Table of Contents

Dronecode Shortcuts

MAVLink Developer Guide

MAVLink is a very lightweight, header-only message marshalling library for micro air vehicles / drones.

MAVLink follows a modern hybrid publish-subscribe and point-to-point design pattern: Data streams are sent / published as **topics** while configuration sub-protocols such as the mission protocol or parameter protocol are point-to-point with retransmission.

Messages are defined within XML files, which may then be generated into appropriate source code for each of the supported languages. Each XML file defines the message set supported by a particular system, also referred to as a "dialect". The reference message set that is implemented by most ground control stations and autopilots is defined in common.xml (most dialects *build on top of* this definition).

MAVLink was first released early 2009 by Lorenz Meier and has now a significant number of contributors.

> Because MAVLink doesn't require any additional framing it is very well suited for applications with very limited communication bandwidth. It's reference implementation in C is highly optimized for resource-constrained systems with limited RAM and flash memory. It is field-proven and deployed in many products where it serves as interoperability interface between components of different manufacturers.

Supported Languages

MAVLink 2 source files can be generated for:

- C
- C++11

MAVLink 1 source files can be generated for:

- C
- C#
- Objective C
- Java
- JavaScript
- Lua
- Swift
- Python (2.7+, 3.3+)

Prebuilt MAVLink Source Files

C MAVLink Source Files (only) are auto-generated for the latest versions of all message specifications/dialects (for both MAVLink 1 and 2):

- c_library_v2 (MAVLink 2)
- c_library_v1 (MAVLink 1)

Using Generated Source Files explains how to use these libraries.

Forums and Chat

The core development team and community are active on the following chat channel:

- Slack (sign up) - #mavlink channel

Reporting Bugs & Issues

If you have any problems using MAVLink first post them on the support channels above.

If directed by the development team, issues may be raised on Github here.

Contributing

The Contributing Guide explains the contribution model and the main areas where you can help.

License

MAVLink is licensed under the terms of the Lesser General Public License (version 3) of the Free Software Foundation (LGPLv3). The C-language version of MAVLink is a header-only library which is generated as MIT-licensed code. MAVLink can therefore be used without limits in any closed-source application without publishing the source code of the closed-source application. See the COPYING file for more information.

Getting Started

Download or Generate MAVLink source files for your dialect:

- **Download the pre-built MAVLink source files** if you're working in a C/C++ project and using standard dialects.
- **Generate the MAVLink source files** to use any other supported language, add/modify messages or dialects, or use the example scripts:
  1. Install MAVLink
  2. Generate Language-Specific Source Files.

The following topics explain how to include the files in your project and use MAVLink:

- Use the MAVLink Source Files explains how to include the source files in your project and send messages.
- Message Definitions contains human-readable explanations of the messages.
- Protocols explains the main sub-protocols for working with missions, cameras, images, parameters etc.

Installing MAVLink

This topic explains how to install the MAVLink toolchain, including both XML message definitions and the GUI/command line tools that use them to Generate MAVLink Source Files.

> If you are using Prebuilt MAVLink Source Files you do not need to install or generate the source files. After getting the libraries see Using Generated Source Files.

Prerequisites

The requirements for using the *MAVLink tools* are:

- Python 2.7+ or Python 3.3+
- Python future module
- (Optional) Python TkInter module (required to use the GUI tool).
- `PYTHONPATH` environment variable must be set to the directory path containing the *mavlink* repository.

Installation Steps

The main installation steps are:

1. Install Python 2.7+ or 3.3+.
   - **Windows:** Download from Python for Windows
   - **Ubuntu Linux 16.04:** Python 2.7 and Python 3.0 are already present. If you are using Python3 you will need to install the *pip3* package manager:

     ```
     sudo apt-get install python3-pip
     ```

2. Install the *future* module:
   - **Windows:**

     ```
     pip install future
     ```

   - **Linux:**

     ```
     pip install --user future
     ```

3. (Optionally) Install TkInter
   - **Windows:** Installed already as part of *Python for Windows*
   - **Linux:** Enter the following terminal command:

     ```
     sudo apt-get install python-tk
     ```

4. Clone the mavlink repo (or your fork) into a user-writable directory:

   ```
   git clone https://github.com/mavlink/mavlink.git
   git submodule update --init --recursive
   ```

> Alternatively you can do this in one line:
>
> ```
> git clone https://github.com/mavlink/mavlink.git --recursive
> ```

5. Set `PYTHONPATH` to the directory path containing your *mavlink* repository.
   - **Windows:** `set PYTHONPATH=C:\your_path_to_mavlink_clone`
   - **Linux:** `PYTHONPATH=your_path_to_mavlink_clone`

Now you are ready to Generate Source Files.

Generating Source Files

Language-specific MAVLink libraries ("source files") can be created from XML Message Definitions using Python-based command line or GUI generator tools. The supported output languages for each version of the MAVLink protocol are listed in Supported Languages (these include C, C#, Java, Python etc).

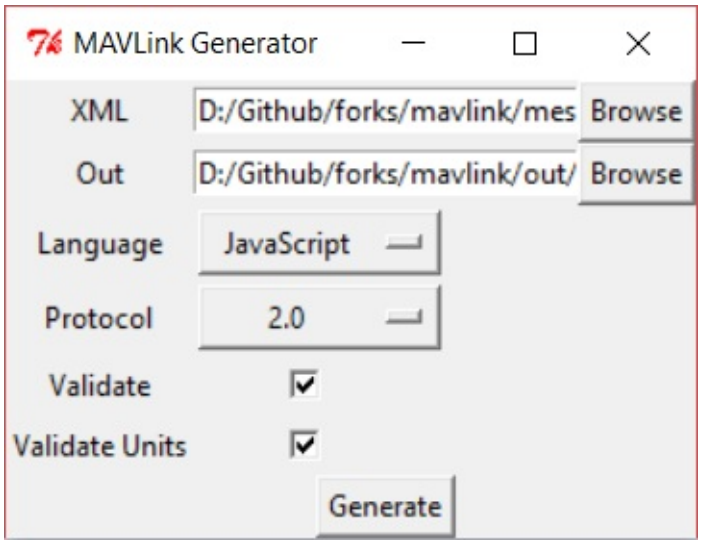
> The tools must already have been set up as described in Install MAVLink (in particular *Tkinter* must be installed to use the GUI tool).

> message_definitions/v1.0/ contains XML definitions for a number of different dialects. Dialect files that have dependencies on other XML files must be located in the same directory. Since most MAVLink dialects depend on the common.xml message set, you should place your dialect with the others in that folder.

MAVLink Generator Tool (GUI)

MAVLink Generator (**mavgenerate.py**) is a header generation tool GUI written in Python. It can be run from anywhere using Python's `-m` argument:

```
python -m mavgenerate
```



Generator Steps:

1. Choose the target XML file (typically in mavlink/message_definitions/1.0).

   > If using a custom dialect, first copy it into the above directory (if the dialect is dependent on **common.xml** it must be located in the same directory).

2. Choose an output directory (e.g. **mavlink/include**).
3. Select the target output programming language.
4. Select the target MAVLink protocol version (ideally 2.0)

   > Generation will fail if the protocol is not supported by the selected programming language.

5. Optionally check *Validate* and/or *Validate Units* (if checked validates XML specifications).
6. Click **Generate** to create the source files.

Mavgen (Command Line)

mavgen.py is a command-line interface for generating a language-specific MAVLink library. After the `mavlink` directory has been added to the `PYTHONPATH`, it can be run by executing from the command line.

> This is the backend used by **mavgenerate.py**. The documentation below explains all the options for both tools.

For example, to generate *MAVLink 2* C libraries for a dialect named **your_custom_dialect.xml**:

```
python -m pymavlink.tools.mavgen --lang=C --wire-protocol=2.0 --output=generated/include/mavlink/v2.0 message_definit
ions/v1.0/your_custom_dialect.xml
```

The full syntax can be output by running *mavgen* with the `-h` flag (reproduced below):

```
usage: mavgen.py [-h] [-o OUTPUT]
                 [--lang {C,CS,JavaScript,Python,WLua,ObjC,Swift,Java,C++11}]
                 [--wire-protocol {0.9,1.0,2.0}] [--no-validate]
                 [--error-limit ERROR_LIMIT] [--strict-units]
                 XML [XML ...]

This tool generate implementations from MAVLink message definitions

positional arguments:
  XML                   MAVLink definitions

optional arguments:
  -h, --help            show this help message and exit
  -o OUTPUT, --output OUTPUT
                        output directory.
  --lang {C,CS,JavaScript,Python,WLua,ObjC,Swift,Java,C++11}
                        language of generated code [default: Python]
  --wire-protocol {0.9,1.0,2.0}
                        MAVLink protocol version. [default: 1.0]
  --no-validate         Do not perform XML validation. Can speed up code
                        generation if XML files are known to be correct.
  --error-limit ERROR_LIMIT
                        maximum number of validation errors to display
  --strict-units        Perform validation of units attributes.
```

Using Generated Source Files

The generated MAVLink dialect libraries are used differently depending on the programming language. Language-specific details are given in each of the following sections.

> At time of writing we only have usage instructions for the C source files. If you have experience with the other supported languages please help us update those sections!

C

To use MAVLink, include the **mavlink.h** header file in your project:

```
#include <mavlink/mavlink.h>
```

If headers for multiple dialects and/or versions are installed, your include path might instead look similar to the following:

```
#include <mavlink/v2.0/common/mavlink.h>
```

> *Do not include the individual message files.* If you generate your own headers, you will have to add their output location to your C compiler's search path.

When compiling the project, we recommend that you specify the top-level output directory AND all generated dialects and versions (this will give the greatest compatibility with existing code and examples):

```
$ gcc ... -I generated/include -I generated/include/mavlink/v2.0/common ...
```

Multiple streams, a.k.a. "channels"

The C MAVLink library utilizes a "channel" metaphor to allow for simultaneous processing of multiple, independent MAVLink streams in the same program. It is therefore important to use the correct channel for each operation as all receiving and transmitting functions provided by MAVLink require a channel. If only one MAVLink stream exists, channel 0 should be used by specifying the `MAVLINK_COMM_0` constant.

Receiving

MAVLink reception is then done using `mavlink_helpers.h:mavlink_parse_char()`.

Transmitting

Transmitting can be done by using the `mavlink_msg_*_pack()` function, where one is defined for every message. The packed message can then be serialized with `mavlink_helpers.h:mavlink_msg_to_send_buffer()` and then writing the resultant byte array out over the appropriate serial interface.

It is possible to simplify the above by writing wrappers around the transmitting/receiving code. A multi-byte writing macro can be defined, `MAVLINK_SEND_UART_BYTES()`, or a single-byte function can be defined, `comm_send_ch()`, that wrap the low-level driver for transmitting the data. If this is done, `MAVLINK_USE_CONVENIENCE_FUNCTIONS` must be defined.

C++11

TBD

C

TBD

Objective C

TBD

Java

TBD

JavaScript

TBD

Lua

TBD

Swift

TBD

Python

TBD

Scripts/Examples

MAVLink C-UART Interface Example

The C-UART Interface Example is a simple C example of a MAVLink to UART interface for Unix-like systems.

The example source code demonstrates how to set up serial communication between Pixhawk and an offboard computer via USB or a telemetry radio, how to put the vehicle in offboard mode, and how to send and receive MAVLink messages over the interface.

Source code and instructions for how to run the example can be found in the Github repo:
mavlink/c_uart_interface_example

MAVLink UDP Example

The MAVLink UDP Example is a simple C example of a MAVLink UDP interface for Unix-like systems.

The example sends some data to *QGroundControl* using MAVLink over UDP. *QGroundControl* responds with heartbeats and other messages, which are then printed by this program.

Source code and instructions for how to run the example can be found in the Github repo here:
mavlink/mavlink/examples/linux

Pymavlink Scripts

This MAVLink library also comes with supporting libraries and scripts for using, manipulating, and parsing MAVLink streams within the pymavlink, **pymavlink/tools**, and **pymavlink/examples** directories.

