 1 | U1 | `msgID` | - | - | Message ID of the Acknowledged Message |

## 2.9.2 UBX-ACK-NAK (0x05 0x00)

### 2.9.2.1 Message not acknowledged

| ***Message*** | **UBX-ACK-NAK**<br>**Message not acknowledged** | | | | | |
|---|---|---|---|---|---|---|
| *Type* | Output | | | | | |
| *Comment* | Output upon processing of an input message. A UBX-ACK-NAK is sent as soon as possible but at least within one second. | | | | | |
| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
| | 0xb5 0x62 | 0x05 | 0x00 | 2 | *see below* | CK_A CK_B |

*Payload description:*

| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
|---|---|---|---|---|---|
| 0 | U1 | `clsID` | - | - | Class ID of the Not-Acknowledged Message |
| 1 | U1 | `msgID` | - | - | Message ID of the Not-Acknowledged Message |

# 2.10 UBX-CFG (0x06)

The messages in the UBX-CFG class are used to configure the receiver and poll current configuration values as well as for sending commands to the receiver. Unless stated otherwise, any message in this class sent to the receiver is either acknowledged (by a UBX-ACK-ACK message) if processed successfully or rejected (with a UBX-ACK-NAK message) if processed unsuccessfully.

## 2.10.1 UBX-CFG-PRT (0x06 0x00)

### 2.10.1.1 Polls the configuration for one I/O port

| ***Message*** | **UBX-CFG-PRT**<br>**Polls the configuration for one I/O port** |
|---|---|
| *Type* | Poll request |

| | |
|---|---|
| *Comment* | **This message is deprecated in protocol versions greater than 23.01. Use** UBX-CFG-VALSET, UBX-CFG-VALGET, UBX-CFG-VALDEL **instead.**<br>See the Legacy UBX Message Fields Reference for the corresponding configuration item.<br>Sending this message with a port ID as payload results in having the receiver return the configuration for the specified port. |

| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
|---|---|---|---|---|---|---|
| | 0xb5 0x62 | 0x06 | 0x00 | 1 | *see below* | CK_A CK_B |

*Payload description:*

| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
|---|---|---|---|---|---|
| 0 | U1 | `PortID` | - | - | Port identifier number (see the other versions of CFG-PRT for valid values) |

### 2.10.1.2 Port configuration for UART ports

| ***Message*** | **UBX-CFG-PRT**<br>**Port configuration for UART ports** |
|---|---|
| *Type* | Get/set |
| *Comment* | **This message is deprecated in protocol versions greater than 23.01. Use** UBX-CFG-VALSET, UBX-CFG-VALGET, UBX-CFG-VALDEL **instead.**<br>See the Legacy UBX Message Fields Reference for the corresponding configuration item.<br>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.<br>Note that this message can affect baud rate and other transmission parameters. Because there may be messages queued for transmission there may be uncertainty about which protocol applies to such messages. In addition a message currently in transmission may be corrupted by a protocol change. Host data reception parameters may have to be changed to be able to receive future messages, including the acknowledge message resulting from the CFG-PRT message. |

| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
|---|---|---|---|---|---|---|
| | 0xb5 0x62 | 0x06 | 0x00 | 20 | *see below* | CK_A CK_B |

*Payload description:*

| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
|---|---|---|---|---|---|
| 0 | U1 | `portID` | - | - | Port identifier number (see the integration manual for valid UART port IDs) |
| 1 | U1 | `reserved0` | - | - | Reserved |
| 2 | X2 | `txReady` | - | - | |
| bit 0 | U:1 | `en` | - | - | Enable TX ready feature for this port |
| bit 1 | U:1 | `pol` | - | - | Polarity<br>• 0 High-active<br>• 1 Low-active |
| bits 6...2 | U:5 | `pin` | - | - | PIO to be used (must not be in use by another function) |
| bits 15...7 | U:9 | `thres` | - | - | Threshold<br>The given threshold is multiplied by 8 bytes.<br>The TX ready PIN goes active after >= thres*8 bytes are pending for the port and going inactive after the last pending bytes have been written to hardware (0-4 bytes before end of stream).<br>• 0x000 no threshold<br>• 0x001 8byte<br>• 0x002 16byte<br>• ...<br>• 0x1FE 4080byte |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | • 0x1FF 4088byte |
| 4 | X4 | mode | - | - | A bit mask describing the UART mode |
| bits 7...6 | U:2 | charLen | - | - | Character length<br>• 00 5bit (not supported)<br>• 01 6bit (not supported)<br>• 10 7bit (supported only with parity)<br>• 11 8bit |
| bits 11...9 | U:3 | parity | - | - | • 000 Even parity<br>• 001 Odd parity<br>• 10X No parity<br>• X1X Reserved |
| bits 13...12 | U:2 | nStopBits | - | - | Number of Stop bits<br>• 00 1 Stop bit<br>• 01 1.5 Stop bit<br>• 10 2 Stop bit<br>• 11 0.5 Stop bit |
| 8 | U4 | baudRate | - | Bits/s | Baud rate in bits/second |
| 12 | X2 | inProtoMask | - | - | A mask describing which input protocols are active.<br>Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port. |
| bit 0 | U:1 | inUbx | - | - | UBX protocol |
| bit 1 | U:1 | inNmea | - | - | NMEA protocol |
| bit 2 | U:1 | inRtcm | - | - | RTCM2 protocol |
| bit 5 | U:1 | inRtcm3 | - | - | RTCM3 protocol (not supported for protocol versions less than 20.00) |
| 14 | X2 | outProtoMask | - | - | A mask describing which output protocols are active.<br>Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port. |
| bit 0 | U:1 | outUbx | - | - | UBX protocol |
| bit 1 | U:1 | outNmea | - | - | NMEA protocol |
| bit 5 | U:1 | outRtcm3 | - | - | RTCM3 protocol (not supported for protocol versions less than 20.00) |
| 16 | X2 | flags | - | - | Flags bit mask |
| bit 1 | U:1 | extendedTx Timeout | - | - | Extended TX timeout: if set, the port will time out if allocated TX memory >=4 kB and no activity for 1.5 s. If not set the port will time out if no activity for 1.5 s regardless on the amount of allocated TX memory (not supported for protocol versions less than 13.01). |
| 18 | U1[2] | reserved1 | - | - | Reserved |

### 2.10.1.3 Port configuration for USB port

| ***Message*** | **UBX-CFG-PRT**<br>**Port configuration for USB port** |
|---|---|
| *Type* | Get/set |
| *Comment* | **This message is deprecated in protocol versions greater than 23.01. Use** UBX-CFG-VALSET, UBX-CFG-VALGET, UBX-CFG-VALDEL **instead.**<br>See the Legacy UBX Message Fields Reference for the corresponding configuration item.<br>Several configurations can be concatenated to one input message. In this case the payload length can be a multiple of the normal length (see the other versions of CFG-PRT). Output messages from the module contain only one configuration unit. |

| Message structure | Header | Class | ID | Length (Bytes) | Payload | Checksum |
|---|---|---|---|---|---|---|
| | 0xb5 0x62 | 0x06 | 0x00 | 20 | see below | CK_A CK_B |

*Payload description:*

| Byte offset | Type | Name | Scale | Unit | Description |
|---|---|---|---|---|---|
| 0 | U1 | portID | - | - | Port identifier number (= 3 for USB port) |
| 1 | U1 | reserved0 | - | - | Reserved |
| 2 | X2 | txReady | - | - | |
| bit 0 | U:1 | en | - | - | Enable TX ready feature for this port |
| bit 1 | U:1 | pol | - | - | Polarity<br>• 0 High-active<br>• 1 Low-active |
| bits 6...2 | U:5 | pin | - | - | PIO to be used (must not be in use by another function) |
| bits 15...7 | U:9 | thres | - | - | Threshold<br>The given threshold is multiplied by 8 bytes.<br>The TX ready PIN goes active after >= thres*8 bytes are pending for the port and going inactive after the last pending bytes have been written to hardware (0-4 bytes before end of stream).<br>• 0x000 no threshold<br>• 0x001 8byte<br>• 0x002 16byte<br>• ...<br>• 0x1FE 4080byte<br>• 0x1FF 4088byte |
| 4 | U1[8] | reserved1 | - | - | Reserved |
| 12 | X2 | inProtoMask | - | - | A mask describing which input protocols are active.<br>Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port. |
| bit 0 | U:1 | inUbx | - | - | UBX protocol |
| bit 1 | U:1 | inNmea | - | - | NMEA protocol |
| bit 2 | U:1 | inRtcm | - | - | RTCM2 protocol |
| bit 5 | U:1 | inRtcm3 | - | - | RTCM3 protocol (not supported for protocol versions less than 20.00) |
| 14 | X2 | outProtoMask | - | - | A mask describing which output protocols are active.<br>Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port. |
| bit 0 | U:1 | outUbx | - | - | UBX protocol |
| bit 1 | U:1 | outNmea | - | - | NMEA protocol |
| bit 5 | U:1 | outRtcm3 | - | - | RTCM3 protocol (not supported for protocol versions less than 20.00) |
| 16 | U1[2] | reserved2 | - | - | Reserved |
| 18 | U1[2] | reserved3 | - | - | Reserved |

### 2.10.1.4 Port configuration for SPI port

| ***Message*** | **UBX-CFG-PRT** |
|---|---|
| | **Port configuration for SPI port** |
| *Type* | Get/set |

| | |
|---|---|
| *Comment* | **This message is deprecated in protocol versions greater than 23.01. Use** UBX-CFG-VALSET, UBX-CFG-VALGET, UBX-CFG-VALDEL **instead.**<br>See the Legacy UBX Message Fields Reference for the corresponding configuration item.<br>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. |

| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
|---|---|---|---|---|---|---|
| | 0xb5 0x62 | 0x06 | 0x00 | 20 | *see below* | CK_A CK_B |

*Payload description:*

| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
|---|---|---|---|---|---|
| 0 | U1 | `portID` | - | - | Port identifier number (= 4 for SPI port) |
| 1 | U1 | `reserved0` | - | - | Reserved |
| 2 | X2 | `txReady` | - | - | |
| bit 0 | U:1 | `en` | - | - | Enable TX ready feature for this port |
| bit 1 | U:1 | `pol` | - | - | Polarity<br>• 0 High-active<br>• 1 Low-active |
| bits 6...2 | U:5 | `pin` | - | - | PIO to be used (must not be in use by another function) |
| bits 15...7 | U:9 | `thres` | - | - | Threshold<br>The given threshold is multiplied by 8 bytes.<br>The TX ready PIN goes active after >= thres*8 bytes are pending for the port and going inactive after the last pending bytes have been written to hardware (0-4 bytes before end of stream).<br>• 0x000 no threshold<br>• 0x001 8byte<br>• 0x002 16byte<br>• ...<br>• 0x1FE 4080byte<br>• 0x1FF 4088byte |
| 4 | X4 | `mode` | - | - | SPI Mode Flags |
| bits 2...1 | U:2 | `spiMode` | - | - | • 00 SPI Mode 0: CPOL = 0, CPHA = 0<br>• 01 SPI Mode 1: CPOL = 0, CPHA = 1<br>• 10 SPI Mode 2: CPOL = 1, CPHA = 0<br>• 11 SPI Mode 3: CPOL = 1, CPHA = 1 |
| bits 13...8 | U:6 | `ffCnt` | - | - | Number of bytes containing 0xFF to receive before switching off reception. Range: 0 (mechanism off) - 63 |
| 8 | U1[4] | `reserved1` | - | - | Reserved |
| 12 | X2 | `inProtoMask` | - | - | A mask describing which input protocols are active.<br>Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port.<br>(The bitfield inRtcm3 is not supported for protocol versions less than 20.00) |
| bit 0 | U:1 | `inUbx` | - | - | |
| bit 1 | U:1 | `inNmea` | - | - | |
| bit 2 | U:1 | `inRtcm` | - | - | |
| bit 5 | U:1 | `inRtcm3` | - | - | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 14 | | X2 | outProtoMask | - | - | A mask describing which output protocols are active.<br>Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port.<br>(The bitfield outRtcm3 is not supported for protocol versions less than 20.00) |
| | bit 0 | U:1 | outUbx | - | - | |
| | bit 1 | U:1 | outNmea | - | - | |
| | bit 5 | U:1 | outRtcm3 | - | - | |
| 16 | | X2 | flags | - | - | Flags bit mask |
| | bit 1 | U:1 | extendedTx Timeout | - | - | Extended TX timeout: if set, the port will time out if allocated TX memory >=4 kB and no activity for 1.5 s.<br>(not supported for protocol versions less than 13.01) |
| 18 | | U1[2] | reserved2 | - | - | Reserved |

### 2.10.1.5 Port configuration for I2C (DDC) port

| ***Message*** | **UBX-CFG-PRT**<br>**Port configuration for I2C (DDC) port** |
|---|---|
| *Type* | Get/set |
| *Comment* | **This message is deprecated in protocol versions greater than 23.01. Use** UBX-CFG-VALSET, UBX-CFG-VALGET, UBX-CFG-VALDEL **instead.**<br>See the Legacy UBX Message Fields Reference for the corresponding configuration item.<br>Several configurations can be concatenated to one input message. In this case the payload length can be a multiple of the normal length (see the other versions of CFG-PRT). Output messages from the module contain only one configuration unit. |

| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
|---|---|---|---|---|---|---|
| | 0xb5 0x62 | 0x06 | 0x00 | 20 | *see below* | CK_A CK_B |

*Payload description:*

| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
|---|---|---|---|---|---|
| 0 | U1 | portID | - | - | Port identifier number (= 0 for I2C (DDC) port) |
| 1 | U1 | reserved0 | - | - | Reserved |
| 2 | X2 | txReady | - | - | TX ready PIN configuration (not supported for protocol versions 24.00, and 29.00) |
| bit 0 | U:1 | en | - | - | Enable TX ready feature for this port |
| bit 1 | U:1 | pol | - | - | Polarity<br>• 0 High-active<br>• 1 Low-active |
| bits 6...2 | U:5 | pin | - | - | PIO to be used (must not be in use by another function) |
| bits 15...7 | U:9 | thres | - | - | Threshold<br>The given threshold is multiplied by 8 bytes.<br>The TX ready PIN goes active after >= thres*8 bytes are pending for the port and going inactive after the last pending bytes have been written to hardware (0-4 bytes before end of stream).<br>• 0x000 no threshold<br>• 0x001 8byte<br>• 0x002 16byte<br>• ...<br>• 0x1FE 4080byte<br>• 0x1FF 4088byte |
| 4 | X4 | mode | - | - | I2C (DDC) Mode Flags |

| | | | | | | |
|---|---|---|---|---|---|---|
| | bits 7...1 | U:7 | slaveAddr | - | - | Slave address<br>Range: 0x07 < slaveAddr < 0x78. Bit 0 must be 0 |
| 8 | | U1[4] | reserved1 | - | - | Reserved |
| 12 | | X2 | inProtoMask | - | - | A mask describing which input protocols are active.<br>Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port.<br>(The bitfield inRtcm3 is not supported for protocol versions less than 20.00) |
| | bit 0 | U:1 | inUbx | - | - | |
| | bit 1 | U:1 | inNmea | - | - | |
| | bit 2 | U:1 | inRtcm | - | - | |
| | bit 5 | U:1 | inRtcm3 | - | - | |
| 14 | | X2 | outProtoMask | - | - | A mask describing which output protocols are active.<br>Each bit of this mask is used for a protocol. Through that, multiple protocols can be defined on a single port.<br>(The bitfield outRtcm3 is not supported for protocol versions less than 20.00) |
| | bit 0 | U:1 | outUbx | - | - | |
| | bit 1 | U:1 | outNmea | - | - | |
| | bit 5 | U:1 | outRtcm3 | - | - | |
| 16 | | X2 | flags | - | - | Flags bit mask |
| | bit 1 | U:1 | extendedTx Timeout | - | - | Extended TX timeout: if set, the port will time out if allocated TX memory >=4 kB and no activity for 1.5 s (not supported for protocol versions less than 13.01). |
| 18 | | U1[2] | reserved2 | - | - | Reserved |