The scripts have the following requirements:

- Python 2.7+ and 3.3+
- `PYTHONPATH` specifies the directory path that contains the `mavlink` repository.
- Write access to the entire **mavlink** folder.
- Your dialect's XML file is in `message_definitions/*/`

The scripts can be executed by running Python with the `-m` switch, which indicates that the given script exists on the `PYTHONPATH`. The following code runs **mavlogdump.py** in **/pymavlink/tools/** on the recorded MAVLink stream `test_run.mavlink` (other scripts in **/tools** and **/scripts** can be run in a similar fashion):

```
python -m pymavlink.tools.mavlogdump test_run.mavlink
```

> Using the `-m` switch is the proper way to run Python scripts that are part of a library as per PEP-328 (and the rejected PEP-3122).

Generate Source Files for ROS

To add MAVlink messages/dialects while working with ROS:

1. Follow the MAVROS source install instructions to install the mavlink-gbp-release which is the MAVlink library released for ROS.
2. Uninstall the MAVlink package for ROS (if previously installed).

   ```
   sudo apt-get remove ros-${rosversion -d}-mavlink
   ```

   or source `devel/setup.bash` of your catkin workspace to override the library directory.
3. In the `mavlink-gbp-release` , add the new MAVlink message to `common.xml` or add the new dialect `dialect_name.xml` in the `message definitions` . Do not checkout your MAVlink branch because it is not the version synced with ROS.
4. Generate the MAVlink headers `catkin build mavlink` . You can find the headers in `~/catkin_ws/build/mavlink/include/` .

MAVLink Version

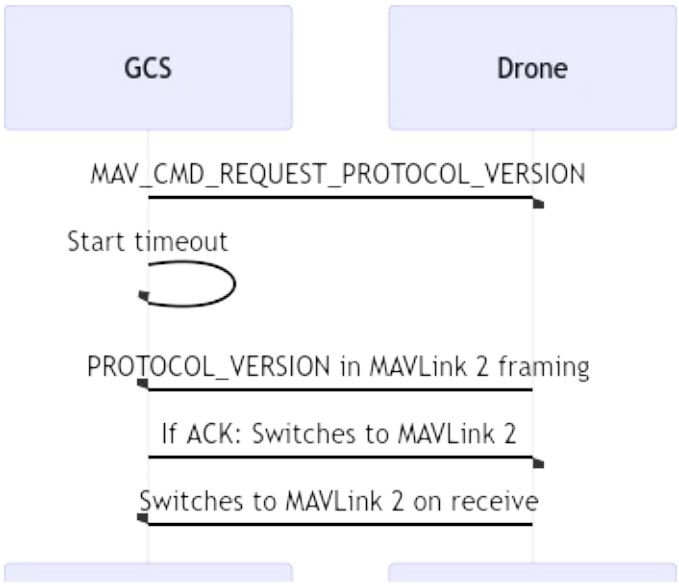
MAVLink is deployed in two major versions: v1.0, which was widely adopted around 2013 and v2.0, which was adopted by major users early 2017 but there are still quite a few legacy peripherals in use not supporting it. The MAVLink 2.0 C/C++ and Python libraries are backwards compatible and support also MAVLink 1.0. This has tremendously simplified the transition.

Version Handshaking

Support for MAVLink 2 is indicated in the AUTOPILOT_VERSION message by the MAVLINK2 flag. This is sufficient if the communication link between autopilot and GCS is completely transparent. However, most communication links are not completely transparent as they either include routing or in case of fixed-length wireless implementations on packetization. In order to also test the link, the MAVLink 2 handshake protocol sends a MAVLink 2 frame to test the complete communication chain.

To do so, the GCS sends a COMMAND_LONG or COMMAND_INT message with the command ID MAV_CMD_REQUEST_PROTOCOL_VERSION.

If the system supports MAVLink 2 and the handshake it will respond with PROTOCOL_VERSION **encoded as MAVLink 2 packet**. If it does not support MAVLink 2 it should NACK the command. The GCS should fall back to a timeout in case the command interface is not implemented properly. The diagram below illustrates the complete sequence.



Semi-transparent legacy radios

Some popular legacy radios (e.g. the SiK radio series) operate in semi-transparent mode by injecting RADIO_STATUS messages into the MAVLink message stream. Per MAVLink spec these should actually emit a heartbeat with the same system ID and a different component ID than the autopilot to be discoverable. However, an additional heartbeat could be an issue for deployed systems. Therefore these radios can alternatively confirm their v2 compliance by emitting `RADIO_STATUS` in v2 message format after receiving the first v2 MAVLink frame.

MAVLink 2

MAVLink version 2 is a backward-compatible new version. The main new features are:

- Support for more than 256 message IDs (now 24 bit or over 16 million packets)
- Packet signing (authentication)
- Support for extending existing messages
- Support for variable length arrays

Upgrading an existing C installation

The existing v1 library can be simply upgraded by dropping in the v2 library from Github:

https://github.com/mavlink/c_library_v2

The v2 library will send packets in MAVLink v2 framing by default. In order to default to v1, run this code snippet on boot:

```
mavlink_status_t* chan_state = mavlink_get_channel_status(MAVLINK_COMM_0);
chan_state->flags |= MAVLINK_STATUS_FLAG_OUT_MAVLINK1;
```

It is advisable to switch to MAVLink v2 when the communication partner sends MAVLink v2 - the complete handshaking for versions is described in this guide. The minimal solution is to watch incoming packets using code similar to this:

```
if (mavlink_parse_char(MAVLINK_COMM_0, buf[i], &msg, &status)) {

    // check if we received version 2 and request a switch.
    if (!(mavlink_get_channel_status(MAVLINK_COMM_0)->flags & MAVLINK_STATUS_FLAG_IN_MAVLINK1)) {
        // this will only switch to proto version 2
        chan_state->flags &= ~(MAVLINK_STATUS_FLAG_OUT_MAVLINK1);
    }
}
```

Protocol Overview

MAVLink is a binary telemetry protocol designed for resource-constrained systems and bandwidth-constrained links. MAVLink is deployed in two major versions: v1.0 and v2.0, which is backwards-compatible (v2.0 implementations can parse and send v1.0 packets). Telemetry data streams are sent in a multicast design while protocol aspects that change the system configuration and require guaranteed delivery like the mission protocol or parameter protocol are point-to-point with retransmission.

MAVLink 1 Packet Format

Below is the over-the-wire format for a MAVLink 1 packet. The in-memory representation might differ.

```
uint8_t magic;               ///< protocol magic marker
uint8_t len;                 ///< Length of payload
uint8_t seq;                 ///< Sequence of packet
uint8_t sysid;               ///< ID of message sender system/aircraft
uint8_t compid;              ///< ID of the message sender component
uint8_t msgid;               ///< ID of message in payload
uint8_t payload[max 255];    ///< A maximum of 255 payload bytes
uint16_t checksum;           ///< X.25 CRC
```

MAVLink 2 Packet Format

Below is the over-the-wire format for a MAVLink 2 packet. The in-memory representation might differ.

```
uint8_t magic;              ///< protocol magic marker
uint8_t len;                ///< Length of payload
uint8_t incompat_flags;     ///< flags that must be understood
uint8_t compat_flags;       ///< flags that can be ignored if not understood
uint8_t seq;                ///< Sequence of packet
uint8_t sysid;              ///< ID of message sender system/aircraft
uint8_t compid;             ///< ID of the message sender component
uint8_t msgid 0:7;          ///< first 8 bits of the ID of the message
uint8_t msgid 8:15;         ///< middle 8 bits of the ID of the message
uint8_t msgid 16:23;        ///< last 8 bits of the ID of the message
uint8_t target_sysid;       ///< Optional field for point-to-point messages, used for payload else
uint8_t target_compid;      ///< Optional field for point-to-point messages, used for payload else
uint8_t payload[max 253];   ///< A maximum of 253 payload bytes
uint16_t checksum;          ///< X.25 CRC
uint8_t signature[13];      ///< Signature which allows ensuring that the link is tamper-proof
```

Serialization

The over-the-wire format of MAVLink is optimized for resource-constrained systems and hence the field order is not the same as in the XML specification. The over-the-wire generator sorts all fields of the message according to size, with the largest fields (uint64_t) first, then down to smaller fields. The sorting is done using a stable sorting algorithm, which ensures that fields that do not need to be reordered stay in the same order. This prevents alignment issues on the encoding / decoding systems and allows for very efficient packing / unpacking.

Multicast Streams vs. Guaranteed Delivery

MAVLink is built for hybrid networks where high-rate data streams from data sources (commonly drones) flow to data sinks (commonly ground stations), but are mixed with transfers requiring guaranteed delivery. The key insight is that for most **telemetry streams** there is not a known or single recipient: Instead, typically an onboard computer, a ground control station and a cloud system all need the same data stream.

On the other hand configuring the **onboard mission** or changing the system configuration with **onboard parameters** requires point-to-point communication with guaranteed delivery. MAVLink achieves very high efficiency by allowing both modes of operation.

Topic Mode (publish-subscribe)

In topic mode the protocol will not emit a target system and component ID for messages to save link bandwidth. Typical examples for this communication mode are all autopilot data streams like position, attitude, etc.

The main benefit of this multicast mode is that no additional overhead is generated and multiple subscribers can all receive this data.

Point-to-Point Mode

In point-to-point mode MAVLink uses a target ID and target component. In most cases where these fields are used the sub-protocol also ensures guaranteed delivery (missions, parameters, commands).

Integrity Checks / Checksum

MAVLink implements two integrity checks: The first check is on the integrity of the packet during transmission using the X.25 checksum (CRC-16-CCITT). This however only ensures that the data has not been altered on the link - it does not ensure consistency with the data definition. The second integrity check is on the data description to ensure that two messages with the same ID are indeed containing the same information. To achieve this the data definition itself is run through CRC-16-CCITT and the resulting value is used to seed the packet CRC. Most reference implementations store this constant in an array named **CRC_EXTRA**.

Mission Protocol

The mission protocol is a sub-protocol supporting guaranteed delivery of messages. It allows to transfer a mission over a lossy link.

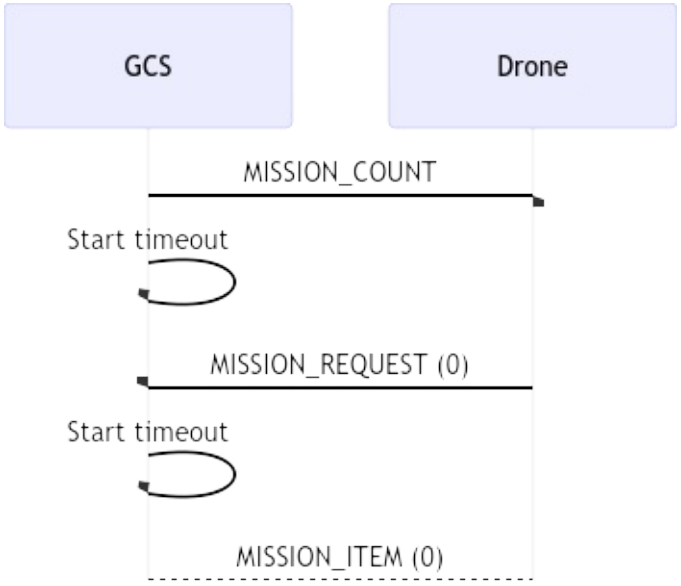
MAVLink 2 Extensions

MAVLink 2 supports several different mission types in the MISSION_COUNT and MISSION_REQUEST messages using the `mission_type` field. This instructs the autopilot to read and write to different lists containing the regular mission (MAV_MISSION_TYPE_MISSION), geofence (MAV_MISSION_TYPE_FENCE) and safe points used as alternate landing sites (MAV_MISSION_TYPE_RALLY).

On MAVLink 1 links the mission type defaults to `MISSION` and the ground control station should not attempt to send geofence or alternate landing spot coordinates.

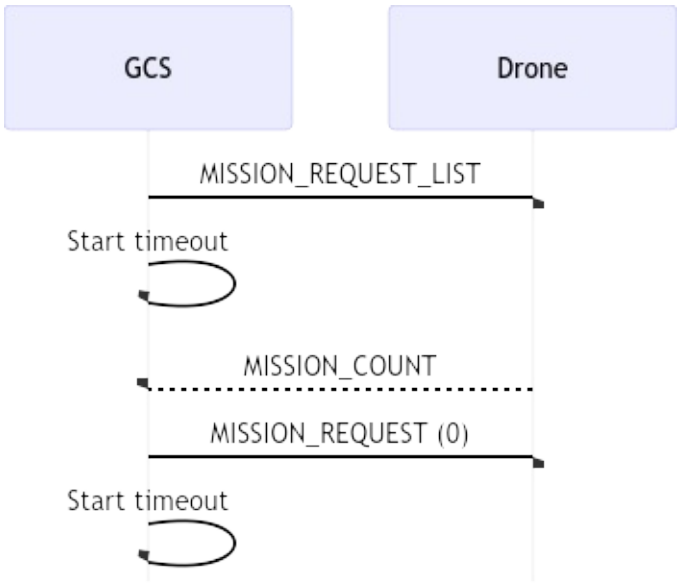
Upload a Mission to the Vehicle

The diagram below shows how the ground control station (GCS) can upload a mission to the drone. When MAVLink 2 is used several different mission types (regular waypoints or geofence coordinates, etc.) can be selected in the MISSION_COUNT and MISSION_REQUEST messages using the `mission_type` field.



Download a Mission from the Vehicle

The diagram below shows the communication sequence to download a mission from the drone. When MAVLink 2 is used several different mission types (regular waypoints or geofence coordinates, etc.) can be selected in the MISSION_COUNT and MISSION_REQUEST messages using the `mission_type` field.



Mission File Format

The standard file format for missions is JSON, as implemented in the QGroundControl reference implementation. The JSON file format has additional meta data which is not serialized over the link. The JSON file below shows an example mission with two waypoints.

```
{
    "fileType": "Plan",
    "geoFence": {
        "polygon": [
        ],
        "version": 1
    },
    "groundStation": "QGroundControl",
    "mission": {
        "cruiseSpeed": 16,
        "firmwareType": 12,
        "hoverSpeed": 4,
        "items": [
            {
                "autoContinue": true,
                "command": 22,
                "coordinate": [
                    47.385913889999998,
                    8.5520674900000007,
                    15
                ],
                "doJumpId": 1,
                "frame": 3,
                "params": [
                    0,
                    0,
                    0,
                    null
                ],
                "type": "SimpleItem"
            },
            {
                "autoContinue": true,
                "command": 16,
                "coordinate": [
                    47.383052030000002,
                    8.5556602700000006,
                    15
                ],
                "doJumpId": 2,
                "frame": 3,
                "params": [
                    0,
                    0,
                    0,
                    null
                ],
                "type": "SimpleItem"
            }
        ],
        "plannedHomePosition": [
            47.386183686176871,
            8.5520674900000007,
            15
        ],
        "vehicleType": 2,
        "version": 2
    },
    "rallyPoints": {
        "points": [
        ],
        "version": 1
    },
    "version": 1
}
```

Parameter Protocol

> This content has not been fully reviewed since being ported from the old website (and may be out of date). Updates/re-validation welcome!

The parameter protocol is used to exchange key system settings and guarantees delivery.

It can be both implemented on a microcontroller (e.g. the pxIMU with ARM7) and in standard software (e.g. px_multitracker process in Linux).

Supported Data Types

MAVLink (v1.0, v2.0) supports these data types:

- `uint32_t` - 32bit unsigned integer (use the ENUM value MAV_PARAM_TYPE_UINT32)
- `int32_t` - 32bit signed integer (use the ENUM value MAV_PARAM_TYPE_INT32)
- `float` - IEEE754 single precision floating point number (use the ENUM value MAV_PARAM_TYPE_FLOAT)

> All parameters are send as the float value of `mavlink_param_union_t`, which means that a parameter should be byte-wise converted with this union to a byte-wise float (no type conversion). This is necessary in order to not limit the maximum precision for scaled integer params. E.g. GPS coordinates can only be expressed with single float precision up to a few meters, while GPS coordinates in 1E7 scaled integers provide very high accuracy.

```
mavlink_param_union_t param;

int32_t integer = 20000;

param.param_int32 = integer;
param.type = MAV_PARAM_TYPE_INT32;

// Then send the param by providing the float bytes to the send function
mavlink_msg_param_set_send(xx, xx, param.param_float, param.type, xx);
```

Multi-System and Multi-Component Support

MAVLink supports multiple systems / airplanes in parallel on the same link. In addition to this, it also supports multiple MAVLink-enabled devices in the same airplane. The protocol for example allows to communicate over one radio link with the autopilot and a payload unit. For this reason the parameter protocol also differentiates between components. To get a complete parameter list from a system, send the request parameter message with `target_component` set to 0 (enum value: MAV_COMP_ID_ALL). All onboard components should respond to parameter request messages with their ID or with ID `MAV_COMP_ID_ALL` (0).

> *QGroundControl* by default queries all components of a system (it only queries the currently selected system, not all systems) and therefore sends ID 0 (`MAV_COMP_ID_ALL`).

Graphical User Interface in QGroundControl

For this reason the parameter interface *discriminates between systems (one system is one airplane) and components (one component is one entity in the architecture, e.g. the IMU or a Linux process).* This allows to transparently access the individual component parameters without the need of a central onboard unit that translates the parameter read/write requests for the onboard components.

As can be seen on the image below, each component is represented by a top-level node in the parameter tree. The system (the MAV) can be selected in the top-level drop-down menu. The GUI keeps track of changed parameters will send those parameters which changed to the appropriate components.

To facilitate the use of many parameters, the tree is structures at the top level according to the first underscore ("_") in the parameter name. So `PID_POS_X_P` and `PID_POS_Y_P` will be grouped below the `PID` node.

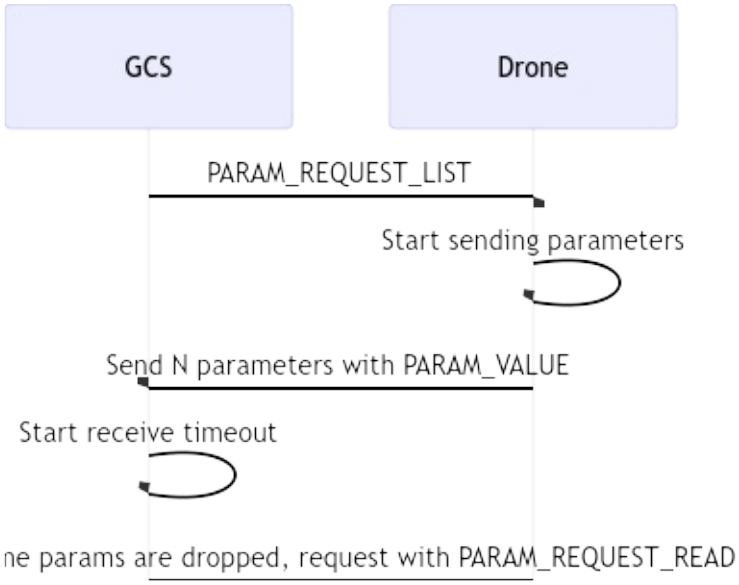
Communication / State Machine

The onboard parameters are identified by a 16-char string (without `\0`) and store a floating point (IEE 754 single-precision, 4 bytes) value. This key->value pair has many important properties:

- The human-readable name is very helpful for users, yet it is still small enough
- The GCS does not have to know in advance what onboard parameters exist
- Support for unknown autopilots, as long as they implement the protocol, is guaranteed
- Adding a parameter is only a change to the onboard code.

Read Parameters

Reading the parameter list is activated by sending the PARAM_REQUEST_LIST message. The onboard component should start to transmit the parameters individually after receiving this message. The sending should be delayed after each parameter, in order to not use up the full radio bandwidth.

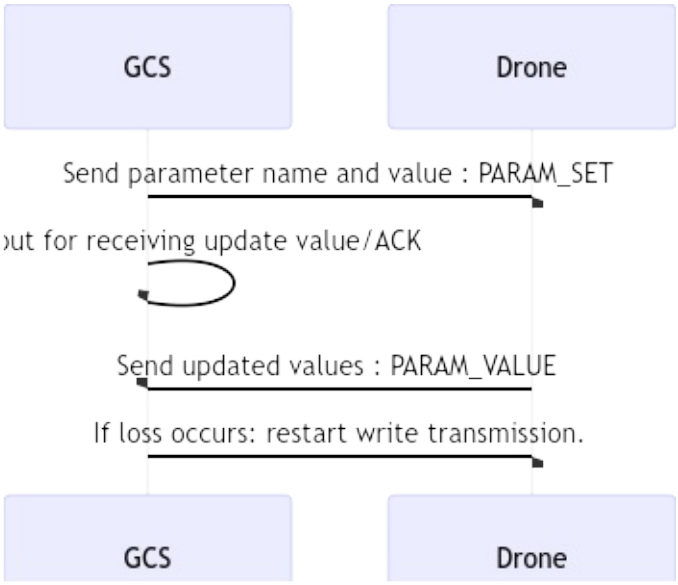


Read Single Parameter

A single parameter can be read by the PARAM_REQUEST_READ message.

Write Parameters

As a GCS does not have its own list of the parameters on startup, before writing a parameter first the parameter list has to be read once. After that, parameters can be written individually by sending the key->value pair to the component. Provided the GCS keeps track of changed parameters, it will only need to transmit those which have changed in value. The Drone (MAV) **has to acknowledge the write operation** by emitting a PARAM_VALUE value message with the newly written parameter value.

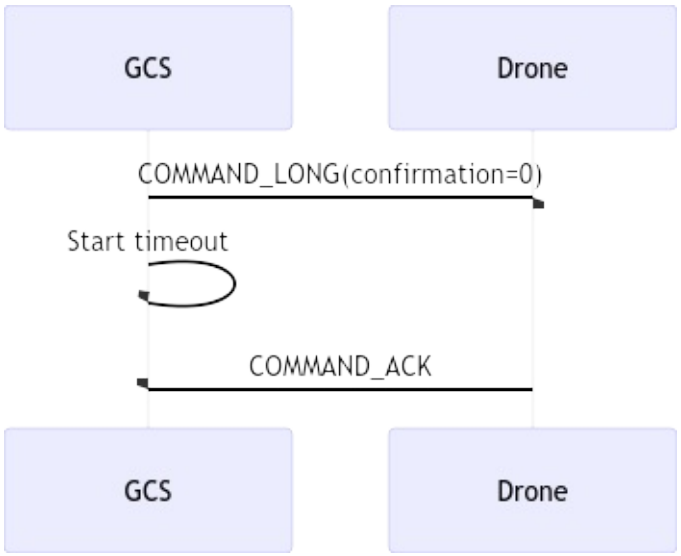


QGroundControl Parameter Files

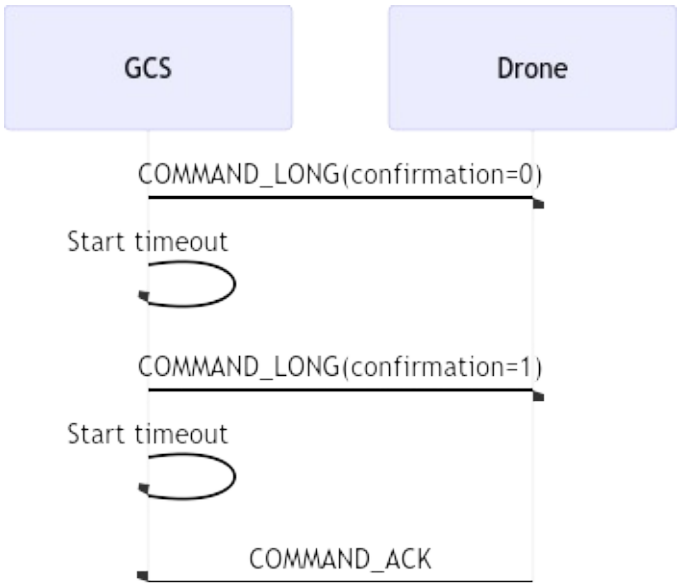
QGroundControl can save the current values of the onboard parameters in a text file. The file can then be imported again and transmitted to the Drone. This allows to e.g. configure several vehicles completely similar with safe default values.

Command Protocol

The MAVLink command protocol allows guaranteed delivery of commands. It consists of the original command message and the matching acknowledgement (ACK).



If the command drops the sender should increase the confirmation field:



Camera Protocol

The camera protocol is used to configure camera payloads and request their status. It supports photo and video cameras and includes messages to query and configure the onboard camera storage.

Camera Identification

The first step is to determine if a camera exists. Camera components are supposed to send heartbeats just like any other component. There are pre-defined component IDs for cameras - see MAV_COMP_ID_CAMERA. If a camera component exists, once a heartbeat is received a MAV_CMD_REQUEST_CAMERA_INFORMATION message is sent from the GCS. The camera component will then reply with a CAMERA_INFORMATION message.