## 2.10.2 UBX-CFG-RST (0x06 0x04)

### 2.10.2.1 Reset receiver / Clear backup data structures

| ***Message*** | **UBX-CFG-RST**<br>**Reset receiver / Clear backup data structures** | | | | | |
|---|---|---|---|---|---|---|
| *Type* | Command | | | | | |
| *Comment* | Do not expect this message to be acknowledged by the receiver.<br>• Newer FW version will not acknowledge this message at all.<br>• Older FW version will acknowledge this message but the acknowledge may not be sent completely before the receiver is reset. | | | | | |
| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
| | 0xb5 0x62 | 0x06 | 0x04 | 4 | *see below* | CK_A CK_B |

*Payload description:*

| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
|---|---|---|---|---|---|
| 0 | X2 | navBbrMask | - | - | BBR sections to clear. The following special sets apply:<br>• 0x0000 Hot start<br>• 0x0001 Warm start<br>• 0xFFFF Cold start |
| bit 0 | U:1 | eph | - | - | Ephemeris |
| bit 1 | U:1 | alm | - | - | Almanac |
| bit 2 | U:1 | health | - | - | Health |

| | | | | | |
|---|---|---|---|---|---|
| bit 3 | U:1 | klob | - | - | Klobuchar parameters |
| bit 4 | U:1 | pos | - | - | Position |
| bit 5 | U:1 | clkd | - | - | Clock drift |
| bit 6 | U:1 | osc | - | - | Oscillator parameter |
| bit 7 | U:1 | utc | - | - | UTC correction + GPS leap seconds parameters |
| bit 8 | U:1 | rtc | - | - | RTC |
| bit 11 | U:1 | sfdr | - | - | SFDR Parameters (only available on the ADR/UDR/ HPS product variant) and weak signal compensation estimates |
| bit 12 | U:1 | vmon | - | - | SFDR Vehicle Monitoring Parameter (only available on the ADR/UDR/HPS product variant) |
| bit 13 | U:1 | tct | - | - | TCT Parameters (only available on the ADR/UDR/HPS product variant) |
| bit 15 | U:1 | aop | - | - | Autonomous orbit parameters |
| 2 | U1 | resetMode | - | - | Reset Type<br>• 0x00 = Hardware reset (watchdog) immediately<br>• 0x01 = Controlled software reset<br>• 0x02 = Controlled software reset (GNSS only)<br>• 0x04 = Hardware reset (watchdog) after shutdown<br>• 0x08 = Controlled GNSS stop<br>• 0x09 = Controlled GNSS start |
| 3 | U1 | reserved0 | - | - | Reserved |

## 2.10.3 UBX-CFG-VALDEL (0x06 0x8c)

### 2.10.3.1 Delete configuration item values

| ***Message*** | **UBX-CFG-VALDEL**<br>**Delete configuration item values** | | | | | |
|---|---|---|---|---|---|---|
| *Type* | Set | | | | | |
| *Comment* | Overview:<br>• This message can be used to delete saved configuration to effectively revert the item values to defaults.<br>• This message can delete saved configuration from the flash configuration layer and the BBR configuration layer. The changes will not be effective until these layers are loaded into the RAM layer.<br>• This message is limited to containing a maximum of 64 keys up for deletion; i.e. N is a maximum of 64.<br>• This message can be used multiple times and every time the result will be applied immediately. To send this message multiple times with the result being applied at the end, see version 1 of UBX-CFG-VALDEL that supports transactions.<br>• This message does not check if the resulting configuration is valid.<br>• See Receiver configuration for details.<br>This message returns a UBX-ACK-NAK and no configuration is applied:<br>• if any key is unknown to the receiver FW<br>• if the layer's bitfield does not specify a layer to delete a value from.<br>Notes:<br>• If a key is sent multiple times within the same message, then the value is effectively deleted only once.<br>• Attempting to delete items that have not been set before, or that have already been deleted, is considered a valid request. | | | | | |
| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
| | 0xb5 0x62 | 0x06 | 0x8c | 4 + [0..n]·4 | *see below* | CK_A CK_B |

*Payload description:*

| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| 0 | U1 | version | - | - | Message version (0x00 for this version) |
| 1 | X1 | layers | - | - | The layers where the configuration should be deleted from |
| bit 1 | U:1 | bbr | - | - | Delete configuration from the BBR layer |
| bit 2 | U:1 | flash | - | - | Delete configuration from the Flash layer |
| 2 | U1[2] | reserved0 | - | - | Reserved |
| *Start of repeated group (N times)* | | | | | |
| 4 + n·4 | U4 | keys | - | - | Configuration key IDs of the configuration items to be deleted |
| *End of repeated group (N times)* | | | | | |

### 2.10.3.2 Delete configuration item values (with transaction)

| ***Message*** | **UBX-CFG-VALDEL**<br>**Delete configuration item values (with transaction)** |
|---|---|
| *Type* | Set |
| *Comment* | Overview:<br>• This message can be used to delete saved configuration to effectively revert them to defaults.<br>• This message can delete saved configuration from the flash configuration layer and the BBR configuration layer. The changes will not be effective until these layers are loaded into the RAM layer.<br>• This message is limited to containing a maximum of 64 keys up for deletion; i.e. N is a maximum of 64.<br>• This message can be used multiple times with the result being managed within a transaction.<br>• This message does not check if the resulting configuration is valid.<br>• See Receiver configuration for details.<br>• See version 0 of UBX-CFG-VALDEL for simplified version of this message.<br>This message returns a UBX-ACK-NAK, cancels any started transaction, and no configuration is applied:<br>• if any key within a transaction is unknown to the receiver FW<br>• if an invalid transaction state transition is requested<br>• if the layer's bitfield changes within a transaction<br>• if the layer's bitfield does not specify a layer to delete a value from.<br>Notes:<br>• Any request for another UBX-CFG- message type (including UBX-CFG-VALSET and UBX-CFG-VALGET) will cancel any started transaction, and no configuration is applied.<br>• This message can be sent with no keys to delete for the purposes of managing the transaction state transition.<br>• If a key is sent multiple times within the same message or within the same transaction, then the value is effectively deleted only once.<br>• Attempting to delete items that have not been set before, or that have already been deleted, is considered a valid request. |

| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
|---|---|---|---|---|---|---|
| | 0xb5 0x62 | 0x06 | 0x8c | 4 + [0..n]·4 | *see below* | CK_A CK_B |

*Payload description:*

| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
|---|---|---|---|---|---|
| 0 | U1 | version | - | - | Message version (0x01 for this version) |
| 1 | X1 | layers | - | - | The layers where the configuration should be deleted from |
| bit 1 | U:1 | bbr | - | - | Delete configuration from the BBR layer |
| bit 2 | U:1 | flash | - | - | Delete configuration from the Flash layer |
| 2 | X1 | transaction | - | - | Transaction action to be applied: |
| bits 1...0 | U:2 | action | - | - | Transaction action to be applied:<br>• 0 = Transactionless UBX-CFG-VALDEL: In the next UBX-CFG-VALDEL, it can be either 0 or 1. |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | If a transaction has not yet been started, the incoming configuration is applied. If a transaction has already been started, cancels any started transaction and the incoming configuration is applied.<br>• 1 = (Re)Start deletion transaction: In the next UBX-CFG-VALDEL, it can be either 0, 1, 2 or 3. If a transaction has not yet been started, a transaction will be started. If a transaction has already been started, restarts the transaction, effectively removing all previous non-applied UBX-CFG-VALDEL messages.<br>• 2 = Deletion transaction ongoing: In the next UBX-CFG-VALDEL, it can be either 0, 1, 2 or 3.<br>• 3 = Apply and end a deletion transaction: In the next UBX-CFG-VALDEL, it can be either 0 or 1. |
| 3 | U1 | `reserved0` | - | - | Reserved |
| *Start of repeated group (N times)* | | | | | |
| 4 + n·4 | U4 | `keys` | - | - | Configuration key IDs of the configuration items to be deleted |
| *End of repeated group (N times)* | | | | | |

## 2.10.4 UBX-CFG-VALGET (0x06 0x8b)

### 2.10.4.1 Get configuration items

| ***Message*** | **UBX-CFG-VALGET**<br>**Get configuration items** | | | | | |
|---|---|---|---|---|---|---|
| *Type* | Poll request | | | | | |
| *Comment* | Overview:<br>• This message is used to get configuration values by providing a list of configuration key IDs, which identify the configuration items to retrieve.<br>• This message can specify the configuration layer where the values of the specified configuration items are retrieved from.<br>• This message is limited to containing a maximum of 64 key IDs.<br>• See Receiver configuration for details.<br>This message returns a UBX-ACK-NAK:<br>• if any key is unknown to the receiver FW<br>• if the layer field specifies an invalid layer to get the value from<br>• if the keys array specifies more than 64 key IDs.<br>Notes:<br>• If a value is requested multiple times within the same poll request, then the reply will contain it multiple times.<br>• The provided keys can be complete key values (group and item specifiers) or wild-card specifications. A complete key value will constitute a request for one key-value pair. A key value that has a valid group specifier and 0xffff in the item part of the key value (bits 0-15) constitutes a request for all items in the specified group. A key with a value of 0xfff in the group part of the key value (bits 16-27) is a request for all items known to the receiver in all groups.<br>• The response message is limited to containing a maximum of 64 key-value pairs. If there are wild-card specifications then there may be more than 64 possible responses. In order to handle this, the 'position' field can specify that the response message should skip this number of key-value pairs before it starts constructing the message. This allows a large set of values to be retrieved 64 at a time. If the response contains less than 64 key-value pairs then all values have been reported, otherwise there may be more to read.<br>• It is not possible to retrieve configuration values for the same configuration item from multiple configuration layers. Separate poll requests must be made for each desired layer. | | | | | |
| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
| | 0xb5 0x62 | 0x06 | 0x8b | 4 + [0..n]·4 | *see below* | CK_A CK_B |

| *Payload description:* | | | | | |
|---|---|---|---|---|---|
| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
| 0 | U1 | `version` | - | - | Message version (0x00 for this version) |
| 1 | U1 | `layer` | - | - | The layer from which the configuration items should be retrieved:<br>• 0 - RAM layer<br>• 1 - BBR layer<br>• 2 - Flash layer<br>• 7 - Default layer |
| 2 | U2 | `position` | - | - | Skip this many key values before constructing output message |
| *Start of repeated group (N times)* | | | | | |
| 4 + n·4 | U4 | `keys` | - | - | Configuration key IDs of the configuration items to be retrieved |
| *End of repeated group (N times)* | | | | | |

### 2.10.4.2 Configuration items

| ***Message*** | **UBX-CFG-VALGET**<br>**Configuration items** | | | | | |
|---|---|---|---|---|---|---|
| *Type* | Polled | | | | | |
| *Comment* | This message is output by the receiver to return requested configuration data (key and value pairs).<br>See Receiver configuration for details. | | | | | |
| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
| | 0xb5 0x62 | 0x06 | 0x8b | 4 + [0..n] | *see below* | CK_A CK_B |

| *Payload description:* | | | | | |
|---|---|---|---|---|---|
| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
| 0 | U1 | `version` | - | - | Message version (0x01 for this version) |
| 1 | U1 | `layer` | - | - | The layer from which the configuration item was retrieved:<br>• 0 - RAM layer<br>• 1 - BBR<br>• 2 - Flash<br>• 7 - Default |
| 2 | U2 | `position` | - | - | Number of configuration items skipped in the result set before constructing this message (mirrors the equivalent field in the request message) |
| *Start of repeated group (N times)* | | | | | |
| 4 + n | U1 | `cfgData` | - | - | Configuration data (key and value pairs) |
| *End of repeated group (N times)* | | | | | |

## 2.10.5 UBX-CFG-VALSET (0x06 0x8a)

### 2.10.5.1 Set configuration item values

| ***Message*** | **UBX-CFG-VALSET**<br>**Set configuration item values** |
|---|---|
| *Type* | Set |
| *Comment* | Overview:<br>• This message is used to set a configuration by providing configuration data (a list of key and value pairs), which identify the configuration items to change, and their new values. |

- This message is limited to containing a maximum of 64 key-value pairs.
- This message can be used multiple times and every time the result will be applied immediately. To send this message multiple times with the result being applied at the end, see version 1 of UBX-CFG-VALSET that supports transactions.
- See Receiver configuration for details.

This message returns a UBX-ACK-NAK and no configuration is applied:

- if any key is unknown to the receiver FW
- if the layer's bitfield does not specify a layer to save a value to
- if the requested configuration is not valid. The validity of a configuration is checked only if the message requests to apply the configuration to the RAM configuration layer.

Notes:

- If a key is sent multiple times within the same message, then the value eventually being applied is the last sent.

| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
|---|---|---|---|---|---|---|
| | 0xb5 0x62 | 0x06 | 0x8a | 4 + [0..n] | *see below* | CK_A CK_B |

*Payload description:*

| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
|---|---|---|---|---|---|
| 0 | U1 | `version` | - | - | Message version (0x00 for this version) |
| 1 | X1 | `layers` | - | - | The layers where the configuration should be applied |
| bit 0 | U:1 | `ram` | - | - | Update configuration in the RAM layer |
| bit 1 | U:1 | `bbr` | - | - | Update configuration in the BBR layer |
| bit 2 | U:1 | `flash` | - | - | Update configuration in the Flash layer |
| 2 | U1[2] | `reserved0` | - | - | Reserved |
| *Start of repeated group (N times)* | | | | | |
| 4 + n | U1 | `cfgData` | - | - | Configuration data (key and value pairs) |
| *End of repeated group (N times)* | | | | | |

### 2.10.5.2 Set configuration item values (with transaction)

| ***Message*** | **UBX-CFG-VALSET**<br>**Set configuration item values (with transaction)** |
|---|---|
| *Type* | Set |
| *Comment* | Overview:<br>• This message is used to set a configuration by providing configuration data (a list of key and value pairs), which identify the configuration items to change, and their new values.<br>• This message is limited to containing a maximum of 64 key-value pairs.<br>• This message can be used multiple times with the result being managed within a transaction. Within a transaction there is no limit on the number key-value pairs; a transaction is effectively limited to the number of known keys.<br>• See Receiver configuration for details.<br>• See version 0 of UBX-CFG-VALSET for simplified version of this message.<br>This message returns a UBX-ACK-NAK, cancels any started transaction, and no configuration is applied:<br>• if any key within a transaction is unknown to the receiver FW<br>• if an invalid transaction state transition is requested<br>• if the layer's bitfield changes within a transaction<br>• if the layer's bitfield does not specify a layer to save a value to<br>This message returns a UBX-ACK-NAK, and no configuration is applied:<br>• if the requested configuration is not valid. While in a transaction context, only the last message that requests to apply the transaction returns a UBX-ACK-NAK. The validity of a configuration is checked only if the message requests to apply the configuration to the RAM configuration layer. This also applies to a transactionless request.<br>Notes:<br>• Any request for another UBX-CFG-message type (including UBX-CFG-VALDEL and UBX-CFG-VALGET) will cancel any started transaction, and no configuration is applied. |

- This message can be sent with no key/values to set for the purposes of managing the transaction state transition.
- If a key is sent multiple times within the same message or within the same transaction, then the value eventually being applied is the last sent.

| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
|---|---|---|---|---|---|---|
| | 0xb5 0x62 | 0x06 | 0x8a | 4 + [0..n] | *see below* | CK_A CK_B |

*Payload description:*

| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
|---|---|---|---|---|---|
| 0 | U1 | `version` | - | - | Message version (0x01 for this version) |
| 1 | X1 | `layers` | - | - | The layers where the configuration should be applied |
| bit 0 | U:1 | `ram` | - | - | Update configuration in the RAM layer |
| bit 1 | U:1 | `bbr` | - | - | Update configuration in the BBR layer |
| bit 2 | U:1 | `flash` | - | - | Update configuration in the Flash layer |
| 2 | U1 | `transaction` | - | - | Transaction action to be applied |
| bits 1...0 | U:2 | `action` | - | - | Transaction action to be applied:<br>• 0 = Transactionless UBX-CFG-VALSET: In the next UBX-CFG-VALSET, it can be either 0 or 1. If a transaction has not yet been started, the incoming configuration is applied (if valid). If a transaction has already been started, cancels any started transaction and the incoming configuration is applied (if valid).<br>• 1 = (Re)Start set transaction: In the next UBX-CFG-VALSET, it can be either 0, 1, 2 or 3. If a transaction has not yet been started, a transaction will be started. If a transaction has already been started, restarts the transaction, effectively removing all previous non-applied UBX-CFG-VALSET messages.<br>• 2 = Set transaction ongoing: In the next UBX-CFG-VALSET, it can be either 0, 1, 2 or 3.<br>• 3 = Apply and end a set transaction: In the next UBX-CFG-VALSET, it can be either 0 or 1. |
| 3 | U1 | `reserved0` | - | - | Reserved |
| *Start of repeated group (N times)* | | | | | |
| 4 + n | U1 | `cfgData` | - | - | Configuration data (key and value pairs) |
| *End of repeated group (N times)* | | | | | |

# 2.11 UBX-INF (0x04)

Messages in the UBX-INF class are used to output strings from the firmware or application code. All messages have an associated type to indicate the nature or priority of the message.

## 2.11.1 UBX-INF-ERROR (0x04 0x00)

### 2.11.1.1 ASCII output with error contents

| ***Message*** | **UBX-INF-ERROR**<br>**ASCII output with error contents** | | | | | |
|---|---|---|---|---|---|---|
| *Type* | Output | | | | | |
| *Comment* | This message has a variable length payload, representing an ASCII string. | | | | | |
| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
| | 0xb5 0x62 | 0x04 | 0x00 | [0..n] | *see below* | CK_A CK_B |

| *Payload description:* | | | | | |
|---|---|---|---|---|---|
| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
| *Start of repeated group (N times)* | | | | | |
| 0 + n | CH | `str` | - | - | ASCII Character |
| *End of repeated group (N times)* | | | | | |

### 2.11.2 UBX-INF-NOTICE (0x04 0x02)

#### 2.11.2.1 ASCII output with informational contents

| ***Message*** | **UBX-INF-NOTICE**<br>**ASCII output with informational contents** | | | | | |
|---|---|---|---|---|---|---|
| *Type* | Output | | | | | |
| *Comment* | This message has a variable length payload, representing an ASCII string. | | | | | |
| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
| | 0xb5 0x62 | 0x04 | 0x02 | [0..n] | *see below* | CK_A CK_B |

| *Payload description:* | | | | | |
|---|---|---|---|---|---|
| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
| *Start of repeated group (N times)* | | | | | |
| 0 + n | CH | `str` | - | - | ASCII Character |
| *End of repeated group (N times)* | | | | | |

### 2.11.3 UBX-INF-WARNING (0x04 0x01)

#### 2.11.3.1 ASCII output with warning contents

| ***Message*** | **UBX-INF-WARNING**<br>**ASCII output with warning contents** | | | | | |
|---|---|---|---|---|---|---|
| *Type* | Output | | | | | |
| *Comment* | This message has a variable length payload, representing an ASCII string. | | | | | |
| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
| | 0xb5 0x62 | 0x04 | 0x01 | [0..n] | *see below* | CK_A CK_B |

| *Payload description:* | | | | | |
|---|---|---|---|---|---|
| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
| *Start of repeated group (N times)* | | | | | |
| 0 + n | CH | `str` | - | - | ASCII Character |
| *End of repeated group (N times)* | | | | | |

## 2.12 UBX-MON (0x0a)

The messages in the UBX-MON class are used to report the receiver status, such as hardware status or I/O subsystem statistics.

### 2.12.1 UBX-MON-HW2 (0x0a 0x0b)

#### 2.12.1.1 Extended hardware status

| ***Message*** | **UBX-MON-HW2**<br>**Extended hardware status** |
|---|---|
| *Type* | Periodic/polled |

| | |
|---|---|
| *Comment* | Status of different aspects of the hardware such as Imbalance, Low-Level Configuration and POST Results.<br>The first four parameters of this message represent the complex signal from the RF front end. The following rules of thumb apply:<br>• The smaller the absolute value of the variable `ofsI` and `ofsQ`, the better.<br>• Ideally, the magnitude of the I-part (`magI`) and the Q-part (`magQ`) of the complex signal should be the same. |

| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
|---|---|---|---|---|---|---|
| | 0xb5 0x62 | 0x0a | 0x0b | 28 | *see below* | CK_A CK_B |

*Payload description:*

| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
|---|---|---|---|---|---|
| 0 | I1 | `ofsI` | - | - | Imbalance of I-part of complex signal, scaled (-128 = max. negative imbalance, 127 = max. positive imbalance) |
| 1 | U1 | `magI` | - | - | Magnitude of I-part of complex signal, scaled (0 = no signal, 255 = max. magnitude) |
| 2 | I1 | `ofsQ` | - | - | Imbalance of Q-part of complex signal, scaled (-128 = max. negative imbalance, 127 = max. positive imbalance) |
| 3 | U1 | `magQ` | - | - | Magnitude of Q-part of complex signal, scaled (0 = no signal, 255 = max. magnitude) |
| 4 | U1 | `cfgSource` | - | - | Source of low-level configuration<br>(114 = ROM, 111 = OTP, 112 = config pins, 102 = flash image) |
| 5 | U1[3] | `reserved0` | - | - | Reserved |
| 8 | U4 | `lowLevCfg` | - | - | Low-level configuration (obsolete for protocol versions greater than 15.00) |
| 12 | U1[8] | `reserved1` | - | - | Reserved |
| 20 | U4 | `postStatus` | - | - | POST status word |
| 24 | U1[4] | `reserved2` | - | - | Reserved |

## 2.12.2 UBX-MON-VER (0x0a 0x04)

### 2.12.2.1 Poll receiver and software version

| ***Message*** | **UBX-MON-VER**<br>**Poll receiver and software version** |
|---|---|
| *Type* | Poll request |
| *Comment* | |

| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
|---|---|---|---|---|---|---|
| | 0xb5 0x62 | 0x0a | 0x04 | 0 | *see below* | CK_A CK_B |

| | |
|---|---|
| *Payload* | This message has no payload. |

### 2.12.2.2 Receiver and software version

| ***Message*** | **UBX-MON-VER**<br>**Receiver and software version** |
|---|---|
| *Type* | Polled |
| *Comment* | |

| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
|---|---|---|---|---|---|---|
| | 0xb5 0x62 | 0x0a | 0x04 | 40 + [0..n]·30 | *see below* | CK_A CK_B |

| *Payload description:* | | | | | |
|---|---|---|---|---|---|
| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
| 0 | CH[30] | swVersion | - | - | Nul-terminated software version string. |
| 30 | CH[10] | hwVersion | - | - | Nul-terminated hardware version string |
| *Start of repeated group (N times)* | | | | | |
| 40 + n·30 | CH[30] | extension | - | - | Extended software information strings.<br>A series of nul-terminated strings. Each extension field is 30 characters long and contains varying software information. Not all extension fields may appear.<br>Examples of reported information: the software version string of the underlying ROM (when the receiver's firmware is running from flash), the firmware version, the supported protocol version, the module identifier, the flash information structure (FIS) file information, the supported major GNSS, the supported augmentation systems.<br>See Firmware and protocol versions for details. |
| *End of repeated group (N times)* | | | | | |

# 2.13 UBX-RXM (0x02)

The messages in the UBX-RXM class are used to output status and result data from the receiver manager as well as sending commands to the receiver manager.

## 2.13.1 UBX-RXM-PMP (0x02 0x72)

### 2.13.1.1 PMP raw data

| ***Message*** | **UBX-RXM-PMP**<br>**PMP raw data** | | | | | |
|---|---|---|---|---|---|---|
| *Type* | Periodic | | | | | |
| *Comment* | Outputs the raw data when a frame is received. Frame detection algorithm is described in receiver data sheet. If no frame is detected, no message is sent. Periodicity of the message depends on the data rate. The validity of the frame must be verified by host software. For frame verification quality indicators included in this message can be used. | | | | | |
| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
| | 0xb5 0x62 | 0x02 | 0x72 | 528 | *see below* | CK_A CK_B |

| *Payload description:* | | | | | |
|---|---|---|---|---|---|
| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
| 0 | U1 | version | - | - | Message version (0x00 for this version) |
| 1 | U1[3] | reserved0 | - | - | Reserved |
| 4 | U4 | timeTag | - | ms | Time since startup when frame started - if max value of type is reached the counter will be reset |
| 8 | U4[2] | uniqueWord | - | - | Received unique words |
| 16 | U2 | service Identifier | - | - | Received service identifier |
| 18 | U1 | spare | - | - | Received spare data |
| 19 | U1 | uniqueWordBit Errors | - | - | Number of bit errors in both unique words |
| 20 | U1[504] | userData | - | - | Received user data |

| | | | | | |
|---|---|---|---|---|---|
| 524 | U2 | fecBits | - | - | Number of bits corrected by FEC (forward error correction) |
| 526 | U1 | ebno | 2^-3 | dB | Energy per bit to noise power spectral density ratio |
| 527 | U1 | reserved1 | - | - | Reserved |

#### 2.13.1.2 PMP raw data

| ***Message*** | **UBX-RXM-PMP**<br>**PMP raw data** | | | | | |
|---|---|---|---|---|---|---|
| *Type* | Periodic | | | | | |
| *Comment* | Outputs the raw data when a frame is received. Frame detection algorithm is described in receiver data sheet. If no frame is detected, no message is sent. Periodicity of the message depends on the data rate. The validity of the frame must be verified by host software. For frame verification quality indicators included in this message can be used. | | | | | |
| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
| | 0xb5 0x62 | 0x02 | 0x72 | 24 + [0..n] | *see below* | CK_A CK_B |

*Payload description:*

| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
|---|---|---|---|---|---|
| 0 | U1 | version | - | - | Message version (0x01 for this version) |
| 1 | U1 | reserved0 | - | - | Reserved |
| 2 | U2 | numBytesUserData | - | - | Number of bytes the userData block has in this frame (0...504) |
| 4 | U4 | timeTag | - | ms | Time since startup when frame started - if max value of type is reached the counter will be reset |
| 8 | U4[2] | uniqueWord | - | - | Received unique words |
| 16 | U2 | serviceIdentifier | - | - | Received service identifier |
| 18 | U1 | spare | - | - | Received spare data |
| 19 | U1 | uniqueWordBitErrors | - | - | Number of bit errors in both unique words |
| 20 | U2 | fecBits | - | - | Number of bits corrected by FEC (forward error correction) |
| 22 | U1 | ebno | 2^-3 | dB | Energy per bit to noise power spectral density ratio |
| 23 | U1 | reserved1 | - | - | Reserved |
| *Start of repeated group (N times)* | | | | | |
| 24 + n | U1 | userData | - | - | Received user data, which is variable (=numBytesUserData) |
| *End of repeated group (N times)* | | | | | |

### 2.13.2 UBX-RXM-PMREQ (0x02 0x41)

#### 2.13.2.1 Power management request

| ***Message*** | **UBX-RXM-PMREQ**<br>**Power management request** | | | | | |
|---|---|---|---|---|---|---|
| *Type* | Command | | | | | |
| *Comment* | This message requests a power management related task of the receiver. | | | | | |
| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
| | 0xb5 0x62 | 0x02 | 0x41 | 8 | *see below* | CK_A CK_B |

| *Payload description:* | | | | | |
|---|---|---|---|---|---|
| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
| 0 | U4 | duration | - | ms | Duration of the requested task, set to zero for infinite duration. The maximum supported time is 12 days. |
| 4 | X4 | flags | - | - | task flags |
| bit 1 | U:1 | backup | - | - | The receiver goes into backup mode for a time period defined by duration, provided that it is not connected to USB |

### 2.13.2.2 Power management request

| ***Message*** | **UBX-RXM-PMREQ**<br>**Power management request** | | | | | |
|---|---|---|---|---|---|---|
| *Type* | Command | | | | | |
| *Comment* | This message requests a power management related task of the receiver. | | | | | |
| *Message structure* | *Header* | *Class* | *ID* | *Length (Bytes)* | *Payload* | *Checksum* |
| | 0xb5 0x62 | 0x02 | 0x41 | 16 | *see below* | CK_A CK_B |

| *Payload description:* | | | | | |
|---|---|---|---|---|---|
| *Byte offset* | *Type* | *Name* | *Scale* | *Unit* | *Description* |
| 0 | U1 | version | - | - | Message version (0x00 for this version) |
| 1 | U1[3] | reserved0 | - | - | Reserved |
| 4 | U4 | duration | - | ms | Duration of the requested task, set to zero for infinite duration. The maximum supported time is 12 days. |
| 8 | X4 | flags | - | - | task flags |
| bit 1 | U:1 | backup | - | - | The receiver goes into backup mode for a time period defined by duration, provided that it is not connected to USB |
| bit 2 | U:1 | force | - | - | Force receiver backup while USB is connected. USB interface will be disabled. |
| 12 | X4 | wakeupSources | - | - | Configure pins to wake up the receiver. The receiver wakes up if there is either a falling or a rising edge on one of the configured pins. |
| bit 3 | U:1 | uartrx | - | - | Wake up the receiver if there is an edge on the UART RX pin |
| bit 5 | U:1 | extint0 | - | - | Wake up the receiver if there is an edge on the EXTINT0 pin |
| bit 6 | U:1 | extint1 | - | - | Wake up the receiver if there is an edge on the EXTINT1 pin |
| bit 7 | U:1 | spics | - | - | Wake up the receiver if there is an edge on the SPI CS pin |

# 3 Configuration interface

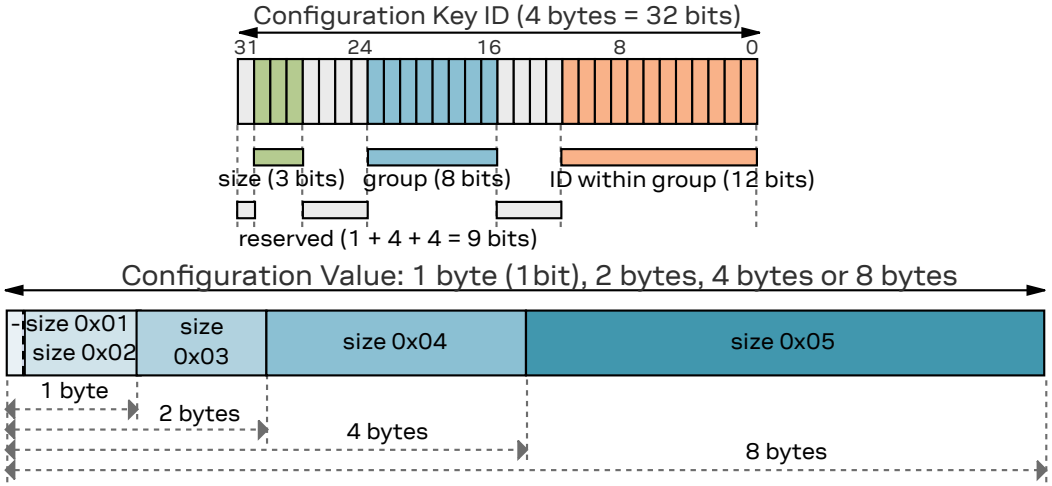
This chapter describes the receiver configuration interface.