This response contains the bare minimum information about the camera and what it can or cannot do. By itself, it is sufficient for default image and/or video capture. However, if a camera provides finer control over its settings, this message will also include an URI to a Camera Definition File. If this URI exists, the GCS will request it (using a standard HTTP GET request), parse it and prepare the UI for the user to control the camera settings. The definition file can be *hosted* anywhere. If the camera component provides an HTTP interface, the definition file can be hosted on the camera itself. Otherwise, it can be hosted by any regular, reachable server. The `CAMERA_INFORMATION` message should provide a version for the definition file (`cam_definition_version`), allowing the GCS to cache it. Once downloaded, it would only be requested again if the version number changes.

> If no response is sent for a `MAV_CMD_REQUEST_CAMERA_INFORMATION` message, it is assumed camera support is not available and no support for it will be provided by the GCS.

If a vehicle has more than one camera, each camera will have a different component ID and send their own heartbeats. The GCS will create multiple instances of a camera controller based on the component ID of each camera. All commands are sent to a specific camera by addressing the command to a specific component ID.

Basic Camera Operations

The CAMERA_INFORMATION message contains a `flags` field indicating the camera capabilities. The flag is a bit field based on the CAMERA_CAP_FLAGS enum. It will tell the GCS if the camera is able to capture still images and/or video, if it needs to be in a certain mode to capture, etc.

Camera Modes

Some cameras must be in a certain mode for still and/or video capture. The CAMERA_INFORMATION message `flags` field uses the CAMERA_CAP_FLAGS_HAS_MODES bit true to inform the GCS that it needs to make sure the camera is in the proper mode prior to sending a start capture (image or video) command. In addition, some cameras can capture images in any mode but with different resolutions. For example, a 20 megapixel camera would take a full resolution image when set to `CAMERA_MODE_IMAGE` but only at the current video resolution if it is set to `CAMERA_MODE_VIDEO` .

To get the current mode, the GCS would send a MAV_CMD_REQUEST_CAMERA_SETTINGS command. The current mode is sent back in the `mode_id` field of the CAMERA_SETTINGS message.

To set the camera to a specific mode, the GCS would send in turn the MAV_CMD_SET_CAMERA_MODE command with the appropriate mode.

Storage Status

Before capturing images and/or videos, the GCS will query the storage status to determine if the camera has enough free space for these operations (and provide the user with feedback as to the current storage status). The GCS will send the MAV_CMD_REQUEST_STORAGE_INFORMATION command and it expects a STORAGE_INFORMATION response. For formatting (or erasing depending on your implementation), the GCS will send a MAV_CMD_STORAGE_FORMAT command.

Camera Capture Status

In addition to querying about storage status, the GCS will also request the current *Camera Capture Status* in order to provide the user with proper UI indicators. The GCS will send a MAV_CMD_REQUEST_CAMERA_CAPTURE_STATUS command and it expects a CAMERA_CAPTURE_STATUS response.

Still Image Capture

If the `flags` field of the CAMERA_INFORMATION message has the CAMERA_CAP_FLAGS_CAPTURE_IMAGE bit set, it will indicate the GCS can send image capture commands to the camera.

To capture an image, the GCS uses the MAV_CMD_IMAGE_START_CAPTURE command. Each time an image is captured, a CAMERA_IMAGE_CAPTURED message is sent back to the GCS.

The `CAMERA_IMAGE_CAPTURED` message not only tells the GCS the image was captured, it is also intended for geo-tagging.

The capture command can be used to request one single image capture or a time lapse. If the command is set to take more than one single image, the GCS might use the MAV_CMD_IMAGE_STOP_CAPTURE command to stop it.

Video Capture

Just like for image capture, if the `flags` field of the CAMERA_INFORMATION message has the CAMERA_CAP_FLAGS_CAPTURE_VIDEO bit set, it will indicate the GCS can send the video capture command to the camera.

To start recording videos, the GCS uses the MAV_CMD_VIDEO_START_CAPTURE command. If requested, the CAMERA_CAPTURE_STATUS message is sent to the GCS at a set interval.

To stop recording, the GCS uses the MAV_CMD_VIDEO_STOP_CAPTURE command.

Message Summary

Message	Description
MAV_CMD_REQUEST_CAMERA_INFORMATION	Send command to request CAMERA_INFORMATION.
CAMERA_INFORMATION	Basic information about camera including URI link to extended information (`cam_definition_uri` field).
CAMERA_CAP_FLAGS	Camera capability flags (Bitmap). For example: ability to capture images in video mode, support for survey mode etc.
MAV_CMD_REQUEST_CAMERA_SETTINGS	Send command to request CAMERA_SETTINGS.
CAMERA_SETTINGS	Timestamp and camera mode information.
MAV_CMD_SET_CAMERA_MODE	Send command to set CAMERA_MODE.
CAMERA_MODE	Camera mode (image, video, survey etc.)
MAV_CMD_REQUEST_STORAGE_INFORMATION	Send command to request STORAGE_INFORMATION.
STORAGE_INFORMATION	Storage information (e.g. number and type of storage devices, total/used/available capacity, read/write speeds).
MAV_CMD_STORAGE_FORMAT	Send command to format the specified storage device.
MAV_CMD_REQUEST_CAMERA_CAPTURE_STATUS	Send command to request CAMERA_CAPTURE_STATUS.
CAMERA_CAPTURE_STATUS	Camera capture status, including current capture type (if any), capture interval, available capacity.
MAV_CMD_IMAGE_START_CAPTURE	Send command to start image capture, specifying the duration between captures and total number of images to capture.
MAV_CMD_IMAGE_STOP_CAPTURE	Send command to stop image capture.
CAMERA_IMAGE_CAPTURED	Information about image captured (returned to GPS every time an image is captured).
MAV_CMD_VIDEO_START_CAPTURE	Send command to start video capture, specifying the frequency that CAMERA_CAPTURE_STATUS messages should be sent while recording.
MAV_CMD_VIDEO_STOP_CAPTURE	Send command to stop video capture.
CAMERA_IMAGE_CAPTURED	Information about image captured (returned to GPS every time an image is captured).

Camera Definition File

A GCS will build a Camera Controller UI for image and video capture using information provided by the CAMERA_INFORMATION message. For very simple cameras, the information in the CAMERA_INFORMATION message itself is sufficient to construct the UI. For more complicated cameras (with settings and options) the information required to build the UI must be supplied in a *Camera Definition File* that is located at the URI specified in the message's `cam_definition_uri` field.

The *Camera Definition File* contains all the camera settings, the options for each setting, and exclusion lists (options that invalidate or are conditional on other settings). In addition, it may contain localisations of GUI strings for display to the user.

At the bottom of this page, you can find a full example of a *Camera Definition File*.

> A *Camera Definition File* is required because the camera options differ so greatly between cameras. It is not reasonable to create specific MAVLink messages for each and every possible option and to tell the GCS the valid options for each camera setting.

Schema

The XML file has 3 main sections (elements):

- Definition
- Parameters
- Localization

Definition

All fields are self explanatory:

```
<definition version="1">
    <model>T100</model>
    <vendor>Foo Industries</vendor>
</definition>
```

Parameters

An extended set of parameter messages is used to define settings and options. These minimally have a parameter name, type and default value (types can be predefined or arbitrary - though arbitrary types are only supported by custom camera controllers). They will also have a description that is displayed to the user and the set of possible options.

Parameters can be simple or quite complex, depending on the behavior they change.

Parameter Types

The type of the parameter follows the enum MAV_PARAM_EXT_TYPE. Within the XML file, these are defined as:

- bool (internally treated as a uint8)
- uint8
- int8
- uint16
- int16
- uint32
- int32

- uint64
- int64
- float
- double
- custom

The `custom` type is a special case that allows for arbitrary data structures of up to 128 bytes. However these are not supported by default - you would need to extend or write your own camera controller within the GCS to interpret this type.

Parameter Definition

The simplest parameter would be a boolean type, which inherently (and automatically) only provides two options (on/off):

```
<parameter name="CAM_IRLOCK" type="bool" default="0">
    <description>Enable IR Lock</description>
</parameter>
```

The `name` attribute is the name of the parameter. This is the name used when requesting or setting the parameter's value using the extended parameter messages. The `description` is what is shown to the user.

More common are parameters that provide options:

```
<parameter name="CAM_WBMODE" type="uint32" default="0">
    <description>White Balance Mode</description>
    <options>
        <option name="Auto" value="0" />
        <option name="Incandescent" value="1" />
        <option name="Sunset" value="3" />
        <option name="Sunny" value="4" />
        <option name="Cloudy" value="5" />
        <option name="Fluorescent" value="7" />
    </options>
</parameter>
```

In this case, the GCS will automatically build a drop down list with the options defined within the `options` group. When sending/receiving the options, the `value` field is used and it is not in any way interpreted by the GCS. The `name` field is used for display only. In other words, using the example above, when the user selects *Sunset*, the GCS will send a PARAM_EXT_SET message with the id `CAM_WBMODE` and a uint32 value of 3.

Exclusion Rules

Some parameters are only relevant when some other parameter is set to some specific option. For example, shutter speed and ISO would only be available when the camera is set to *manual* exposure mode and not shown when the camera is set to *auto* exposure mode. To specify this behavior, you would use the `exclusion` element:

```
<parameter name="CAM_EXPMODE" type="uint32" default="0">
    <description>Exposure Mode</description>
    <options default="0">
        <option name="Auto" value="0">
            <exclusions>
                <exclude>CAM_ISO</exclude>
                <exclude>CAM_SHUTTERSPD</exclude>
            </exclusions>
        </option>
        <option name="Manual" value="1">
            <exclusions>
                <exclude>CAM_EV</exclude>
            </exclusions>
        </option>
    </options>
</parameter>
```

The above example describes an *Exposure Mode* parameter and its two options: *Auto* and *Manual*. When the option is set to *Auto*, both the `CAM_ISO` and `CAM_SHUTTERSPD` parameters (defined elsewhere in the parameter list) are hidden from the UI as they are not applicable. On the other hand, if the option is set to *Manual*, the `CAM_EV` parameter is hidden as it is not applicable while the camera is in *Manual Exposure Mode*.

Required Option Updates

There are cases where an option change requires a parameter to be updated. For example, using the example above, when the camera is set to *Auto Exposure Mode*, it internally might change the ISO and Shutter speed. When the user switches back to *Manual Exposure Mode*, the GCS must request an update for the current ISO and Shutter speed as they may have changed. To do this, you would use the `update` element:

```
<parameter name="CAM_EXPMODE" type="uint32" default="0">
    <description>Exposure Mode</description>
    <updates>
        <update>CAM_SHUTTERSPD</update>
        <update>CAM_ISO</update>
    </updates>
    <options default="0">
        <option name="Auto" value="0">
            <exclusions>
                <exclude>CAM_ISO</exclude>
                <exclude>CAM_SHUTTERSPD</exclude>
            </exclusions>
        </option>
        <option name="Manual" value="1">
            <exclusions>
                <exclude>CAM_EV</exclude>
            </exclusions>
        </option>
    </options>
</parameter>
```

This tells the GCS that when the `CAM_EXPMODE` parameter changes, both the `CAM_SHUTTERSPD` and the `CAM_ISO` must be updated (requested from the camera).

Range Limit

Suppose your camera has the following ISO options:

```
<parameter name="CAM_ISO" type="uint32" default="100">
    <description>ISO</description>
    <options>
        <option name="50" value="50" />
        <option name="100" value="100" />
        <option name="150" value="150" />
        <option name="200" value="200" />
        <option name="300" value="300" />
        <option name="400" value="400" />
        <option name="600" value="600" />
        <option name="800" value="800" />
        <option name="1600" value="1600" />
        <option name="3200" value="3200" />
        <option name="6400" value="6400" />
    </options>
</parameter>
```

But this full range is only available when in *Photo Mode*. For whatever reason, when the camera is set to *Video Mode*, only a subset of the above range is valid. In this case, you would use the `parameterrange` element:

```
<parameter name="CAM_MODE" type="uint32" default="1" control="0">
    <description>Camera Mode</description>
    <options>
        <option name="Photo" value="0" />
        <option name="Video" value="1">
            <parameterranges>
                <parameterrange parameter="CAM_ISO" condition="CAM_EXPMODE=1">
                    <roption name="100" value="100" />
                    <roption name="150" value="150" />
                    <roption name="200" value="200" />
                    <roption name="300" value="300" />
                    <roption name="400" value="400" />
                    <roption name="600" value="600" />
                    <roption name="800" value="800" />
                    <roption name="1600" value="1600" />
                    <roption name="3200" value="3200" />
                </parameterrange>
            </parameterranges>
        </option>
    </options>
</parameter>
```

This indicates to the GCS that when the `CAM_MODE` parameter is set to *Video*, only the given range for the `CAM_ISO` parameter is valid. It additionally gives a condition that this is only the case when the `CAM_EXPOSURE` mode is set to *Manual* (1).