## 3.1 Configuration database

The configuration database in the receiver's RAM holds the current configuration, which is used by the receiver at run-time. It is constructed on startup of the receiver from several sources of configuration. These sources are called *Configuration Layers*. The current configuration is called the *RAM Layer*. Any configuration in any layer is organized as *Configuration Items*, where each Configuration Item is referenced to by a unique *Configuration Key ID* and holds a single *Configuration Value*.

## 3.2 Configuration items

The following figure shows the structure of a *Configuration Item*, which consists of a *(Configuration) Key ID* and its *(Configuration) Value*:




A Configuration Key ID is a 32-bit integer value, which is split into the following parts:

- Bit 31: Currently unused. Reserved for future use.
- Bits 30...28: Three bits that indicate the storage size of a Configuration Value (range 0x01-0x05, see below)
- Bits 27...24: Currently unused. Reserved for future use.
- Bits 23...16: Eight bits that define a unique group ID (range 0x01-0xfe)
- Bits 15...12: Currently unused. Reserved for future use.
- Bits 11...0: Twelve bits that define a unique item ID within a group (range 0x001-0xffe)

The entire 32-bit value is the unique Key ID, which uniquely identifies a particular item. The numeric representation of the Key ID uses the lower-case hexadecimal format, such as `0x20c400a1`. An easier, more readable text representation uses the form CFG-*GROUP-ITEM*. This is also referred to as the *(Configuration) Key Name*.

Supported storage size identifiers (bits 30...28 of the Key ID) are:

- 0x01: one bit (the actual storage used is one byte, but only the least significant bit is used)
- 0x02: one byte
- 0x03: two bytes
- 0x04: four bytes

- 0x05: eight bytes

Each Configuration Item is of a certain type, which defines the interpretation of the raw binary data (see also UBX data types):

- U1, U2, U4, U8: unsigned little-endian integers of 8-, 16-, 32- and 64-bit widths
- I1, I2, I4, I8: signed little-endian, two's complement integers of 8-, 16-, 32- and 64-bit widths
- R4, R8: IEEE 754 single (32-bit) and double (64-bit) precision floats
- E1, E2, E4: unsigned little-endian enumeration of 8-, 16-, and 32-bit widths
- X1, X2, X4, X8: unsigned little-endian integers of 8-, 16-, 32- and 64-bit widths for bitfields and other binary data, such as strings
- L: single-bit boolean (true = 1, false = 0), stored as U1

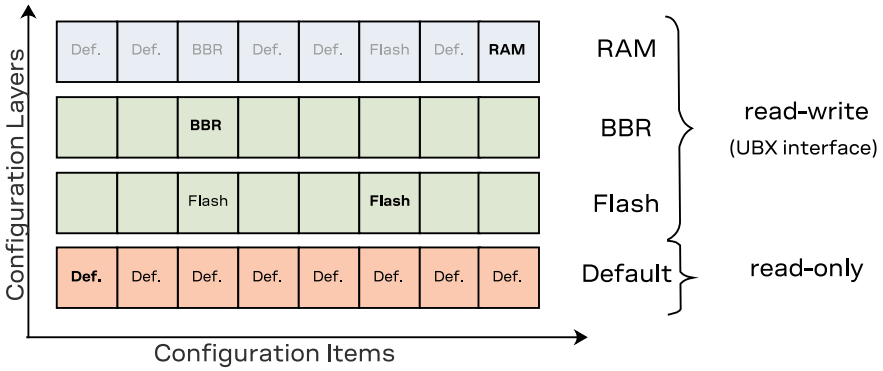
## 3.3 Configuration layers

Several *Configuration Layers* exist. They are separate sources of Configuration Items. Some of the layers are read-only and others are modifiable. Layers are organized in terms of priority. Values in a high-priority layer will replace values stored in low-priority layer. On startup of the receiver all configuration layers are read and the items within each layer are stacked up in order to create the *Current Configuration*, which is used by the receiver at run-time.

The following configuration layers are available (in order of priority, highest priority first):

- **RAM**: This layer contains items stored in volatile RAM. This is the Current Configuration. The value of any item can be set by the user at run-time (see UBX protocol interface) and it will become effective immediately.
- **BBR**: This layer contains items stored in the battery-backed RAM. The contents in this layer are preserved as long as a battery backup supply is provided during off periods. The value of any item can be set by the user at run-time (see UBX protocol interface) and it will become effective upon a restart of the receiver.
- **Flash**: This layer contains items stored permanently in the external flash memory. This layer is only available if there is a usable external flash memory. The value of any item can be set by the user at run-time (see UBX protocol interface) and it will become effective upon a restart of the receiver.
- **Default:** This layer contains all items known to the running receiver software and their hard-coded default values. Data in this layer is not writable.

The stacking of the configuration items from the different layers (sources) in order to construct the Current Configuration in the RAM Layer is depicted in the following figure. For each defined item, i.e. for each item in the Default Layer, the receiver software goes through the layers above and stacks all the found items on top. Some items may not be present in every layer. The result is the RAM Layer filled with all configuration items given Configuration Values coming from the highest priority layer the corresponding item was present. In the example figure below bold text indicates the source of the value in the Current Configuration (the RAM Layer). Empty boxes mean that the layer can hold the item but that it is not currently stored there. Boxes with text mean that an item is currently stored in the layer.




In the example figure above several items (e.g. the first item) are only set in the Default Layer and hence the default value ends up in Current Configuration in the RAM Layer. The third item is present in the Default, Flash and BBR Layers. The value from the BBR Layer has the highest priority and therefore it ends up in the RAM Layer. On the other hand, the default value of the sixth item is changed by the value in the Flash Layer. The value of the last item is changed in the RAM Layer only, i.e. upon startup the value in the RAM Layer was the value from the Default Layer, but the user has changed the value in the RAM Layer at run-time.

## 3.4 Configuration interface access

The following sections describe the existing interfaces to access the Configuration Database.

### 3.4.1 UBX protocol interface

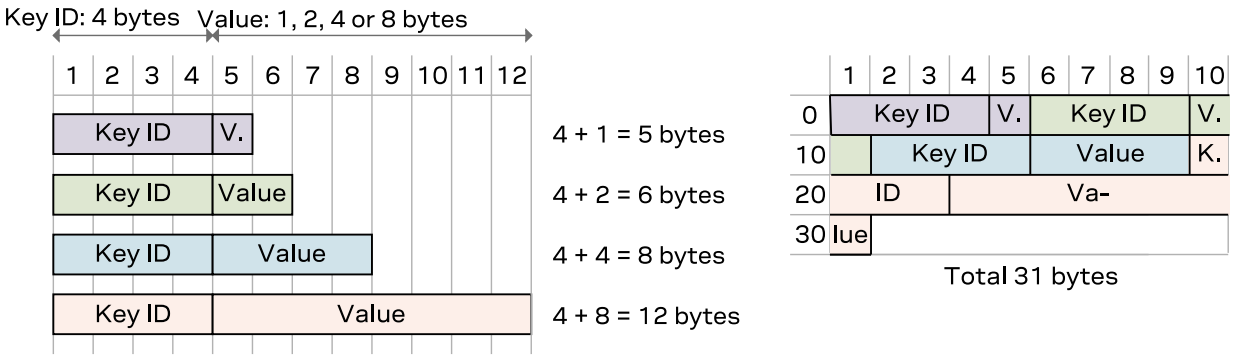
The following UBX protocol messages are available to access the Configuration Database:

- UBX-CFG-VALGET to read configuration items from the database
- UBX-CFG-VALSET to set configuration items in the database
- UBX-CFG-VALDEL to delete configuration items from the database

## 3.5 Configuration data

Configuration data is the binary representation of a list of Key ID and Value pairs. It is formed by concatenating keys (U4 values) and values (variable type) without any padding. This format is used in the UBX-CFG-VALSET and UBX-CFG-VALGET messages.

The figure below shows an example. The four Items (Key ID - Value pairs) on the left use the four fundamental storage sizes: one byte (L, U1, I1, E1 and X1 types), 2 bytes (U2, I2, E2 and X2 types), four byte (U4, I4, E4, X4 and R4 types) and eight bytes (U8, I8, X8 and R8 types). When concatenated (right) the Key IDs and Values are not aligned and there is no padding.



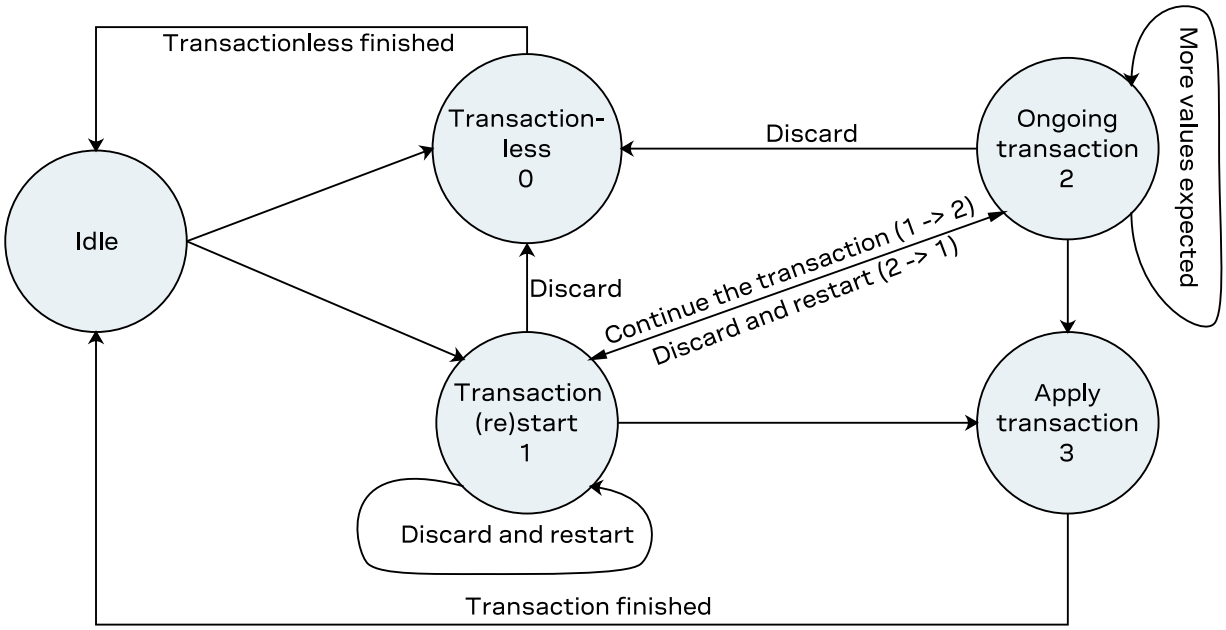
Note that this is an arbitrary example and any number of items of any value storage size can be concatenated the same way.

## 3.6 Configuration transactions

The configuration interface supports two mechanisms of configuration: the first is a transactionless mechanism where sent configuration changes are applied immediately to the configuration layer(s) requested. The second mechanism is a configuration transaction.

A transaction offers a way of queuing multiple configuration changes. It is particularly useful where different configuration keys depend on each other in such a way that sending one before the other can cause the configuration to be rejected. The queued configuration change requests are stored then checked collectively before being applied to the receiver.

A transaction can have the following states described in the figure below.




When starting a transaction, the user must specify the layer(s) the changes will be applied to. This list of configuration layer(s) must be observed throughout the transaction states. Modifying the configuration layer(s) mid-transaction will cause the transaction to be aborted and no queued changes will be applied.

In the start transaction state, the receiver will lock the configuration database so that changes from another entity or message cannot be applied. It is possible to send a configuration key-value pairs with the start transaction state. These will be queued waiting to be applied.

In the ongoing state, a configuration key and value must be sent. The receiver will abort the transaction and not apply any changes if this condition is violated. Key-value pairs sent in the ongoing state will be queued waiting to be applied.

In the apply state, the queued changes will be collectively checked and applied to the requested configuration layer(s). Note that any additional key-value pairs sent within the apply state will be ignored.

Note that a transaction can only come from a single source, a UBX-CFG-VALSET message or a UBX-CFG-VALDEL message. This means that in any given transaction it is not possible to mix a delete

and a save request. Starting a transaction from a different source will abort the current transaction and no queued changes would be applied.

Refer to UBX-CFG-VALSET and UBX-CFG-VALDEL messages for a detailed description of how to set up a configuration transaction, its limitations and conditions that would cause the transaction to be rejected.

## 3.7 Configuration reset behavior

The RAM layer is always rebuilt from the layers below when the chip's processor comes out from reset. When using UBX-CFG-RST the processor goes through a reset cycle with these reset types (`resetMode` field):

- `0x00` hardware reset (watchdog) immediately
- `0x01` controlled software reset
- `0x04` hardware reset (watchdog) after shutdown

See section Forcing a receiver reset in the integration manual.