This example also tells the GCS not to display this parameter to the user (`control="0"`). Camera Mode is a standard parameter defined in the CAMERA_INFORMATION message and it's handled by the GCS in that way. The parameter definition above was created in order to tell the GCS the rules that are applied when changes to the mode occur.

Localization

The `localization` element is used for defining localized strings for display to users. If found, the GCS will use to replace all `description` and options `name` values found in the file with the strings defined here. Here is an example for German localization (de_DE):

```
<localization>
    <locale name="de_DE">
        <strings original="Camera Mode" translated="Kamera Modus" />
        <strings original="Photo" translated="Foto" />
        <strings original="Video" translated="Video" />
        <strings original="White Balance Mode" translated="Weißabgleich Modus" />
        <strings original="Auto" translated="Auto" />
        <strings original="Incandescent" translated="Glühlampen" />
        <strings original="Sunset" translated="Sonnenuntergang" />
        <strings original="Sunny" translated="Sonnig" />
        <strings original="Cloudy" translated="Bewölkt" />
        <strings original="Fluorescent" translated="Fluoreszierende" />
    </locale>
</localization>
```

When the GCS loads and parses the XML file, it will check and see if it can find a localized version appropriate to the system language. If it finds a localisation, it will proceed to replace all occurrences of `original` with `translated` . If something is not found, the default English string is used. You can have as many locales as deemed necessary.

Protocol Definition

Once the Camera Definition File is loaded by the GCS, it will request all parameters from the camera using the PARAM_EXT_REQUEST_LIST message. In response, the camera will send back all parameters using the PARAM_EXT_VALUE message.

When the user makes a selection, the GCS will send the new option using the PARAM_EXT_SET message and it will expect in response a PARAM_EXT_ACK message.

When the GCS requires a current option for a given parameter, it will use the PARAM_EXT_REQUEST_READ message and it will expect in response a PARAM_EXT_VALUE message.

Full Camera Definition File Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<mavlinkcamera>
    <definition version="7">
        <model>T100</model>
        <vendor>Foo Industries</vendor>
    </definition>
    <parameters>
        <!-- control = 0 tells us this should not create an automatic UI control -->
        <parameter name="CAM_MODE" type="uint32" default="1" control="0">
            <description>Camera Mode</description>
            <!-- This tells us when this parameter changes, these parameters must be updated (requested)-->
            <updates>
                <update>CAM_SHUTTERSPD</update>
                <update>CAM_ISO</update>
                <update>CAM_VIDRES</update>
                <update>CAM_ASPECTRATIO</update>
            </updates>
            <options>
                <option name="Photo" value="0">
                    <!-- This tells us when Camera Mode is set to Photo mode, the following parameters should be igno
red (hidden from UI or disabled)-->
                    <exclusions>
                        <exclude>CAM_VIDRES</exclude>
                        <exclude>CAM_VIDFMT</exclude>
                    </exclusions>
                </option>
                <option name="Video" value="1">
                    <!-- Conversely, when Camera Mode is set to Photo mode, the following parameters should be ignore
d (hidden from UI or disabled)-->
                    <exclusions>
                        <exclude>CAM_PHOTOFMT</exclude>
                        <exclude>CAM_PHOTOQUAL</exclude>
                        <exclude>CAM_COLORMODE</exclude>
                    </exclusions>
                    <parameterranges>
                        <parameterrange parameter="CAM_ISO" condition="CAM_EXPMODE=1">
                            <roption name="100" value="100" />
                            <roption name="150" value="150" />
                            <roption name="200" value="200" />
                            <roption name="300" value="300" />
                            <roption name="400" value="400" />
                            <roption name="600" value="600" />
                            <roption name="800" value="800" />
                            <roption name="1600" value="1600" />
                            <roption name="3200" value="3200" />
                        </parameterrange>
                    </parameterranges>
                </option>
            </options>
        </parameter>
        <parameter name="CAM_WBMODE" type="uint32" default="0">
            <description>White Balance Mode</description>
            <options>
                <option name="Auto" value="0" />
                <option name="Incandescent" value="1" />
                <option name="Sunset" value="3" />
                <option name="Sunny" value="4" />
                <option name="Cloudy" value="5" />
                <option name="Fluorescent" value="7" />
            </options>
```

```
        </parameter>
        <parameter name="CAM_EXPMODE" type="uint32" default="0">
            <description>Exposure Mode</description>
            <updates>
                <update>CAM_SHUTTERSPD</update>
                <update>CAM_ISO</update>
            </updates>
            <options default="0">
                <option name="Auto" value="0">
                    <exclusions>
                        <exclude>CAM_ISO</exclude>
                        <exclude>CAM_SHUTTERSPD</exclude>
                    </exclusions>
                </option>
                <option name="Manual" value="1">
                    <exclusions>
                        <exclude>CAM_EV</exclude>
                    </exclusions>
                </option>
            </options>
        </parameter>
        <parameter name="CAM_SHUTTERSPD" type="float" default="0.016666">
            <description>Shutter Speed</description>
            <options>
                <option name="4" value="4" />
                <option name="3" value="3" />
                <option name="2" value="2" />
                <option name="1" value="1" />
                <option name="1/30" value="0.033333" />
                <option name="1/60" value="0.016666" />
                <option name="1/125" value="0.008" />
                <option name="1/250" value="0.004" />
                <option name="1/500" value="0.002" />
                <option name="1/1000" value="0.001" />
                <option name="1/2000" value="0.0005" />
                <option name="1/4000" value="0.00025" />
                <option name="1/8000" value="0.000125" />
            </options>
        </parameter>
        <parameter name="CAM_ISO" type="uint32" default="100">
            <description>ISO</description>
            <options>
                <option name="100" value="100" />
                <option name="150" value="150" />
                <option name="200" value="200" />
                <option name="300" value="300" />
                <option name="400" value="400" />
                <option name="600" value="600" />
                <option name="800" value="800" />
                <option name="1600" value="1600" />
                <option name="3200" value="3200" />
                <option name="6400" value="6400" />
            </options>
        </parameter>
        <parameter name="CAM_EV" type="float" default="0">
            <description>Exposure Compensation</description>
            <options>
                <option name="-3" value="-3" />
                <option name="-2.5" value="-2.5" />
                <option name="-2" value="-2" />
                <option name="-1.5" value="-1.5" />
                <option name="-1" value="-1" />
                <option name="-0.5" value="-0.5" />
                <option name="0" value="0" />
                <option name="+0.5" value="0.5" />
                <option name="+1" value="1" />
                <option name="+1.5" value="1.5" />
                <option name="+2" value="2" />
                <option name="+2.5" value="2.5" />
                <option name="+3" value="3" />
            </options>
        </parameter>
```

```
        <parameter name="CAM_VIDRES" type="uint32" default="0">
            <description>Video Resolution</description>
            <updates>
                <update>CAM_SHUTTERSPD</update>
                <update>CAM_ISO</update>
                <update>CAM_ASPECTRATIO</update>
            </updates>
            <options>
                <!-- 4096 x 2160 -->
                <option name="4096 x 2160 60fps (UHD)" value="0">
                    <exclusions>
                        <exclude>CAM_VIDFMT</exclude>
                    </exclusions>
                    <parameterranges>
                        <!-- When Camera Mode is Video and Exposure Mode is Manual, Shutter Speed cannot be slower th
an the frame rate -->
                        <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                            <roption name="1/60" value="0.016666" />
                            <roption name="1/125" value="0.008" />
                            <roption name="1/250" value="0.004" />
                            <roption name="1/500" value="0.002" />
                            <roption name="1/1000" value="0.001" />
                            <roption name="1/2000" value="0.0005" />
                            <roption name="1/4000" value="0.00025" />
                            <roption name="1/8000" value="0.000125" />
                        </parameterrange>
                    </parameterranges>
                </option>
                <option name="4096 x 2160 50fps (UHD)" value="1">
                    <exclusions>
                        <exclude>CAM_VIDFMT</exclude>
                    </exclusions>
                    <parameterranges>
                        <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                            <roption name="1/60" value="0.016666" />
                            <roption name="1/125" value="0.008" />
                            <roption name="1/250" value="0.004" />
                            <roption name="1/500" value="0.002" />
                            <roption name="1/1000" value="0.001" />
                            <roption name="1/2000" value="0.0005" />
                            <roption name="1/4000" value="0.00025" />
                            <roption name="1/8000" value="0.000125" />
                        </parameterrange>
                    </parameterranges>
                </option>
                <option name="4096 x 2160 48fps (UHD)" value="2">
                    <exclusions>
                        <exclude>CAM_VIDFMT</exclude>
                    </exclusions>
                    <parameterranges>
                        <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                            <roption name="1/60" value="0.016666" />
                            <roption name="1/125" value="0.008" />
                            <roption name="1/250" value="0.004" />
                            <roption name="1/500" value="0.002" />
                            <roption name="1/1000" value="0.001" />
                            <roption name="1/2000" value="0.0005" />
                            <roption name="1/4000" value="0.00025" />
                            <roption name="1/8000" value="0.000125" />
                        </parameterrange>
                    </parameterranges>
                </option>
                <option name="4096 x 2160 30fps (UHD)" value="3">
                    <parameterranges>
                        <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                            <roption name="1/30" value="0.033333" />
                            <roption name="1/60" value="0.016666" />
                            <roption name="1/125" value="0.008" />
                            <roption name="1/250" value="0.004" />
                            <roption name="1/500" value="0.002" />
                            <roption name="1/1000" value="0.001" />
                            <roption name="1/2000" value="0.0005" />
```

```
                <roption name="1/4000" value="0.00025" />
                <roption name="1/8000" value="0.000125" />
            </parameterrange>
        </parameterranges>
    </option>
    <option name="4096 x 2160 25fps (UHD)" value="4">
        <parameterranges>
            <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                <roption name="1/30" value="0.033333" />
                <roption name="1/60" value="0.016666" />
                <roption name="1/125" value="0.008" />
                <roption name="1/250" value="0.004" />
                <roption name="1/500" value="0.002" />
                <roption name="1/1000" value="0.001" />
                <roption name="1/2000" value="0.0005" />
                <roption name="1/4000" value="0.00025" />
                <roption name="1/8000" value="0.000125" />
            </parameterrange>
        </parameterranges>
    </option>
    <option name="4096 x 2160 24fps (UHD)" value="5">
        <parameterranges>
            <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                <roption name="1/30" value="0.033333" />
                <roption name="1/60" value="0.016666" />
                <roption name="1/125" value="0.008" />
                <roption name="1/250" value="0.004" />
                <roption name="1/500" value="0.002" />
                <roption name="1/1000" value="0.001" />
                <roption name="1/2000" value="0.0005" />
                <roption name="1/4000" value="0.00025" />
                <roption name="1/8000" value="0.000125" />
            </parameterrange>
        </parameterranges>
    </option>
    <!-- 3840 x 2160 -->
    <option name="3840 x 2160 60fps (UHD)" value="6">
        <exclusions>
            <exclude>CAM_VIDFMT</exclude>
        </exclusions>
        <parameterranges>
            <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                <roption name="1/60" value="0.016666" />
                <roption name="1/125" value="0.008" />
                <roption name="1/250" value="0.004" />
                <roption name="1/500" value="0.002" />
                <roption name="1/1000" value="0.001" />
                <roption name="1/2000" value="0.0005" />
                <roption name="1/4000" value="0.00025" />
                <roption name="1/8000" value="0.000125" />
            </parameterrange>
        </parameterranges>
    </option>
    <option name="3840 x 2160 50fps (UHD)" value="7">
        <exclusions>
            <exclude>CAM_VIDFMT</exclude>
        </exclusions>
        <parameterranges>
            <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                <roption name="1/60" value="0.016666" />
                <roption name="1/125" value="0.008" />
                <roption name="1/250" value="0.004" />
                <roption name="1/500" value="0.002" />
                <roption name="1/1000" value="0.001" />
                <roption name="1/2000" value="0.0005" />
                <roption name="1/4000" value="0.00025" />
                <roption name="1/8000" value="0.000125" />
            </parameterrange>
        </parameterranges>
    </option>
    <option name="3840 x 2160 48fps (UHD)" value="8">
        <exclusions>
```

```
                    <exclude>CAM_VIDFMT</exclude>
                </exclusions>
                <parameterranges>
                    <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                        <roption name="1/60" value="0.016666" />
                        <roption name="1/125" value="0.008" />
                        <roption name="1/250" value="0.004" />
                        <roption name="1/500" value="0.002" />
                        <roption name="1/1000" value="0.001" />
                        <roption name="1/2000" value="0.0005" />
                        <roption name="1/4000" value="0.00025" />
                        <roption name="1/8000" value="0.000125" />
                    </parameterrange>
                </parameterranges>
            </option>
            <option name="3840 x 2160 30fps (UHD)" value="9">
                <parameterranges>
                    <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                        <roption name="1/30" value="0.033333" />
                        <roption name="1/60" value="0.016666" />
                        <roption name="1/125" value="0.008" />
                        <roption name="1/250" value="0.004" />
                        <roption name="1/500" value="0.002" />
                        <roption name="1/1000" value="0.001" />
                        <roption name="1/2000" value="0.0005" />
                        <roption name="1/4000" value="0.00025" />
                        <roption name="1/8000" value="0.000125" />
                    </parameterrange>
                </parameterranges>
            </option>
            <option name="3840 x 2160 25fps (UHD)" value="10">
                <parameterranges>
                    <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                        <roption name="1/30" value="0.033333" />
                        <roption name="1/60" value="0.016666" />
                        <roption name="1/125" value="0.008" />
                        <roption name="1/250" value="0.004" />
                        <roption name="1/500" value="0.002" />
                        <roption name="1/1000" value="0.001" />
                        <roption name="1/2000" value="0.0005" />
                        <roption name="1/4000" value="0.00025" />
                        <roption name="1/8000" value="0.000125" />
                    </parameterrange>
                </parameterranges>
            </option>
            <option name="3840 x 2160 24fps (UHD)" value="11">
                <parameterranges>
                    <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                        <roption name="1/30" value="0.033333" />
                        <roption name="1/60" value="0.016666" />
                        <roption name="1/125" value="0.008" />
                        <roption name="1/250" value="0.004" />
                        <roption name="1/500" value="0.002" />
                        <roption name="1/1000" value="0.001" />
                        <roption name="1/2000" value="0.0005" />
                        <roption name="1/4000" value="0.00025" />
                        <roption name="1/8000" value="0.000125" />
                    </parameterrange>
                </parameterranges>
            </option>
            <!-- 2720 x 1530 -->
            <option name="2720 x 1530 60fps (UHD)" value="12">
                <parameterranges>
                    <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                        <roption name="1/60" value="0.016666" />
                        <roption name="1/125" value="0.008" />
                        <roption name="1/250" value="0.004" />
                        <roption name="1/500" value="0.002" />
                        <roption name="1/1000" value="0.001" />
                        <roption name="1/2000" value="0.0005" />
                        <roption name="1/4000" value="0.00025" />
                        <roption name="1/8000" value="0.000125" />
```

```
                </parameterrange>
            </parameterranges>
        </option>
        <option name="2720 x 1530 48fps (UHD)" value="13">
            <parameterranges>
                <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                    <roption name="1/60" value="0.016666" />
                    <roption name="1/125" value="0.008" />
                    <roption name="1/250" value="0.004" />
                    <roption name="1/500" value="0.002" />
                    <roption name="1/1000" value="0.001" />
                    <roption name="1/2000" value="0.0005" />
                    <roption name="1/4000" value="0.00025" />
                    <roption name="1/8000" value="0.000125" />
                </parameterrange>
            </parameterranges>
        </option>
        <option name="2720 x 1530 30fps (UHD)" value="14">
            <parameterranges>
                <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                    <roption name="1/30" value="0.033333" />
                    <roption name="1/60" value="0.016666" />
                    <roption name="1/125" value="0.008" />
                    <roption name="1/250" value="0.004" />
                    <roption name="1/500" value="0.002" />
                    <roption name="1/1000" value="0.001" />
                    <roption name="1/2000" value="0.0005" />
                    <roption name="1/4000" value="0.00025" />
                    <roption name="1/8000" value="0.000125" />
                </parameterrange>
            </parameterranges>
        </option>
        <option name="2720 x 1530 24fps (UHD)" value="15">
            <parameterranges>
                <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                    <roption name="1/30" value="0.033333" />
                    <roption name="1/60" value="0.016666" />
                    <roption name="1/125" value="0.008" />
                    <roption name="1/250" value="0.004" />
                    <roption name="1/500" value="0.002" />
                    <roption name="1/1000" value="0.001" />
                    <roption name="1/2000" value="0.0005" />
                    <roption name="1/4000" value="0.00025" />
                    <roption name="1/8000" value="0.000125" />
                </parameterrange>
            </parameterranges>
        </option>
        <!-- 1920 x 1080 -->
        <option name="1920 x 1080 120fps (FHD)" value="16">
            <parameterranges>
                <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                    <roption name="1/125" value="0.008" />
                    <roption name="1/250" value="0.004" />
                    <roption name="1/500" value="0.002" />
                    <roption name="1/1000" value="0.001" />
                    <roption name="1/2000" value="0.0005" />
                    <roption name="1/4000" value="0.00025" />
                    <roption name="1/8000" value="0.000125" />
                </parameterrange>
            </parameterranges>
        </option>
        <option name="1920 x 1080 60fps (FHD)" value="17">
            <parameterranges>
                <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                    <roption name="1/60" value="0.016666" />
                    <roption name="1/125" value="0.008" />
                    <roption name="1/250" value="0.004" />
                    <roption name="1/500" value="0.002" />
                    <roption name="1/1000" value="0.001" />
                    <roption name="1/2000" value="0.0005" />
                    <roption name="1/4000" value="0.00025" />
                    <roption name="1/8000" value="0.000125" />
```

```
                    </parameterrange>
                </parameterranges>
            </option>
            <option name="1920 x 1080 50fps (FHD)" value="18">
                <parameterranges>
                    <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                        <roption name="1/60" value="0.016666" />
                        <roption name="1/125" value="0.008" />
                        <roption name="1/250" value="0.004" />
                        <roption name="1/500" value="0.002" />
                        <roption name="1/1000" value="0.001" />
                        <roption name="1/2000" value="0.0005" />
                        <roption name="1/4000" value="0.00025" />
                        <roption name="1/8000" value="0.000125" />
                    </parameterrange>
                </parameterranges>
            </option>
            <option name="1920 x 1080 48fps (FHD)" value="19">
                <parameterranges>
                    <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                        <roption name="1/60" value="0.016666" />
                        <roption name="1/125" value="0.008" />
                        <roption name="1/250" value="0.004" />
                        <roption name="1/500" value="0.002" />
                        <roption name="1/1000" value="0.001" />
                        <roption name="1/2000" value="0.0005" />
                        <roption name="1/4000" value="0.00025" />
                        <roption name="1/8000" value="0.000125" />
                    </parameterrange>
                </parameterranges>
            </option>
            <option name="1920 x 1080 30fps (FHD)" value="20">
                <parameterranges>
                    <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                        <roption name="1/30" value="0.033333" />
                        <roption name="1/60" value="0.016666" />
                        <roption name="1/125" value="0.008" />
                        <roption name="1/250" value="0.004" />
                        <roption name="1/500" value="0.002" />
                        <roption name="1/1000" value="0.001" />
                        <roption name="1/2000" value="0.0005" />
                        <roption name="1/4000" value="0.00025" />
                        <roption name="1/8000" value="0.000125" />
                    </parameterrange>
                </parameterranges>
            </option>
            <option name="1920 x 1080 25fps (FHD)" value="21">
                <parameterranges>
                    <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                        <roption name="1/30" value="0.033333" />
                        <roption name="1/60" value="0.016666" />
                        <roption name="1/125" value="0.008" />
                        <roption name="1/250" value="0.004" />
                        <roption name="1/500" value="0.002" />
                        <roption name="1/1000" value="0.001" />
                        <roption name="1/2000" value="0.0005" />
                        <roption name="1/4000" value="0.00025" />
                        <roption name="1/8000" value="0.000125" />
                    </parameterrange>
                </parameterranges>
            </option>
            <option name="1920 x 1080 24fps (FHD)" value="22">
                <parameterranges>
                    <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                        <roption name="1/30" value="0.033333" />
                        <roption name="1/60" value="0.016666" />
                        <roption name="1/125" value="0.008" />
                        <roption name="1/250" value="0.004" />
                        <roption name="1/500" value="0.002" />
                        <roption name="1/1000" value="0.001" />
                        <roption name="1/2000" value="0.0005" />
                        <roption name="1/4000" value="0.00025" />
```

```
                <roption name="1/8000" value="0.000125" />
            </parameterrange>
        </parameterranges>
    </option>
    <!-- 1280 x 720 -->
    <option name="1280 x 720 120fps (HD)" value="23">
        <parameterranges>
            <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                <roption name="1/125" value="0.008" />
                <roption name="1/250" value="0.004" />
                <roption name="1/500" value="0.002" />
                <roption name="1/1000" value="0.001" />
                <roption name="1/2000" value="0.0005" />
                <roption name="1/4000" value="0.00025" />
                <roption name="1/8000" value="0.000125" />
            </parameterrange>
        </parameterranges>
    </option>
    <option name="1280 x 720 60fps (HD)" value="24">
        <parameterranges>
            <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                <roption name="1/60" value="0.016666" />
                <roption name="1/125" value="0.008" />
                <roption name="1/250" value="0.004" />
                <roption name="1/500" value="0.002" />
                <roption name="1/1000" value="0.001" />
                <roption name="1/2000" value="0.0005" />
                <roption name="1/4000" value="0.00025" />
                <roption name="1/8000" value="0.000125" />
            </parameterrange>
        </parameterranges>
    </option>
    <option name="1280 x 720 48fps (HD)" value="25">
        <parameterranges>
            <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                <roption name="1/60" value="0.016666" />
                <roption name="1/125" value="0.008" />
                <roption name="1/250" value="0.004" />
                <roption name="1/500" value="0.002" />
                <roption name="1/1000" value="0.001" />
                <roption name="1/2000" value="0.0005" />
                <roption name="1/4000" value="0.00025" />
                <roption name="1/8000" value="0.000125" />
            </parameterrange>
        </parameterranges>
    </option>
    <option name="1280 x 720 30fps (HD)" value="26">
        <parameterranges>
            <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                <roption name="1/30" value="0.033333" />
                <roption name="1/60" value="0.016666" />
                <roption name="1/125" value="0.008" />
                <roption name="1/250" value="0.004" />
                <roption name="1/500" value="0.002" />
                <roption name="1/1000" value="0.001" />
                <roption name="1/2000" value="0.0005" />
                <roption name="1/4000" value="0.00025" />
                <roption name="1/8000" value="0.000125" />
            </parameterrange>
        </parameterranges>
    </option>
    <option name="1280 x 720 24fps (HD)" value="27">
        <parameterranges>
            <parameterrange parameter="CAM_SHUTTERSPD" condition="CAM_MODE=1 AND CAM_EXPMODE=1">
                <roption name="1/30" value="0.033333" />
                <roption name="1/60" value="0.016666" />
                <roption name="1/125" value="0.008" />
                <roption name="1/250" value="0.004" />
                <roption name="1/500" value="0.002" />
                <roption name="1/1000" value="0.001" />
                <roption name="1/2000" value="0.0005" />
                <roption name="1/4000" value="0.00025" />
```