## 3.8 Configuration overview

| Group | Description |
|---|---|
| CFG-HW | Hardware configuration |
| CFG-I2C | Configuration of the I2C interface |
| CFG-I2CINPROT | Input protocol configuration of the I2C interface |
| CFG-I2COUTPROT | Output protocol configuration of the I2C interface |
| CFG-INFMSG | Information message configuration |
| CFG-ITFM | Jamming and interference monitor configuration |
| CFG-MSGOUT | Message output configuration |
| CFG-PM | Configuration for receiver power management |
| CFG-PMP | Point to multipoint (PMP) configuration |
| CFG-RATE | Navigation and measurement rate configuration |
| CFG-RINV | Remote inventory |
| CFG-SPI | Configuration of the SPI interface |
| CFG-SPIINPROT | Input protocol configuration of the SPI interface |
| CFG-SPIOUTPROT | Output protocol configuration of the SPI interface |
| CFG-TXREADY | TX ready configuration |
| CFG-UART1 | Configuration of the UART1 interface |
| CFG-UART1INPROT | Input protocol configuration of the UART1 interface |
| CFG-UART1OUTPROT | Output protocol configuration of the UART1 interface |
| CFG-UART2 | Configuration of the UART2 interface |
| CFG-UART2INPROT | Input protocol configuration of the UART2 interface |
| CFG-UART2OUTPROT | Output protocol configuration of the UART2 interface |
| CFG-USB | Configuration of the USB interface |
| CFG-USBINPROT | Input protocol configuration of the USB interface |
| CFG-USBOUTPROT | Output protocol configuration of the USB interface |

## 3.9 Configuration reference

### 3.9.1 CFG-HW: Hardware configuration

Hardware configuration settings.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-HW-ANT_CFG_VOLTCTRL* | 0x10a3002e | L | - | - | Active antenna voltage control flag |
| Enable active antenna voltage control flag. Used by EXT and MADC engines. | | | | | |
| *CFG-HW-ANT_CFG_SHORTDET* | 0x10a3002f | L | - | - | Short antenna detection flag |
| Enable short antenna detection flag. Used by EXT and MADC engines. | | | | | |
| *CFG-HW-ANT_CFG_SHORTDET_POL* | 0x10a30030 | L | - | - | Short antenna detection polarity |
| Set to true if polarity of the antenna short detection is active low. Used by EXT engine. | | | | | |
| *CFG-HW-ANT_CFG_OPENDET* | 0x10a30031 | L | - | - | Open antenna detection flag |
| Enable open antenna detection flag. Used by EXT and MADC engines. | | | | | |
| *CFG-HW-ANT_CFG_OPENDET_POL* | 0x10a30032 | L | - | - | Open antenna detection polarity |
| Set to true if polarity of the antenna open detection is active low. Used by EXT engine. | | | | | |
| *CFG-HW-ANT_CFG_PWRDOWN* | 0x10a30033 | L | - | - | Power down antenna flag |
| Enable power down antenna logic in the event of antenna short circuit. CFG-HW-ANT_CFG_SHORTDET must be enabled to use this feature. Used by EXT and MADC engines. | | | | | |
| *CFG-HW-ANT_CFG_PWRDOWN_POL* | 0x10a30034 | L | - | - | Power down antenna logic polarity |
| Set to true if polarity of the antenna power down logic is active high. Used by EXT and MADC engines. | | | | | |
| *CFG-HW-ANT_CFG_RECOVER* | 0x10a30035 | L | - | - | Automatic recovery from short state flag |
| Enable automatic recovery from short state. Used by EXT and MADC engines. | | | | | |
| *CFG-HW-ANT_SUP_SWITCH_PIN* | 0x20a30036 | U1 | - | - | ANT1 PIO number |
| Antenna Switch (ANT1) PIO number. Used by EXT and MADC engines. | | | | | |
| *CFG-HW-ANT_SUP_SHORT_PIN* | 0x20a30037 | U1 | - | - | ANT0 PIO number |
| Antenna Short (ANT0) PIO number. Used by EXT engine. | | | | | |
| *CFG-HW-ANT_SUP_OPEN_PIN* | 0x20a30038 | U1 | - | - | ANT2 PIO number |
| Antenna Switch (ANT2) PIO number. Used by EXT engine. | | | | | |

**Table 1: CFG-HW configuration items**

### 3.9.2 CFG-I2C: Configuration of the I2C interface

Settings needed to configure the I2C communication interface.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-I2C-ADDRESS* | 0x20510001 | U1 | - | - | I2C slave address of the receiver (7 bits) |
| *CFG-I2C-EXTENDEDTIMEOUT* | 0x10510002 | L | - | - | Flag to disable timeouting the interface after 1.5 s |
| *CFG-I2C-ENABLED* | 0x10510003 | L | - | - | Flag to indicate if the I2C interface should be enabled |

**Table 2: CFG-I2C configuration items**

### 3.9.3 CFG-I2CINPROT: Input protocol configuration of the I2C interface

Input protocol enable flags of the I2C interface.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-I2CINPROT-UBX* | 0x10710001 | L | - | - | Flag to indicate if UBX should be an input protocol on I2C |

**Table 3: CFG-I2CINPROT configuration items**

### 3.9.4 CFG-I2COUTPROT: Output protocol configuration of the I2C interface

Output protocol enable flags of the I2C interface.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-I2COUTPROT-UBX* | 0x10720001 | L | - | - | Flag to indicate if UBX should be an output protocol on I2C |

**Table 4: CFG-I2COUTPROT configuration items**

### 3.9.5 CFG-INFMSG: Information message configuration

Information message configuration for the NMEA and UBX protocols.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-INFMSG-UBX_I2C* | 0x20920001 | X1 | - | - | Information message enable flags for the UBX protocol on the I2C interface<br>See Table 6 below for a list of possible constants for this item. |
| *CFG-INFMSG-UBX_UART1* | 0x20920002 | X1 | - | - | Information message enable flags for the UBX protocol on the UART1 interface<br>See Table 6 below for a list of possible constants for this item. |
| *CFG-INFMSG-UBX_UART2* | 0x20920003 | X1 | - | - | Information message enable flags for the UBX protocol on the UART2 interface<br>See Table 6 below for a list of possible constants for this item. |
| *CFG-INFMSG-UBX_USB* | 0x20920004 | X1 | - | - | Information message enable flags for the UBX protocol on the USB interface<br>See Table 6 below for a list of possible constants for this item. |
| *CFG-INFMSG-UBX_SPI* | 0x20920005 | X1 | - | - | Information message enable flags for the UBX protocol on the SPI interface<br>See Table 6 below for a list of possible constants for this item. |
| *CFG-INFMSG-NMEA_I2C* | 0x20920006 | X1 | - | - | Information message enable flags for the NMEA protocol on the I2C interface<br>See Table 6 below for a list of possible constants for this item. |
| *CFG-INFMSG-NMEA_UART1* | 0x20920007 | X1 | - | - | Information message enable flags for the NMEA protocol on the UART1 interface<br>See Table 6 below for a list of possible constants for this item. |
| *CFG-INFMSG-NMEA_UART2* | 0x20920008 | X1 | - | - | Information message enable flags for the NMEA protocol on the UART2 interface<br>See Table 6 below for a list of possible constants for this item. |
| *CFG-INFMSG-NMEA_USB* | 0x20920009 | X1 | - | - | Information message enable flags for the NMEA protocol on the USB interface<br>See Table 6 below for a list of possible constants for this item. |
| *CFG-INFMSG-NMEA_SPI* | 0x2092000a | X1 | - | - | Information message enable flags for the NMEA protocol on the SPI interface<br>See Table 6 below for a list of possible constants for this item. |

**Table 5: CFG-INFMSG configuration items**

| Constant | Value | Description |
|---|---|---|
| *ERROR* | 0x01 | Enable ERROR information messages |

| Constant | Value | Description |
|---|---|---|
| *WARNING* | 0x02 | Enable WARNING information messages |
| *NOTICE* | 0x04 | Enable NOTICE information messages |
| *TEST* | 0x08 | Enable TEST information messages |
| *DEBUG* | 0x10 | Enable DEBUG information messages |

**Table 6: Constants for CFG-INFMSG-UBX_I2C, CFG-INFMSG-UBX_UART1, CFG-INFMSG-UBX_UART2, CFG-INFMSG-UBX_USB, CFG-INFMSG-UBX_SPI, CFG-INFMSG-NMEA_I2C, CFG-INFMSG-NMEA_UART1, CFG-INFMSG-NMEA_UART2, CFG-INFMSG-NMEA_USB, CFG-INFMSG-NMEA_SPI**

### 3.9.6 CFG-ITFM: Jamming and interference monitor configuration

Configuration of jamming and interference monitor.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-ITFM-BBTHRESHOLD* | 0x20410001 | U1 | - | - | Broadband jamming detection threshold |
| *CFG-ITFM-CWTHRESHOLD* | 0x20410002 | U1 | - | - | CW jamming detection threshold |
| *CFG-ITFM-ENABLE* | 0x1041000d | L | - | - | Enable interference detection |
| *CFG-ITFM-ANTSETTING* | 0x20410010 | E1 | - | - | Antenna setting |
| See Table 8 below for a list of possible constants for this item. | | | | | |
| *CFG-ITFM-ENABLE_AUX* | 0x10410013 | L | - | - | Scan auxiliary bands |
| Set to true to scan auxiliary bands.<br>Supported on u-blox 8 / u-blox M8 only, otherwise ignored. | | | | | |

**Table 7: CFG-ITFM configuration items**

| Constant | Value | Description |
|---|---|---|
| *UNKNOWN* | 0 | Unknown |
| *PASSIVE* | 1 | Passive |
| *ACTIVE* | 2 | Active |

**Table 8: Constants for CFG-ITFM-ANTSETTING**

### 3.9.7 CFG-MSGOUT: Message output configuration

For each message and port a separate output rate (per second, per epoch) can be configured.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-MSGOUT-UBX_LOG_INFO_I2C* | 0x20910259 | U1 | - | - | Output rate of the UBX-LOG-INFO message on port I2C |
| *CFG-MSGOUT-UBX_LOG_INFO_SPI* | 0x2091025d | U1 | - | - | Output rate of the UBX-LOG-INFO message on port SPI |
| *CFG-MSGOUT-UBX_LOG_INFO_UART1* | 0x2091025a | U1 | - | - | Output rate of the UBX-LOG-INFO message on port UART1 |
| *CFG-MSGOUT-UBX_LOG_INFO_UART2* | 0x2091025b | U1 | - | - | Output rate of the UBX-LOG-INFO message on port UART2 |
| *CFG-MSGOUT-UBX_LOG_INFO_USB* | 0x2091025c | U1 | - | - | Output rate of the UBX-LOG-INFO message on port USB |
| *CFG-MSGOUT-UBX_MON_HW2_I2C* | 0x209101b9 | U1 | - | - | Output rate of the UBX-MON-HW2 message on port I2C |
| *CFG-MSGOUT-UBX_MON_HW2_SPI* | 0x209101bd | U1 | - | - | Output rate of the UBX-MON-HW2 message on port SPI |
| *CFG-MSGOUT-UBX_MON_HW2_UART1* | 0x209101ba | U1 | - | - | Output rate of the UBX-MON-HW2 message on port UART1 |

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-MSGOUT-UBX_MON_HW2_UART2* | 0x209101bb | U1 | - | - | Output rate of the UBX-MON-HW2 message on port UART2 |
| *CFG-MSGOUT-UBX_MON_HW2_USB* | 0x209101bc | U1 | - | - | Output rate of the UBX-MON-HW2 message on port USB |
| *CFG-MSGOUT-UBX_MON_HW_I2C* | 0x209101b4 | U1 | - | - | Output rate of the UBX-MON-HW message on port I2C |
| *CFG-MSGOUT-UBX_MON_HW_SPI* | 0x209101b8 | U1 | - | - | Output rate of the UBX-MON-HW message on port SPI |
| *CFG-MSGOUT-UBX_MON_HW_UART1* | 0x209101b5 | U1 | - | - | Output rate of the UBX-MON-HW message on port UART1 |
| *CFG-MSGOUT-UBX_MON_HW_UART2* | 0x209101b6 | U1 | - | - | Output rate of the UBX-MON-HW message on port UART2 |
| *CFG-MSGOUT-UBX_MON_HW_USB* | 0x209101b7 | U1 | - | - | Output rate of the UBX-MON-HW message on port USB |
| *CFG-MSGOUT-UBX_MON_IO_I2C* | 0x209101a5 | U1 | - | - | Output rate of the UBX-MON-IO message on port I2C |
| *CFG-MSGOUT-UBX_MON_IO_SPI* | 0x209101a9 | U1 | - | - | Output rate of the UBX-MON-IO message on port SPI |
| *CFG-MSGOUT-UBX_MON_IO_UART1* | 0x209101a6 | U1 | - | - | Output rate of the UBX-MON-IO message on port UART1 |
| *CFG-MSGOUT-UBX_MON_IO_UART2* | 0x209101a7 | U1 | - | - | Output rate of the UBX-MON-IO message on port UART2 |
| *CFG-MSGOUT-UBX_MON_IO_USB* | 0x209101a8 | U1 | - | - | Output rate of the UBX-MON-IO message on port USB |
| *CFG-MSGOUT-UBX_MON_MSGPP_I2C* | 0x20910196 | U1 | - | - | Output rate of the UBX-MON-MSGPP message on port I2C |
| *CFG-MSGOUT-UBX_MON_MSGPP_SPI* | 0x2091019a | U1 | - | - | Output rate of the UBX-MON-MSGPP message on port SPI |
| *CFG-MSGOUT-UBX_MON_MSGPP_UART1* | 0x20910197 | U1 | - | - | Output rate of the UBX-MON-MSGPP message on port UART1 |
| *CFG-MSGOUT-UBX_MON_MSGPP_UART2* | 0x20910198 | U1 | - | - | Output rate of the UBX-MON-MSGPP message on port UART2 |
| *CFG-MSGOUT-UBX_MON_MSGPP_USB* | 0x20910199 | U1 | - | - | Output rate of the UBX-MON-MSGPP message on port USB |
| *CFG-MSGOUT-UBX_MON_RXBUF_I2C* | 0x209101a0 | U1 | - | - | Output rate of the UBX-MON-RXBUF message on port I2C |
| *CFG-MSGOUT-UBX_MON_RXBUF_SPI* | 0x209101a4 | U1 | - | - | Output rate of the UBX-MON-RXBUF message on port SPI |
| *CFG-MSGOUT-UBX_MON_RXBUF_UART1* | 0x209101a1 | U1 | - | - | Output rate of the UBX-MON-RXBUF message on port UART1 |
| *CFG-MSGOUT-UBX_MON_RXBUF_UART2* | 0x209101a2 | U1 | - | - | Output rate of the UBX-MON-RXBUF message on port UART2 |
| *CFG-MSGOUT-UBX_MON_RXBUF_USB* | 0x209101a3 | U1 | - | - | Output rate of the UBX-MON-RXBUF message on port USB |
| *CFG-MSGOUT-UBX_MON_RXR_I2C* | 0x20910187 | U1 | - | - | Output rate of the UBX-MON-RXR message on port I2C |
| *CFG-MSGOUT-UBX_MON_RXR_SPI* | 0x2091018b | U1 | - | - | Output rate of the UBX-MON-RXR message on port SPI |
| *CFG-MSGOUT-UBX_MON_RXR_UART1* | 0x20910188 | U1 | - | - | Output rate of the UBX-MON-RXR message on port UART1 |
| *CFG-MSGOUT-UBX_MON_RXR_UART2* | 0x20910189 | U1 | - | - | Output rate of the UBX-MON-RXR message on port UART2 |

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-MSGOUT-UBX_MON_RXR_USB* | 0x2091018a | U1 | - | - | Output rate of the UBX-MON-RXR message on port USB |
| *CFG-MSGOUT-UBX_MON_TXBUF_I2C* | 0x2091019b | U1 | - | - | Output rate of the UBX-MON-TXBUF message on port I2C |
| *CFG-MSGOUT-UBX_MON_TXBUF_SPI* | 0x2091019f | U1 | - | - | Output rate of the UBX-MON-TXBUF message on port SPI |
| *CFG-MSGOUT-UBX_MON_TXBUF_UART1* | 0x2091019c | U1 | - | - | Output rate of the UBX-MON-TXBUF message on port UART1 |
| *CFG-MSGOUT-UBX_MON_TXBUF_UART2* | 0x2091019d | U1 | - | - | Output rate of the UBX-MON-TXBUF message on port UART2 |
| *CFG-MSGOUT-UBX_MON_TXBUF_USB* | 0x2091019e | U1 | - | - | Output rate of the UBX-MON-TXBUF message on port USB |
| *CFG-MSGOUT-UBX_RXM_PMP_I2C* | 0x2091031d | U1 | - | - | Output rate of the UBX_RXM_PMP message on port I2C |
| *CFG-MSGOUT-UBX_RXM_PMP_SPI* | 0x20910321 | U1 | - | - | Output rate of the UBX_RXM_PMP message on port SPI |
| *CFG-MSGOUT-UBX_RXM_PMP_UART1* | 0x2091031e | U1 | - | - | Output rate of the UBX_RXM_PMP message on port UART1 |
| *CFG-MSGOUT-UBX_RXM_PMP_UART2* | 0x2091031f | U1 | - | - | Output rate of the UBX_RXM_PMP message on port UART2 |
| *CFG-MSGOUT-UBX_RXM_PMP_USB* | 0x20910320 | U1 | - | - | Output rate of the UBX_RXM_PMP message on port USB |
| *CFG-MSGOUT-UBX_RXM_SFRBX_I2C* | 0x20910231 | U1 | - | - | Output rate of the UBX-RXM-SFRBX message on port I2C |
| *CFG-MSGOUT-UBX_RXM_SFRBX_SPI* | 0x20910235 | U1 | - | - | Output rate of the UBX-RXM-SFRBX message on port SPI |
| *CFG-MSGOUT-UBX_RXM_SFRBX_UART1* | 0x20910232 | U1 | - | - | Output rate of the UBX-RXM-SFRBX message on port UART1 |
| *CFG-MSGOUT-UBX_RXM_SFRBX_UART2* | 0x20910233 | U1 | - | - | Output rate of the UBX-RXM-SFRBX message on port UART2 |
| *CFG-MSGOUT-UBX_RXM_SFRBX_USB* | 0x20910234 | U1 | - | - | Output rate of the UBX-RXM-SFRBX message on port USB |

**Table 9: CFG-MSGOUT configuration items**

### 3.9.8 CFG-PM: Configuration for receiver power management

Use this configuration group to manage the two main receiver power save modes (on/off operation, PSMOO or cyclic tracking operation, PSMCT).

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-PM-EXTINTSEL* | 0x20d0000b | E1 | - | - | EXTINT pin select |
| See Table 12 below for a list of possible constants for this item. | | | | | |
| *CFG-PM-EXTINTWAKE* | 0x10d0000c | L | - | - | EXTINT pin control (Wake) |
| Enable to keep receiver awake as long as selected EXTINT pin is "high". | | | | | |
| *CFG-PM-EXTINTBACKUP* | 0x10d0000d | L | - | - | EXTINT pin control (Backup) |
| Enable to force receiver into BACKUP mode when selected EXTINT pin is "low". | | | | | |
| *CFG-PM-EXTINTINACTIVE* | 0x10d0000e | L | - | - | EXTINT pin control (Inactive) |
| Enable to force backup in case EXTINT pin is inactive for time longer than CFG-PM-EXTINTINACTIVITY. | | | | | |
| *CFG-PM-EXTINTINACTIVITY* | 0x40d0000f | U4 | 0.001 | s | Inactivity time out on EXTINT pin if enabled |

**Table 10: CFG-PM configuration items**

| Constant | Value | Description |
|---|---|---|
| *FULL* | 0 | normal operation, no power save mode active |
| *PSMOO* | 1 | PSM ON/OFF operation |
| *PSMCT* | 2 | PSM cyclic tracking operation |

**Table 11: Constants for CFG-PM-OPERATEMODE**

| Constant | Value | Description |
|---|---|---|
| *EXTINT0* | 0 | EXTINT0 Pin |
| *EXTINT1* | 1 | EXTINT1 Pin |

**Table 12: Constants for CFG-PM-EXTINTSEL**

### 3.9.9 CFG-PMP: Point to multipoint (PMP) configuration

This is the configuration for the L-band point to multipoint (PMP) receiver.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-PMP-CENTER_FREQUENCY* | 0x40b10011 | U4 | - | Hz | Center frequency |
| The center frequency for the receiver can be set from 1525000000 to 1559000000 Hz. | | | | | |
| *CFG-PMP-SEARCH_WINDOW* | 0x30b10012 | U2 | - | Hz | Search window |
| Search window can be set from 0 to 65535 Hz. It is +/- this value from the center frequency set by CENTER_FREQUENCY. | | | | | |
| *CFG-PMP-USE_SERVICE_ID* | 0x10b10016 | L | - | - | Use service ID |
| Enable/disable service ID check to confirm the correct service is received. | | | | | |
| *CFG-PMP-SERVICE_ID* | 0x30b10017 | U2 | - | - | Service identifier |
| Defines the expected service ID. | | | | | |
| *CFG-PMP-DATA_RATE* | 0x30b10013 | E2 | - | bps | Data rate |
| The data rate of the received data.<br>See Table 14 below for a list of possible constants for this item. | | | | | |
| *CFG-PMP-USE_DESCRAMBLER* | 0x10b10014 | L | - | - | Use descrambler |
| Enables/disables the descrambler. | | | | | |
| *CFG-PMP-DESCRAMBLER_INIT* | 0x30b10015 | U2 | - | - | Descrambler initialization |
| Set the intialisation value for the descrambler. | | | | | |
| *CFG-PMP-USE_PRESCRAMBLING* | 0x10b10019 | L | - | - | Use prescrambling |
| Enables/disables the prescrambling. | | | | | |
| *CFG-PMP-UNIQUE_WORD* | 0x50b1001a | U8 | - | - | Unique word |
| Defines value of unique word. | | | | | |

**Table 13: CFG-PMP configuration items**

| Constant | Value | Description |
|---|---|---|
| *B600* | 600 | 600 bps |
| *B1200* | 1200 | 1200 bps |
| *B2400* | 2400 | 2400 bps |
| *B4800* | 4800 | 4800 bps |

**Table 14: Constants for CFG-PMP-DATA_RATE**

### 3.9.10 CFG-RATE: Navigation and measurement rate configuration

The configuration items in this group allow the user to alter the rate at which navigation solutions (and the measurements that they depend on) are generated by the receiver. The calculation of the

navigation solution will always be aligned to the top of a second zero (first second of the week) of the configured reference time system. The navigation period is an integer multiple of the measurement period.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-RATE-MEAS* | 0x30210001 | U2 | 0.001 | s | Nominal time between GNSS measurements |
| E.g. 100 ms results in 10 Hz measurement rate, 1000 ms = 1 Hz measurement rate. The minimum value is 25. | | | | | |
| *CFG-RATE-NAV* | 0x30210002 | U2 | - | - | Ratio of number of measurements to number of navigation solutions |
| E.g. 5 means five measurements for every navigation solution. The minimum value is 1. The maximum value is 127. | | | | | |
| *CFG-RATE-TIMEREF* | 0x20210003 | E1 | - | - | Time system to which measurements are aligned |
| See Table 16 below for a list of possible constants for this item. | | | | | |

**Table 15: CFG-RATE configuration items**

| Constant | Value | Description |
|---|---|---|
| *UTC* | 0 | Align measurements to UTC time |
| *GPS* | 1 | Align measurements to GPS time |
| *GLO* | 2 | Align measurements to GLONASS time |
| *BDS* | 3 | Align measurements to BeiDou time |
| *GAL* | 4 | Align measurements to Galileo time |
| *NAVIC* | 5 | Align measurements to NavIC time |

**Table 16: Constants for CFG-RATE-TIMEREF**

## 3.9.11 CFG-RINV: Remote inventory

The remote inventory enables storing user-defined data in the non-volatile memory of the receiver. The data can be either binary or a string of ASCII characters. In the latter case, it can optionally be output at startup after the boot screen.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-RINV-DUMP* | 0x10c70001 | L | - | - | Dump data at startup |
| When true, data will be dumped to the interface on startup, unless CFG-RINV-BINARY is set. | | | | | |
| *CFG-RINV-BINARY* | 0x10c70002 | L | - | - | Data is binary |
| When true, the data is treated as binary data. | | | | | |
| *CFG-RINV-DATA_SIZE* | 0x20c70003 | U1 | - | - | Size of data |
| Size of data to store/be stored in the remote inventory (maximum 30 bytes). | | | | | |
| *CFG-RINV-CHUNK0* | 0x50c70004 | X8 | - | - | Data bytes 1-8 (LSB) |
| Data to store/be stored in remote inventory - max 8 bytes, left-most in LSB, e.g. string ABCD will appear as 0x44434241. | | | | | |
| *CFG-RINV-CHUNK1* | 0x50c70005 | X8 | - | - | Data bytes 9-16 |
| Data to store/be stored in remote inventory - max 8 bytes, left-most in LSB, e.g. string ABCD will appear as 0x44434241. | | | | | |
| *CFG-RINV-CHUNK2* | 0x50c70006 | X8 | - | - | Data bytes 17-24 |
| Data to store/be stored in remote inventory - max 8 bytes, left-most in LSB, e.g. string ABCD will appear as 0x44434241. | | | | | |
| *CFG-RINV-CHUNK3* | 0x50c70007 | X8 | - | - | Data bytes 25-30 (MSB) |

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| Data to store/be stored in remote inventory - max 6 bytes, left-most in LSB, e.g. string ABCD will appear as 0x44434241. | | | | | |

**Table 17: CFG-RINV configuration items**

## 3.9.12 CFG-SPI: Configuration of the SPI interface

Settings needed to configure the SPI communication interface.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-SPI-MAXFF* | 0x20640001 | U1 | - | - | Number of bytes containing 0xFF to receive before switching off reception. Range: 0 (mechanism off) - 63 |
| *CFG-SPI-CPOLARITY* | 0x10640002 | L | - | - | Clock polarity select: 0: Active Hight Clock, SCLK idles low, 1: Active Low Clock, SCLK idles high |
| *CFG-SPI-CPHASE* | 0x10640003 | L | - | - | Clock phase select: 0: Data captured on first edge of SCLK, 1: Data captured on second edge of SCLK |
| *CFG-SPI-EXTENDEDTIMEOUT* | 0x10640005 | L | - | - | Flag to disable timeouting the interface after 1.5s |
| *CFG-SPI-ENABLED* | 0x10640006 | L | - | - | Flag to indicate if the SPI interface should be enabled |

**Table 18: CFG-SPI configuration items**

## 3.9.13 CFG-SPIINPROT: Input protocol configuration of the SPI interface

Input protocol enable flags of the SPI interface.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-SPIINPROT-UBX* | 0x10790001 | L | - | - | Flag to indicate if UBX should be an input protocol on SPI |

**Table 19: CFG-SPIINPROT configuration items**

## 3.9.14 CFG-SPIOUTPROT: Output protocol configuration of the SPI interface

Output protocol enable flags of the SPI interface.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-SPIOUTPROT-UBX* | 0x107a0001 | L | - | - | Flag to indicate if UBX should be an output protocol on SPI |

**Table 20: CFG-SPIOUTPROT configuration items**

## 3.9.15 CFG-TXREADY: TX ready configuration

Configuration of the TX ready pin.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-TXREADY-ENABLED* | 0x10a20001 | L | - | - | Flag to indicate if TX ready pin mechanism should be enabled |
| *CFG-TXREADY-POLARITY* | 0x10a20002 | L | - | - | The polarity of the TX ready pin: false:high-active, true:low-active |
| *CFG-TXREADY-PIN* | 0x20a20003 | U1 | - | - | Pin number to use for the TX ready functionality |
| *CFG-TXREADY-THRESHOLD* | 0x30a20004 | U2 | - | - | Amount of data that should be ready on the interface before triggering the TX ready pin |
| *CFG-TXREADY-INTERFACE* | 0x20a20005 | E1 | - | - | Interface where the TX ready feature should be linked to |

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| See Table 22 below for a list of possible constants for this item. | | | | | |

**Table 21: CFG-TXREADY configuration items**

| Constant | Value | Description |
|---|---|---|
| *I2C* | 0 | I2C interface |
| *SPI* | 1 | SPI interface |

**Table 22: Constants for CFG-TXREADY-INTERFACE**

## 3.9.16 CFG-UART1: Configuration of the UART1 interface

Settings needed to configure the UART1 communication interface.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-UART1-BAUDRATE* | 0x40520001 | U4 | - | - | The baud rate that should be configured on the UART1 |
| *CFG-UART1-STOPBITS* | 0x20520002 | E1 | - | - | Number of stopbits that should be used on UART1 |
| See Table 24 below for a list of possible constants for this item. | | | | | |
| *CFG-UART1-DATABITS* | 0x20520003 | E1 | - | - | Number of databits that should be used on UART1 |
| See Table 25 below for a list of possible constants for this item. | | | | | |
| *CFG-UART1-PARITY* | 0x20520004 | E1 | - | - | Parity mode that should be used on UART1 |
| See Table 26 below for a list of possible constants for this item. | | | | | |
| *CFG-UART1-ENABLED* | 0x10520005 | L | - | - | Flag to indicate if the UART1 should be enabled |

**Table 23: CFG-UART1 configuration items**

| Constant | Value | Description |
|---|---|---|
| *HALF* | 0 | 0.5 stopbits |
| *ONE* | 1 | 1.0 stopbits |
| *ONEHALF* | 2 | 1.5 stopbits |
| *TWO* | 3 | 2.0 stopbits |

**Table 24: Constants for CFG-UART1-STOPBITS**

| Constant | Value | Description |
|---|---|---|
| *EIGHT* | 0 | 8 databits |
| *SEVEN* | 1 | 7 databits |

**Table 25: Constants for CFG-UART1-DATABITS**

| Constant | Value | Description |
|---|---|---|
| *NONE* | 0 | No parity bit |
| *ODD* | 1 | Add an odd parity bit |
| *EVEN* | 2 | Add an even parity bit |

**Table 26: Constants for CFG-UART1-PARITY**

## 3.9.17 CFG-UART1INPROT: Input protocol configuration of the UART1 interface

Input protocol enable flags of the UART1 interface.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-UART1INPROT-UBX* | 0x10730001 | L | - | - | Flag to indicate if UBX should be an input protocol on UART1 |

**Table 27: CFG-UART1INPROT configuration items**

### 3.9.18 CFG-UART1OUTPROT: Output protocol configuration of the UART1 interface

Output protocol enable flags of the UART1 interface.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-UART1OUTPROT-UBX* | 0x10740001 | L | - | - | Flag to indicate if UBX should be an output protocol on UART1 |

**Table 28: CFG-UART1OUTPROT configuration items**

### 3.9.19 CFG-UART2: Configuration of the UART2 interface

Settings needed to configure the UART2 communication interface.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-UART2-BAUDRATE* | 0x40530001 | U4 | - | - | The baud rate that should be configured on the UART2 |
| *CFG-UART2-STOPBITS* | 0x20530002 | E1 | - | - | Number of stopbits that should be used on UART2<br>See Table 30 below for a list of possible constants for this item. |
| *CFG-UART2-DATABITS* | 0x20530003 | E1 | - | - | Number of databits that should be used on UART2<br>See Table 31 below for a list of possible constants for this item. |
| *CFG-UART2-PARITY* | 0x20530004 | E1 | - | - | Parity mode that should be used on UART2<br>See Table 32 below for a list of possible constants for this item. |
| *CFG-UART2-ENABLED* | 0x10530005 | L | - | - | Flag to indicate if the UART2 should be enabled |
| *CFG-UART2-REMAP* | 0x10530006 | L | - | - | UART2 Remapping |