```
                        <roption name="1/8000" value="0.000125" />
                    </parameterrange>
                </parameterranges>
            </option>
        </options>
    </parameter>
    <parameter name="CAM_VIDFMT" type="uint32" default="0">
        <description>Video Format</description>
        <updates>
            <update>CAM_SHUTTERSPD</update>
            <update>CAM_ISO</update>
            <update>CAM_VIDRES</update>
        </updates>
        <options>
            <option name="H264" value="1" />
            <option name="HEVC" value="3">
                <parameterranges>
                    <!-- When Mode is HEVC, 4K res limit is 30fps -->
                    <parameterrange parameter="CAM_VIDRES" condition="CAM_MODE=1">
                        <roption name="4096 x 2160 30fps (UHD)" value="3" />
                        <roption name="4096 x 2160 25fps (UHD)" value="4" />
                        <roption name="4096 x 2160 24fps (UHD)" value="5" />
                        <roption name="3840 x 2160 30fps (UHD)" value="9" />
                        <roption name="3840 x 2160 25fps (UHD)" value="10" />
                        <roption name="3840 x 2160 24fps (UHD)" value="11" />
                        <roption name="2720 x 1530 60fps (UHD)" value="12" />
                        <roption name="2720 x 1530 48fps (UHD)" value="13" />
                        <roption name="2720 x 1530 30fps (UHD)" value="14" />
                        <roption name="2720 x 1530 24fps (UHD)" value="15" />
                        <roption name="1920 x 1080 120fps (FHD)" value="16" />
                        <roption name="1920 x 1080 60fps (FHD)" value="17" />
                        <roption name="1920 x 1080 50fps (FHD)" value="18" />
                        <roption name="1920 x 1080 48fps (FHD)" value="19" />
                        <roption name="1920 x 1080 30fps (FHD)" value="20" />
                        <roption name="1920 x 1080 25fps (FHD)" value="21" />
                        <roption name="1920 x 1080 24fps (FHD)" value="22" />
                        <roption name="1280 x 720 120fps (HD)" value="23" />
                        <roption name="1280 x 720 60fps (HD)" value="24" />
                        <roption name="1280 x 720 48fps (HD)" value="25" />
                        <roption name="1280 x 720 30fps (HD)" value="26" />
                        <roption name="1280 x 720 24fps (HD)" value="27" />
                    </parameterrange>
                </parameterranges>
            </option>
        </options>
    </parameter>
    <parameter name="CAM_COLORMODE" type="uint32" default="1">
        <description>Color Mode</description>
        <options>
            <option name="Neutral" value="0" />
            <option name="Enhanced" value="1" />
            <option name="Night" value="3" />
            <option name="Unprocessed" value="2" />
        </options>
    </parameter>
    <parameter name="CAM_PHOTOFMT" type="uint32" default="0">
        <description>Image Format</description>
        <options>
            <option name="Jpeg" value="0" />
            <option name="Raw" value="1" />
            <option name="Jpeg+Raw" value="2" />
        </options>
    </parameter>
    <parameter name="CAM_PHOTOQUAL" type="uint32" default="1">
        <description>Image Quality</description>
        <options>
            <option name="Low" value="0" />
            <option name="Medium" value="1" />
            <option name="High" value="2" />
            <option name="Ultra" value="3" />
        </options>
    </parameter>
```

```
    </parameters>
    <localization>
        <!-- If no appropriate locale is found, the original (above) will be used -->
        <!-- At runtime, the code will go through every "description" and "option name" looking for "original" and re
place it with "translated" -->
        <locale name="de_DE">
            <strings original="Camera Mode" translated="Kamera Modus" />
            <strings original="Photo" translated="Foto" />
            <strings original="White Balance Mode" translated="Weißabgleich Modus" />
            <strings original="Incandescent" translated="Glühlampen" />
            <strings original="Sunset" translated="Sonnenuntergang" />
            <strings original="Sunny" translated="Sonnig" />
            <strings original="Cloudy" translated="Bewölkt" />
            <strings original="Fluorescent" translated="Fluoreszierende" />
            <strings original="Lock" translated="Sperre" />
            <strings original="Exposure Mode" translated="Belichtungsmodus" />
            <strings original="Manual" translated="Manuell" />
            <strings original="Shutter Speed" translated="Verschlusszeit" />
            <strings original="Exposure Compensation" translated="Belichtungskorrektur" />
            <strings original="Video Resolution" translated="Videoauflösung" />
            <strings original="Average" translated="Durchschnitt" />
            <strings original="Center" translated="Zentrum" />
            <strings original="Color Mode" translated="Farbmodus" />
            <strings original="Neutral" translated="Neutral" />
            <strings original="Enhanced" translated="Verbessert" />
            <strings original="Night" translated="Nacht" />
            <strings original="Unprocessed" translated="Unverarbeitete" />
            <strings original="Image Format" translated="Bildformat" />
            <strings original="Image Quality" translated="Bildqualität" />
            <strings original="High" translated="Hoch" />
        </locale>
    </localization>
</mavlinkcamera>
```

Gimbal Configuration Protocol

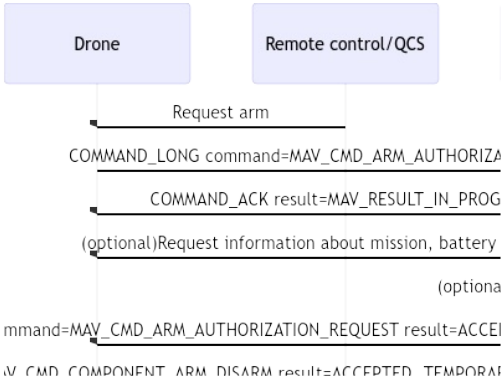
The gimbal configuration message set is using a number of commands and few special-purpose messages to configure a payload mount.

Arm authorization

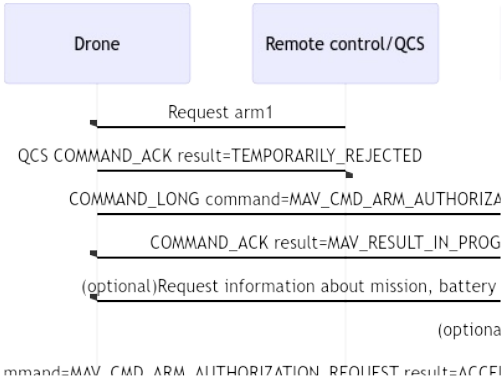
When enabled by setting a parameter on flight stack, the drone will only arm the motors if authorized by a external entity. This external entity is reposible to request any information that it need from the drone and from other souces(example: weather) and authorize or not the arm procedure.

This will be useful to comply to NASA UTM (https://utm.arc.nasa.gov/) but can also be useful for private companies.

Authorization flow



In case the authorizer need a lot of time to get and process the information is better have another authorization flow to avoid arm the drone at unexpected time.



Message parameters:

COMMAND_LONG

```
    command=MAV_CMD_ARM_AUTHORIZATION_REQUEST
    target_system=system id of arm authorizer
    target_component=component id of arm authorizer
```

COMMAND_ACK

```
    command=MAV_CMD_ARM_AUTHORIZATION_REQUEST
    result=ACCEPTED, TEMPORARILY_REJECTED or DENIED
    progress/result_param1=if result is TEMPORARILY_REJECTED or DENIED the reason should be set MAV_ARM_AUTH_DENIED_R
EASON otherwise it should be set as 0
    result_param2=if result is ACCEPTED the it should be set with the time in seconds that this authorization is vali
d otherwise an aditional information about why it was denied should be set. example: for result_param1=MAV_ARM_AUTH_D
ENIED_REASON_INVALID_WAYPOINT or MAV_ARM_AUTH_DENIED_REASON_AIRSPACE_IN_USE it may have the index of the waypoint tha
t caused it to be denied.
    target_system=system id of the drone
    target_component=component id of the drone
```

Image Transmission Protocol

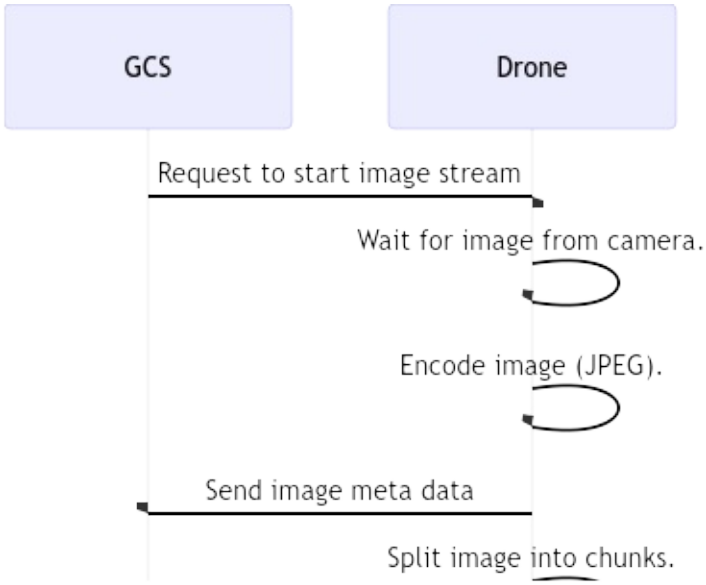
The image transmission protocol uses MAVLink as the communication channel to transport any kind of image (raw images, Kinect data, etc.) from one MAVLink node to another. It basically takes a live camera image, splits it into small chunks and sends it over MAVLink.

This topic describes how the image streaming functionality works and covers both the communication protocol and implementation details (for a vehicle and *QGroundControl*).

> At time of writing (March 2018) the protocol is mainly used to transfer images from a vehicle to *QGroundControl* (to show PX4 Flow images for focusing). The protocol could also be used to send any other file types.

Communication

The image streaming component uses two MAVLink messages: A handshake message, DATA_TRANSMISSION_HANDSHAKE, to initiate, control and stop the image streaming; and a data container message, ENCAPSULATED_DATA, to transport the image data.



1. The communication is initiated by the *QGroundControl* with a request to start the stream. To do so, one must set the following fields in the MAVLink message:
   - `target` : to the ID of the targeted MAV,
   - `state` : to 0 for a request,
   - `id` : an ID for the image stream,

     > For the moment, the image streamer only supports one stream per image type and therefore requires you to set the `id` to the same integer as the `type` field.

   - `type` : any of the types in the enum MAVLINK_DATA_STREAM_TYPE in **mavlink.h**,
   - `freq` : bigger than 0 for "frames per seconds", lower than 0 for "seconds per frame"

It is possible to request for a specific image quality. To do so, you must set the "quality" field. All other fields should be zero in the initial request.

1. When the targeted MAV receives the handshake request, it sends back an acknowledgment and starts the image stream at the requested framerate. The handshake ACK packet normally contains the same values as requested by the GCS (`state` set to 1, because it's an ACK), and adds data about the size of the next sent image:
   - The field `packets` contains the number of MAVLink `ENCAPSULATED_DATA` packets,
   - the field `payload` specifies the size of the payload of each data packet (normally 252 bytes),
   - and the `size` field specifies the image size in bytes.
2. The image data is then split into chunks to fit into normal MAVLink messages. They are then packed into `ENCAPSULATED_DATA` packets and sent over MAVLink. Every packet contains a sequence number as well as the ID of the image stream it belongs to. The image streamer now sends periodically new images, there is no further interaction needed. Every new image comes with a new `DATA_TRANSMISSION_HANDSHAKE` ACK packet with updated image `size`, `packets` and `payload` fields. After this ACK packet, the new image arrives as a series of `ENCAPSULATED_DATA` packets.

   > The sequence number starts at 0 for every new image of the stream.

3. To stop an image stream you must send a new `DATA_TRANSMISSION_HANDSHAKE` request packet with the frequency set to 0. The MAVLink node will acknowledge this by sending back an ACK packet containing the same data as in the request.

Usage / Configuration

To use the two modules on your MAV, you have to do the following steps:

- Compile the `mavconn` middleware for your MAV: Guide, Github.
- Start at least these components on the MAV:

```
px_mavlink_bridge_udp &
px_system_control --heartbeat &
px_camera -o lcm &
```

- Compile and start *QGroundControl*.
- Start the image streaming component (you can add the `-v` flag to see some more output): `px_imagestreamer`.
- Initiate the image stream: Open the HUD widget, right-click into the widget and choose **Enable live Image Streaming**.

You should now be able to see the live video feed with one image per second (default, hardcoded at the moment).

Developer

Out-of-the-box, the image streaming component only implements JPEG streaming of the camera image. To implement your own image stream, you have to do the following:

- Write a MAVLink handler, which handles requests to start image streams of your type of choice.
- Write a data handler, which takes your desired data (i.e. a stereo camera image), encodes it into the format of your choice (i.e. rawimage, JPEG, BMP) and splits/sends the data over MAVLink.
- Extend the data/message handler in the UAS component of *QGroundControl* to correctly handle your data (i.e. unpacking of the chosen format).
- Write or extend a widget to display your data according to your wishes.

Packet Serialization

Attitude Message Example

This first example shows how the MAVLink convenience serialization functions make it simple to send messages over a link.

This is the function definition for the altitude message. Behind the scenes the serializer takes care of encoding the message and sending it out on the serial port.