**Table 29: CFG-UART2 configuration items**

| Constant | Value | Description |
|---|---|---|
| *HALF* | 0 | 0.5 stopbits |
| *ONE* | 1 | 1.0 stopbits |
| *ONEHALF* | 2 | 1.5 stopbits |
| *TWO* | 3 | 2.0 stopbits |

**Table 30: Constants for CFG-UART2-STOPBITS**

| Constant | Value | Description |
|---|---|---|
| *EIGHT* | 0 | 8 databits |
| *SEVEN* | 1 | 7 databits |

**Table 31: Constants for CFG-UART2-DATABITS**

| Constant | Value | Description |
|---|---|---|
| *NONE* | 0 | No parity bit |
| *ODD* | 1 | Add an odd parity bit |

| Constant | Value | Description |
|---|---|---|
| *EVEN* | 2 | Add an even parity bit |

**Table 32: Constants for CFG-UART2-PARITY**

## 3.9.20 CFG-UART2INPROT: Input protocol configuration of the UART2 interface

Input protocol enable flags of the UART2 interface.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-UART2INPROT-UBX* | 0x10750001 | L | - | - | Flag to indicate if UBX should be an input protocol on UART2 |

**Table 33: CFG-UART2INPROT configuration items**

## 3.9.21 CFG-UART2OUTPROT: Output protocol configuration of the UART2 interface

Output protocol enable flags of the UART2 interface.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-UART2OUTPROT-UBX* | 0x10760001 | L | - | - | Flag to indicate if UBX should be an output protocol on UART2 |

**Table 34: CFG-UART2OUTPROT configuration items**

## 3.9.22 CFG-USB: Configuration of the USB interface

Settings needed to configure the USB communication interface.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-USB-ENABLED* | 0x10650001 | L | - | - | Flag to indicate if the USB interface should be enabled |
| *CFG-USB-SELFPOW* | 0x10650002 | L | - | - | Self-powered device |
| *CFG-USB-VENDOR_ID* | 0x3065000a | U2 | - | - | Vendor ID |
| *CFG-USB-PRODUCT_ID* | 0x3065000b | U2 | - | - | Vendor ID |
| *CFG-USB-POWER* | 0x3065000c | U2 | - | mA | Power consumption |
| *CFG-USB-VENDOR_STR0* | 0x5065000d | X8 | - | - | Vendor string characters 0-7 |
| *CFG-USB-VENDOR_STR1* | 0x5065000e | X8 | - | - | Vendor string characters 8-15 |
| *CFG-USB-VENDOR_STR2* | 0x5065000f | X8 | - | - | Vendor string characters 16-23 |
| *CFG-USB-VENDOR_STR3* | 0x50650010 | X8 | - | - | Vendor string characters 24-31 |
| *CFG-USB-PRODUCT_STR0* | 0x50650011 | X8 | - | - | Product string characters 0-7 |
| *CFG-USB-PRODUCT_STR1* | 0x50650012 | X8 | - | - | Product string characters 8-15 |
| *CFG-USB-PRODUCT_STR2* | 0x50650013 | X8 | - | - | Product string characters 16-23 |
| *CFG-USB-PRODUCT_STR3* | 0x50650014 | X8 | - | - | Product string characters 24-31 |
| *CFG-USB-SERIAL_NO_STR0* | 0x50650015 | X8 | - | - | Serial number string characters 0-7 |
| *CFG-USB-SERIAL_NO_STR1* | 0x50650016 | X8 | - | - | Serial number string characters 8-15 |
| *CFG-USB-SERIAL_NO_STR2* | 0x50650017 | X8 | - | - | Serial number string characters 16-23 |
| *CFG-USB-SERIAL_NO_STR3* | 0x50650018 | X8 | - | - | Serial number string characters 24-31 |

**Table 35: CFG-USB configuration items**

## 3.9.23 CFG-USBINPROT: Input protocol configuration of the USB interface

Input protocol enable flags of the USB interface.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-USBINPROT-UBX* | 0x10770001 | L | - | - | Flag to indicate if UBX should be an input protocol on USB |

**Table 36: CFG-USBINPROT configuration items**

### 3.9.24 CFG-USBOUTPROT: Output protocol configuration of the USB interface

Output protocol enable flags of the USB interface.

| Configuration item | Key ID | Type | Scale | Unit | Description |
|---|---|---|---|---|---|
| *CFG-USBOUTPROT-UBX* | 0x10780001 | L | - | - | Flag to indicate if UBX should be an output protocol on USB |

**Table 37: CFG-USBOUTPROT configuration items**

## 3.10 Legacy UBX message fields reference

The following table lists the legacy UBX message fields and the corresponding configuration item. Note that the mapping from UBX-CFG message fields to configuration items is not necessarily 1:1 and that that some legacy UBX-CFG messages may not be available for certain products.

| UBX message and field | Configuration item(s) |
|---|---|
| **UBX-CFG-ANT** | |
| UBX-CFG-ANT.ocd | CFG-HW-ANT_CFG_OPENDET |
| UBX-CFG-ANT.pdwnOnSCD | CFG-HW-ANT_CFG_PWRDOWN |
| UBX-CFG-ANT.pinOCD | CFG-HW-ANT_SUP_OPEN_PIN |
| UBX-CFG-ANT.pinSCD | CFG-HW-ANT_SUP_SHORT_PIN |
| UBX-CFG-ANT.pinSwitch | CFG-HW-ANT_SUP_SWITCH_PIN |
| UBX-CFG-ANT.recovery | CFG-HW-ANT_CFG_RECOVER |
| UBX-CFG-ANT.scd | CFG-HW-ANT_CFG_SHORTDET |
| UBX-CFG-ANT.svcs | CFG-HW-ANT_CFG_VOLTCTRL |
| **UBX-CFG-INF** | |
| UBX-CFG-INF.infMsgMask | CFG-INFMSG-UBX_I2C, CFG-INFMSG-UBX_UART1, CFG-INFMSG-UBX_UART2, CFG-INFMSG-UBX_USB, CFG-INFMSG-UBX_SPI, CFG-INFMSG-NMEA_I2C, CFG-INFMSG-NMEA_UART1, CFG-INFMSG-NMEA_UART2, CFG-INFMSG-NMEA_USB, CFG-INFMSG-NMEA_SPI |
| UBX-CFG-INF.protocolID | CFG-INFMSG-UBX_UART1, CFG-INFMSG-UBX_UART2, CFG-INFMSG-UBX_USB, CFG-INFMSG-UBX_SPI, CFG-INFMSG-NMEA_I2C, CFG-INFMSG-NMEA_UART1, CFG-INFMSG-NMEA_UART2, CFG-INFMSG-NMEA_USB, CFG-INFMSG-NMEA_SPI |
| **UBX-CFG-ITFM** | |
| UBX-CFG-ITFM.antSetting | CFG-ITFM-ANTSETTING |
| UBX-CFG-ITFM.bbThreshold | CFG-ITFM-BBTHRESHOLD |
| UBX-CFG-ITFM.cwThreshold | CFG-ITFM-CWTHRESHOLD |
| UBX-CFG-ITFM.enable | CFG-ITFM-ENABLE |
| UBX-CFG-ITFM.enable2 | CFG-ITFM-ENABLE_AUX |
| **UBX-CFG-PM2** | |
| UBX-CFG-PM2.extintBackup | CFG-PM-EXTINTBACKUP |
| UBX-CFG-PM2.extintInactive | CFG-PM-EXTINTINACTIVE |
| UBX-CFG-PM2.extintInactivityMs | CFG-PM-EXTINTINACTIVITY |
| UBX-CFG-PM2.extintSel | CFG-PM-EXTINTSEL |

| UBX message and field | Configuration item(s) |
|---|---|
| UBX-CFG-PM2.extintWake | CFG-PM-EXTINTWAKE |
| **UBX-CFG-PRT** | |
| UBX-CFG-PRT.en | CFG-TXREADY-ENABLED |
| UBX-CFG-PRT.extendedTxTimeout | CFG-I2C-EXTENDEDTIMEOUT |
| UBX-CFG-PRT.inProtoMask | CFG-I2C-ENABLED |
| UBX-CFG-PRT.inUbx | CFG-I2CINPROT-UBX |
| UBX-CFG-PRT.outProtoMask | CFG-I2C-ENABLED |
| UBX-CFG-PRT.outUbx | CFG-I2COUTPROT-UBX |
| UBX-CFG-PRT.pin | CFG-TXREADY-PIN |
| UBX-CFG-PRT.pol | CFG-TXREADY-POLARITY |
| UBX-CFG-PRT.slaveAddr | CFG-I2C-ADDRESS |
| UBX-CFG-PRT.thres | CFG-TXREADY-THRESHOLD |
| UBX-CFG-PRT.en | CFG-TXREADY-ENABLED |
| UBX-CFG-PRT.extendedTxTimeout | CFG-SPI-EXTENDEDTIMEOUT |
| UBX-CFG-PRT.ffCnt | CFG-SPI-MAXFF |
| UBX-CFG-PRT.inProtoMask | CFG-SPI-ENABLED |
| UBX-CFG-PRT.inUbx | CFG-SPIINPROT-UBX |
| UBX-CFG-PRT.outProtoMask | CFG-SPI-ENABLED |
| UBX-CFG-PRT.outUbx | CFG-SPIOUTPROT-UBX |
| UBX-CFG-PRT.pin | CFG-TXREADY-PIN |
| UBX-CFG-PRT.pol | CFG-TXREADY-POLARITY |
| UBX-CFG-PRT.spiMode | CFG-SPI-CPOLARITY, CFG-SPI-CPHASE |
| UBX-CFG-PRT.thres | CFG-TXREADY-THRESHOLD |
| UBX-CFG-PRT.baudRate | CFG-UART1-BAUDRATE, CFG-UART2-BAUDRATE |
| UBX-CFG-PRT.charLen | CFG-UART1-DATABITS, CFG-UART2-DATABITS |
| UBX-CFG-PRT.inProtoMask | CFG-UART1-ENABLED, CFG-UART2-ENABLED |
| UBX-CFG-PRT.inUbx | CFG-UART1INPROT-UBX, CFG-UART2INPROT-UBX |
| UBX-CFG-PRT.nStopBits | CFG-UART1-STOPBITS, CFG-UART2-STOPBITS |
| UBX-CFG-PRT.outProtoMask | CFG-UART1-ENABLED, CFG-UART2-ENABLED |
| UBX-CFG-PRT.outUbx | CFG-UART1OUTPROT-UBX, CFG-UART2OUTPROT-UBX |
| UBX-CFG-PRT.parity | CFG-UART1-PARITY, CFG-UART2-PARITY |
| UBX-CFG-PRT.inProtoMask | CFG-USB-ENABLED |
| UBX-CFG-PRT.inUbx | CFG-USBINPROT-UBX |
| UBX-CFG-PRT.outProtoMask | CFG-USB-ENABLED |
| UBX-CFG-PRT.outUbx | CFG-USBOUTPROT-UBX |
| **UBX-CFG-RATE** | |
| UBX-CFG-RATE.measRate | CFG-RATE-MEAS |
| UBX-CFG-RATE.navRate | CFG-RATE-NAV |
| UBX-CFG-RATE.timeRef | CFG-RATE-TIMEREF |
| **UBX-CFG-RINV** | |
| UBX-CFG-RINV.data | CFG-RINV-DATA_SIZE, CFG-RINV-CHUNK0, CFG-RINV-CHUNK1, CFG-RINV-CHUNK2, CFG-RINV-CHUNK3 |
| UBX-CFG-RINV.flags | CFG-RINV-DUMP, CFG-RINV-BINARY |
| **UBX-CFG-USB** | |

| UBX message and field | Configuration item(s) |
|---|---|
| UBX-CFG-USB.powerConsumption | CFG-USB-POWER |
| UBX-CFG-USB.powerMode | CFG-USB-SELFPOW |
| UBX-CFG-USB.productID | CFG-USB-PRODUCT_ID |
| UBX-CFG-USB.productString | CFG-USB-PRODUCT_STR0, CFG-USB-PRODUCT_STR1, CFG-USB-PRODUCT_STR2, CFG-USB-PRODUCT_STR3 |
| UBX-CFG-USB.serialNumber | CFG-USB-SERIAL_NO_STR0, CFG-USB-SERIAL_NO_STR1, CFG-USB-SERIAL_NO_STR2, CFG-USB-SERIAL_NO_STR3 |
| UBX-CFG-USB.vendorID | CFG-USB-VENDOR_ID |
| UBX-CFG-USB.vendorString | CFG-USB-VENDOR_STR0, CFG-USB-VENDOR_STR1, CFG-USB-VENDOR_STR2, CFG-USB-VENDOR_STR3 |

**Table 38: Legacy UBX message fields and the corresponding configuration items**

# Configuration defaults

The following tables contain the configuration defaults for the firmware. Some of these values may be changed in production. Refer to the integration manual for product-specific details.

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-HW-ANT_CFG_VOLTCTRL | 0x10a3002e | L | - | - | 0 (false) |
| CFG-HW-ANT_CFG_SHORTDET | 0x10a3002f | L | - | - | 0 (false) |
| CFG-HW-ANT_CFG_SHORTDET_POL | 0x10a30030 | L | - | - | 1 (true) |
| CFG-HW-ANT_CFG_OPENDET | 0x10a30031 | L | - | - | 0 (false) |
| CFG-HW-ANT_CFG_OPENDET_POL | 0x10a30032 | L | - | - | 1 (true) |
| CFG-HW-ANT_CFG_PWRDOWN | 0x10a30033 | L | - | - | 0 (false) |
| CFG-HW-ANT_CFG_PWRDOWN_POL | 0x10a30034 | L | - | - | 1 (true) |
| CFG-HW-ANT_CFG_RECOVER | 0x10a30035 | L | - | - | 0 (false) |
| CFG-HW-ANT_SUP_SWITCH_PIN | 0x20a30036 | U1 | - | - | 16 |
| CFG-HW-ANT_SUP_SHORT_PIN | 0x20a30037 | U1 | - | - | 15 |
| CFG-HW-ANT_SUP_OPEN_PIN | 0x20a30038 | U1 | - | - | 8 |

**Table 39: CFG-HW configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-I2C-ADDRESS | 0x20510001 | U1 | - | - | 132 |
| CFG-I2C-EXTENDEDTIMEOUT | 0x10510002 | L | - | - | 0 (false) |
| CFG-I2C-ENABLED | 0x10510003 | L | - | - | 1 (true) |

**Table 40: CFG-I2C configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-I2CINPROT-UBX | 0x10710001 | L | - | - | 1 (true) |

**Table 41: CFG-I2CINPROT configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-I2COUTPROT-UBX | 0x10720001 | L | - | - | 1 (true) |

**Table 42: CFG-I2COUTPROT configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-INFMSG-UBX_I2C | 0x20920001 | X1 | - | - | 0x07 (ERROR \| WARNING \| NOTICE) |
| CFG-INFMSG-UBX_UART1 | 0x20920002 | X1 | - | - | 0x07 (ERROR \| WARNING \| NOTICE) |
| CFG-INFMSG-UBX_UART2 | 0x20920003 | X1 | - | - | 0x07 (ERROR \| WARNING \| NOTICE) |
| CFG-INFMSG-UBX_USB | 0x20920004 | X1 | - | - | 0x07 (ERROR \| WARNING \| NOTICE) |
| CFG-INFMSG-UBX_SPI | 0x20920005 | X1 | - | - | 0x07 (ERROR \| WARNING \| NOTICE) |
| CFG-INFMSG-NMEA_I2C | 0x20920006 | X1 | - | - | 0x00 |
| CFG-INFMSG-NMEA_UART1 | 0x20920007 | X1 | - | - | 0x00 |
| CFG-INFMSG-NMEA_UART2 | 0x20920008 | X1 | - | - | 0x00 |
| CFG-INFMSG-NMEA_USB | 0x20920009 | X1 | - | - | 0x00 |

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-INFMSG-NMEA_SPI | 0x2092000a | X1 | - | - | 0x00 |

**Table 43: CFG-INFMSG configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-ITFM-BBTHRESHOLD | 0x20410001 | U1 | - | - | 3 |
| CFG-ITFM-CWTHRESHOLD | 0x20410002 | U1 | - | - | 15 |
| CFG-ITFM-ENABLE | 0x1041000d | L | - | - | 0 (false) |
| CFG-ITFM-ANTSETTING | 0x20410010 | E1 | - | - | 0 (UNKNOWN) |
| CFG-ITFM-ENABLE_AUX | 0x10410013 | L | - | - | 0 (false) |

**Table 44: CFG-ITFM configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-MSGOUT-UBX_LOG_INFO_I2C | 0x20910259 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_LOG_INFO_SPI | 0x2091025d | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_LOG_INFO_UART1 | 0x2091025a | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_LOG_INFO_UART2 | 0x2091025b | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_LOG_INFO_USB | 0x2091025c | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_HW2_I2C | 0x209101b9 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_HW2_SPI | 0x209101bd | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_HW2_UART1 | 0x209101ba | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_HW2_UART2 | 0x209101bb | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_HW2_USB | 0x209101bc | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_HW_I2C | 0x209101b4 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_HW_SPI | 0x209101b8 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_HW_UART1 | 0x209101b5 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_HW_UART2 | 0x209101b6 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_HW_USB | 0x209101b7 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_IO_I2C | 0x209101a5 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_IO_SPI | 0x209101a9 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_IO_UART1 | 0x209101a6 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_IO_UART2 | 0x209101a7 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_IO_USB | 0x209101a8 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_MSGPP_I2C | 0x20910196 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_MSGPP_SPI | 0x2091019a | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_MSGPP_UART1 | 0x20910197 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_MSGPP_UART2 | 0x20910198 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_MSGPP_USB | 0x20910199 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_RXBUF_I2C | 0x209101a0 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_RXBUF_SPI | 0x209101a4 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_RXBUF_UART1 | 0x209101a1 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_RXBUF_UART2 | 0x209101a2 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_RXBUF_USB | 0x209101a3 | U1 | - | - | 0 |

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-MSGOUT-UBX_MON_RXR_I2C | 0x20910187 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_RXR_SPI | 0x2091018b | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_RXR_UART1 | 0x20910188 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_RXR_UART2 | 0x20910189 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_RXR_USB | 0x2091018a | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_TXBUF_I2C | 0x2091019b | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_TXBUF_SPI | 0x2091019f | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_TXBUF_UART1 | 0x2091019c | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_TXBUF_UART2 | 0x2091019d | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_MON_TXBUF_USB | 0x2091019e | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_RXM_PMP_I2C | 0x2091031d | U1 | - | - | 1 |
| CFG-MSGOUT-UBX_RXM_PMP_SPI | 0x20910321 | U1 | - | - | 1 |
| CFG-MSGOUT-UBX_RXM_PMP_UART1 | 0x2091031e | U1 | - | - | 1 |
| CFG-MSGOUT-UBX_RXM_PMP_UART2 | 0x2091031f | U1 | - | - | 1 |
| CFG-MSGOUT-UBX_RXM_PMP_USB | 0x20910320 | U1 | - | - | 1 |
| CFG-MSGOUT-UBX_RXM_SFRBX_I2C | 0x20910231 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_RXM_SFRBX_SPI | 0x20910235 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_RXM_SFRBX_UART1 | 0x20910232 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_RXM_SFRBX_UART2 | 0x20910233 | U1 | - | - | 0 |
| CFG-MSGOUT-UBX_RXM_SFRBX_USB | 0x20910234 | U1 | - | - | 0 |

**Table 45: CFG-MSGOUT configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-PM-EXTINTSEL | 0x20d0000b | E1 | - | - | 0 (EXTINT0) |
| CFG-PM-EXTINTWAKE | 0x10d0000c | L | - | - | 0 (false) |
| CFG-PM-EXTINTBACKUP | 0x10d0000d | L | - | - | 0 (false) |
| CFG-PM-EXTINTINACTIVE | 0x10d0000e | L | - | - | 0 (false) |
| CFG-PM-EXTINTINACTIVITY | 0x40d0000f | U4 | 0.001 | s | 0 |

**Table 46: CFG-PM configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-PMP-CENTER_FREQUENCY | 0x40b10011 | U4 | - | Hz | 1539812500 |
| CFG-PMP-SEARCH_WINDOW | 0x30b10012 | U2 | - | Hz | 2200 |
| CFG-PMP-USE_SERVICE_ID | 0x10b10016 | L | - | - | 1 (true) |
| CFG-PMP-SERVICE_ID | 0x30b10017 | U2 | - | - | 50821 |
| CFG-PMP-DATA_RATE | 0x30b10013 | E2 | - | bps | 2400 (B2400) |
| CFG-PMP-USE_DESCRAMBLER | 0x10b10014 | L | - | - | 1 (true) |
| CFG-PMP-DESCRAMBLER_INIT | 0x30b10015 | U2 | - | - | 23560 |
| CFG-PMP-USE_PRESCRAMBLING | 0x10b10019 | L | - | - | 0 (false) |
| CFG-PMP-UNIQUE_WORD | 0x50b1001a | U8 | - | - | 16238547128276412563 |

**Table 47: CFG-PMP configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-RATE-MEAS | 0x30210001 | U2 | 0.001 | s | 1000 |
| CFG-RATE-NAV | 0x30210002 | U2 | - | - | 1 |
| CFG-RATE-TIMEREF | 0x20210003 | E1 | - | - | 1 (GPS) |

**Table 48: CFG-RATE configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-RINV-DUMP | 0x10c70001 | L | - | - | 0 (false) |
| CFG-RINV-BINARY | 0x10c70002 | L | - | - | 0 (false) |
| CFG-RINV-DATA_SIZE | 0x20c70003 | U1 | - | - | 22 |
| CFG-RINV-CHUNK0 | 0x50c70004 | X8 | - | - | 0x203a656369746f4e ("Notice: ") |
| CFG-RINV-CHUNK1 | 0x50c70005 | X8 | - | - | 0x2061746164206f6e ("no data ") |
| CFG-RINV-CHUNK2 | 0x50c70006 | X8 | - | - | 0x0000216465766173 ("saved!\0\0") |
| CFG-RINV-CHUNK3 | 0x50c70007 | X8 | - | - | 0x0000000000000000 |

**Table 49: CFG-RINV configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-SPI-MAXFF | 0x20640001 | U1 | - | - | 50 |
| CFG-SPI-CPOLARITY | 0x10640002 | L | - | - | 0 (false) |
| CFG-SPI-CPHASE | 0x10640003 | L | - | - | 0 (false) |
| CFG-SPI-EXTENDEDTIMEOUT | 0x10640005 | L | - | - | 0 (false) |
| CFG-SPI-ENABLED | 0x10640006 | L | - | - | 0 (false) |

**Table 50: CFG-SPI configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-SPIINPROT-UBX | 0x10790001 | L | - | - | 1 (true) |

**Table 51: CFG-SPIINPROT configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-SPIOUTPROT-UBX | 0x107a0001 | L | - | - | 1 (true) |

**Table 52: CFG-SPIOUTPROT configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-TXREADY-ENABLED | 0x10a20001 | L | - | - | 0 (false) |
| CFG-TXREADY-POLARITY | 0x10a20002 | L | - | - | 0 (false) |
| CFG-TXREADY-PIN | 0x20a20003 | U1 | - | - | 0 |
| CFG-TXREADY-THRESHOLD | 0x30a20004 | U2 | - | - | 0 |
| CFG-TXREADY-INTERFACE | 0x20a20005 | E1 | - | - | 0 (I2C) |

**Table 53: CFG-TXREADY configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-UART1-BAUDRATE | 0x40520001 | U4 | - | - | 9600 |
| CFG-UART1-STOPBITS | 0x20520002 | E1 | - | - | 1 (ONE) |
| CFG-UART1-DATABITS | 0x20520003 | E1 | - | - | 0 (EIGHT) |
| CFG-UART1-PARITY | 0x20520004 | E1 | - | - | 0 (NONE) |

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-UART1-ENABLED | 0x10520005 | L | - | - | 1 (true) |

**Table 54: CFG-UART1 configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-UART1INPROT-UBX | 0x10730001 | L | - | - | 1 (true) |

**Table 55: CFG-UART1INPROT configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-UART1OUTPROT-UBX | 0x10740001 | L | - | - | 1 (true) |

**Table 56: CFG-UART1OUTPROT configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-UART2-BAUDRATE | 0x40530001 | U4 | - | - | 9600 |
| CFG-UART2-STOPBITS | 0x20530002 | E1 | - | - | 1 (ONE) |
| CFG-UART2-DATABITS | 0x20530003 | E1 | - | - | 0 (EIGHT) |
| CFG-UART2-PARITY | 0x20530004 | E1 | - | - | 0 (NONE) |
| CFG-UART2-ENABLED | 0x10530005 | L | - | - | 1 (true) |
| CFG-UART2-REMAP | 0x10530006 | L | - | - | 0 (false) |

**Table 57: CFG-UART2 configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-UART2INPROT-UBX | 0x10750001 | L | - | - | 1 (true) |

**Table 58: CFG-UART2INPROT configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-UART2OUTPROT-UBX | 0x10760001 | L | - | - | 0 (false) |

**Table 59: CFG-UART2OUTPROT configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-USB-ENABLED | 0x10650001 | L | - | - | 1 (true) |
| CFG-USB-SELFPOW | 0x10650002 | L | - | - | 1 (true) |
| CFG-USB-VENDOR_ID | 0x3065000a | U2 | - | - | 5446 |
| CFG-USB-PRODUCT_ID | 0x3065000b | U2 | - | - | 425 |
| CFG-USB-POWER | 0x3065000c | U2 | - | mA | 0 |
| CFG-USB-VENDOR_STR0 | 0x5065000d | X8 | - | - | 0x4120786f6c622d75 ("u-blox A") |
| CFG-USB-VENDOR_STR1 | 0x5065000e | X8 | - | - | 0x2e777777202d2047 ("G - www.") |
| CFG-USB-VENDOR_STR2 | 0x5065000f | X8 | - | - | 0x632e786f6c622d75 ("u-blox.c") |
| CFG-USB-VENDOR_STR3 | 0x50650010 | X8 | - | - | 0x0000000000006d6f ("om\0\0\0\0\0") |
| CFG-USB-PRODUCT_STR0 | 0x50650011 | X8 | - | - | 0x4720786f6c622d75 ("u-blox G") |
| CFG-USB-PRODUCT_STR1 | 0x50650012 | X8 | - | - | 0x656365722053534e ("NSS rece") |
| CFG-USB-PRODUCT_STR2 | 0x50650013 | X8 | - | - | 0x0000000072657669 ("iver\0\0\0\0") |

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-USB-PRODUCT_STR3 | 0x50650014 | X8 | - | - | 0x0000000000000000 |
| CFG-USB-SERIAL_NO_STR0 | 0x50650015 | X8 | - | - | 0x0000000000000000 |
| CFG-USB-SERIAL_NO_STR1 | 0x50650016 | X8 | - | - | 0x0000000000000000 |
| CFG-USB-SERIAL_NO_STR2 | 0x50650017 | X8 | - | - | 0x0000000000000000 |
| CFG-USB-SERIAL_NO_STR3 | 0x50650018 | X8 | - | - | 0x0000000000000000 |

**Table 60: CFG-USB configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-USBINPROT-UBX | 0x10770001 | L | - | - | 1 (true) |

**Table 61: CFG-USBINPROT configuration defaults**

| Configuration item | Key ID | Type | Scale | Unit | Default value |
|---|---|---|---|---|---|
| CFG-USBOUTPROT-UBX | 0x10780001 | L | - | - | 1 (true) |

**Table 62: CFG-USBOUTPROT configuration defaults**

# Related documents

**[1]** NEO-D9S-00B Data sheet, UBX-21040020
NEO-D9S-00A Data sheet, UBX-21008859
NEO-D9S-01A Data sheet, UBX-21008860

**[2]** NEO-D9S integration manual, UBX-19026111

☞ For regular updates to u-blox documentation and to receive product change notifications please register on our homepage (https://www.u-blox.com).

# Revision history

| Revision | Date | Name | Status / Comments |
|---|---|---|---|
| R01 | 21-Oct-2021 | dama | PMP 1.04 release<br>For document legacy revisions see UBX-19048765 |

# Contact

For complete contact information visit us at www.u-blox.com.

**u-blox Offices**

**North, Central and South America**

**u-blox America, Inc.**
Phone: +1 703 483 3180
E-mail: info_us@u-blox.com

**Regional Office West Coast**
Phone: +1 408 573 3640
E-mail: info_us@u-blox.com

**Technical Support**
Phone: +1 703 483 3185
E-mail: support_us@u-blox.com

**Headquarters**

**Europe, Middle East, Africa**

**u-blox AG**
Phone: +41 44 722 74 44
E-mail: info@u-blox.com
Support: support@u-blox.com

**Asia, Australia, Pacific**

**u-blox Singapore Pte. Ltd.**
Phone: +65 6734 3811
E-mail: info_ap@u-blox.com
Support: support_ap@u-blox.com

**Regional Office Australia**
Phone: +61 3 9566 7255
E-mail: info_anz@u-blox.com
Support: support_ap@u-blox.com

**Regional Office China (Beijing)**
Phone: +86 10 68 133 545
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

**Regional Office China (Chongqing)**
Phone: +86 23 6815 1588
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

**Regional Office China (Shanghai)**
Phone: +86 21 6090 4832
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

**Regional Office China (Shenzhen)**
Phone: +86 755 8627 1083
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

**Regional Office India**
Phone: +91 80 4050 9200
E-mail: info_in@u-blox.com
Support: support_in@u-blox.com

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

**Regional Office Japan (Tokyo)**
Phone: +81 3 5775 3850
E-mail: info_jp@u-blox.com
Support: support_jp@u-blox.com

**Regional Office Korea**
Phone: +82 2 542 0861
E-mail: info_kr@u-blox.com
Support: support_kr@u-blox.com

**Regional Office Taiwan**
Phone: +886 2 2657 1090
E-mail: info_tw@u-blox.com
Support: support_tw@u-blox.com