```
static inline void mavlink_msg_attitude_send(mavlink_channel_t chan,
uint32_t time_boot_ms, float roll, float pitch, float yaw,
float rollspeed, float pitchspeed, float yawspeed);
```

This is the function definition for the altitude message. Behind the scenes the serializer takes care of encoding the message and sending it out on the serial port.

```
def attitude_send(self, usec, roll, pitch, yaw,
rollspeed, pitchspeed, yawspeed):
```

Whatever language you are using, the resulting binary data will be the same:

```
0x55 0x1C 0x1E <time> <roll> <pitch> <yaw>
<rollspeed> <pitchspeed> <yawspeed> <crc1> <crc2>
```

Field Reordering and CRC Extra Calculation

MAVLink uses one extra CRC that is added to the message CRC to detect mismatches in message specifications. This is to prevent two devices using different message versions from incorrectly decoding a message with the same length.

Rationale for a Format Checksum

While MAVLink 0.9 was used there were a small number of incidents where the XML describing a message that was in active use changed. The change was such that the length of the message didn't change, but the fields did. Revision 0.9 did check the correct message length, but not the field order, types or field names. This meant that, when a MAV using the old code was talking to a ground station using updated code, the fields were badly corrupted. The MAVLink 0.9 protocol completely relied on everyone being careful not to change the meaning or format of any existing message. With so many people working on MAVLink this was hard to enforce. So for MAVLink 1.0 this problem was solved by adding a 1 byte 'seed' to the checksum based on the XML for the message.

CRC_EXTRA Calculation

When the MAVLink code generator runs, it takes a checksum of the XML structure for each message and creates an array define `MAVLINK_MESSAGE_CRCS`. This is used to initialise the `mavlink_message_crcs[]` array in the C/C++ implementation, and is similarly used in the python (or any other, such as the C# and JavaScript) implementation. When the checksum for a message is calculated, this extra byte is added on the end of the data that the checksum is calculated over. The result is that if the XML changes then the message will be rejected by the recipient as having an incorrect checksum. This ensures that only messages where the sender and recipient are using the same message structure will get through (or at least it makes a mistake much more unlikely, as for any checksum application). If you are doing your own implementation of

MAVLink 1.0 you can get this checksum in one of two ways: you can just use the generated headers, and use `MAVLINK_MESSAGE_CRCS` to get the right seed for each message type or you can re-implement the code that calculates the seed.

As MAVLink reorders internally the message fields according to their size to prevent word / halfword alignment issues (see Data structure alignment (Wikipedia) for further reference) and a wrongly implemented reordering potentially can cause inconsistencies as well, the `CRC_EXTRA` is calculated based on the internal `struct` and over-the-air message layout, not in the XML order.

Field Reordering (MAVLink 1)

The reordering happens as follows:

- Fields are sorted according to their native data size, first `(u)int64_t` and `double`, then `(u)int32_t`, `float`, `(u)int16_t`, `(u)int8_t`.
- If two fields have the same length, their order is preserved as it was present before the data field size ordering
- Arrays are handled based on the data type they use, not based on the total array size
- The over-the-air order is the same as for the `struct` and thus represents the reordered fields
- The CRC field is calculated AFTER the reordering, to ensure that a mistake during field reordering will be caught by a faulty CRC. The provided Python, C and C# reference implementations are tested to have the correct field reordering, this is only a concern for custom implementations.

> This ordering is unique and can be easily implemented in a protocol generator by using a stable sorting algorithm. The alternative to using sorting would be either to use inefficient alignment, which is bad for the target architectures for typical MAVLink applications, or to have function calls in the order of the variable size instead of the application context. This would lead to very confusing function signatures of serialization functions.

Field Reordering (MAVLink 2)

MAVLink 2 messages order the MAVLink 1 ("base") fields in the same way as the MAVLink 1 protocol. Extension fields in MAVLink 1 messages, and all fields in new MAVLink 2 messages (id>255), are ordered in the same way as the source XML.

Python Code Example

This is the python code that calculates the `CRC_EXTRA` seed:

```
def message_checksum(msg):
    '''calculate a 8-bit checksum of the key fields of a message, so we
       can detect incompatible XML changes'''
    crc = mavutil.x25crc(msg.name + ' ')
    for f in msg.ordered_fields:
        crc.accumulate(f.type + ' ')
        crc.accumulate(f.name + ' ')
        if f.array_length:
            crc.accumulate(chr(f.array_length))
    return (crc.crc&0xFF) ^ (crc.crc>>8)
```

> This uses the same x25 checksum that is used at runtime. It calculates a CRC over the message name (such as “RAW_IMU”) followed by the type and name of each field, space separated. The order of the fields is the order they are sent over the wire. For arrays, the array length is also added.

Routing

MAVLink 2.0 has a source system and component ID as well as a destination system and component ID. While the source IDs are part of every message, destination IDs are only assigned for cases where a destination ID is required.

Router Implementation

The MAVLink Router created by Intel allows to mix-and-match different IP protocols with serial ports and route MAVLink traffic.

General Telemetry

MAVLink is designed to support sending continuous telemetry streams including position, velocity, attitude and similar key states of a drone.

UAVCAN Interaction

This chapter describes the MAVLink commands and messages that facilitate data exchange between the operator's equipment and the on-board UAVCAN nodes. The primary motivation is to enable the user to configure, monitor, and control the on-board UAVCAN nodes via the existing MAVLink connection.

Basics

The general description and specification of UAVCAN is available at http://uavcan.org.

The text below will be referring to the following terms:

- **Bridge node** - the piece of on-board equipment that bridges the on-board UAVCAN bus and the MAVLink connection. This function is often performed by the flight management unit, e.g. Pixhawk.
- **Remote equipment** - the other end of the MAVLink channel, e.g. the ground control station.

UAVCAN Node Identification

Every UAVCAN node has a bus-unique identifier referred to as "node ID". The node ID is an integer in the interval [1, 127], where the value 1 is typically used by the autopilot or some other kind of central controlling unit, and the values 126 and 127 are typically used by debugging or monitoring equipment.

Each unit that is capable of communicating via MAVLink and UAVCAN must use the same number for its MAVLink Component ID and the UAVCAN Node ID, otherwise serious inconsistencies may arise. Typically, if there is a single non-redundant autopilot, its UAVCAN Node ID and the MAVLink component ID will be set to 1 (one).

Every outgoing/incoming MAVLink message/command pertaining to a given UAVCAN node will have its field Component ID set to the same value as the Node ID of the referred UAVCAN node.

Node Status Reporting

Node Status Messages

In UAVCAN, the abstract node status information is represented by the standard message type `uavcan.protocol.NodeStatus`. Its MAVLink counterpart is `UAVCAN_NODE_STATUS`.

The bridge node should emit the MAVLink message `UAVCAN_NODE_STATUS` every time it receives a UAVCAN node status message. The bridge node is allowed to decimate the stream of node status messages in order to avoid congestion of the MAVLink channel, but the resulting frequency of status message emission should not be lower than 1 Hz per node.

The remote equipment can monitor which UAVCAN nodes are online by means of tracking the amount of time that passed since the last reception of the node status message for each online node. The remote equipment should consider the node to be offline if its last status message has arrived more than 5 seconds ago.

Extended Node Information

UAVCAN nodes are typically able to report some static information that identifies their type, purpose, vendor, revision, and such, via the standard service type `uavcan.protocol.GetNodeInfo`. In this context, "static" means that the data is not changing while the node is running. This information can be crucial for many important use cases.

The corresponding MAVLink message is `UAVCAN_NODE_INFO`. Its fields are direct mappings of the corresponding fields in the service type `uavcan.protocol.GetNodeInfo`.

The bridge node must emit the message `UAVCAN_NODE_INFO` in the following cases:

- Reception of a service response of type `uavcan.protocol.GetNodeInfo`. In turn, this service must be invoked when the following conditions are observed on the bus (please read the UAVCAN specification for a more detailed description of the principles of bus monitoring):
  - A new node has appeared online.
  - A known node has restarted.
- Reception of the MAVLink command `MAV_CMD_UAVCAN_GET_NODE_INFO`. In this case, the bridge node is required to emit `UAVCAN_NODE_INFO` once for every known node.
- It is also allowed, but not required, to unconditionally emit messages `UAVCAN_NODE_INFO` at a very low rate, in order to guarantee that the remote equipment always has a valid model of the on-board UAVCAN bus.

Configuration Parameter Management

UAVCAN defines a set of standard service types that facilitate the management of configuration parameters on UAVCAN nodes. The respective data type definitions can be found in the namespace `uavcan.protocol.param`.

The UAVCAN-MAVLink bridge does not define any additional messages for configuration parameter management. Instead, the following standard messages are used in the regular way:

- `PARAM_REQUEST_LIST` - used to request the list of configuration parameters from the specified UAVCAN node. Remember that the UAVCAN node is specified via the field Component ID.
- `PARAM_VALUE` - used by the bridge node to report the value of a configuration parameter. The node ID is reflected in the field Component ID.
- `PARAM_SET` - used by the remote equipment to set the value of a configuration parameter. The node ID is reflected in the field Component ID.

Note that the maximum length of a configuration parameter name is defined differently in UAVCAN and MAVLink. In MAVLink, the maximum length is 16 characters, whereas in UAVCAN the limit is 92 characters. Should the bridge node encounter long configuration parameter names that exceed the MAVLink's limit, it should exercise its best effort to reduce the name length presented on the MAVLink side while avoiding ambiguity. Designers of UAVCAN nodes, on their part, should avoid using configuration parameter names more than 16 characters long, until this deficiency of the MAVLink protocol is fixed.

Internet Access Bridge

UAVCAN defines a set of standard messages that facilitate communication between UAVCAN nodes and remote hosts on the Internet or LAN. The tentative specification can be viewed on GitHub. In the future, the set of MAVLink messages should be extended to allow forwarding of data packets between the bridge node and the Internet via the remote equipment (e.g. ground control station). If you're interested in this feature, please report to the UAVCAN mailing list.

Message Definitions

MAVLink messages are defined in XML files in the mavlink/message definitions folder. The messages that are common to all systems are defined in common.xml (only messages contained in this file are considered standard messages).

The common messages are provided as human-readable tables in: Common.

Vendor Specific Extensions (Dialects)

MAVLink protocol-specific and vendor-specific messages (dialects) are stored in separate XML files. These often include the common message definition, extending it with needed vendor or protocol specific messages.

> While a dialect can include any other message definition, care should be taken when including a definition file that includes another file (only a single level of nesting is tested).

> Vendor forks of MAVLink may contain messages that are not yet merged, and hence will not appear in this documentation.

The human-readable forms of the vendor XML files are linked below:

- ardupilotmega.xml
- ASLUAV.xml
- autoquad.xml
- icarous.xml
- matrixpilot.xml
- minimal.xml
- paparazzi.xml
- python_array_test.xml
- slugs.xml
- standard.xml
- test.xml
- ualberta.xml
- uAvionix.xml

MAVLINK Common Message Set

These messages define the common message set, which is the reference message set implemented by most ground control stations and autopilots.

This is a human-readable form of the XML definition file: common.xml.

> MAVLink 2 messages have an ID > 255 and are marked up using **(MAVLink 2)** in their description.

> MAVLink 2 extension fields that have been added to MAVLink 1 messages are displayed in blue.

MAVLink Protocol Version

The current MAVLink version is 2.3. The minor version numbers (after the dot) range from 1-255.

This file has protocol dialect: 0.

MAVLink Type Enumerations

MAV_AUTOPILOT

Micro air vehicle / autopilot classes. This identifies the individual model.

CMD ID	Field Name	Description
0	MAV_AUTOPILOT_GENERIC	Generic autopilot, full support for everything
1	MAV_AUTOPILOT_RESERVED	Reserved for future us
2	MAV_AUTOPILOT_SLUGS	SLUGS autopilot, http://slugsuav.soe.ucs
3	MAV_AUTOPILOT_ARDUPILOTMEGA	ArduPilotMega / ArduCopter, http://diydrones.com
4	MAV_AUTOPILOT_OPENPILOT	OpenPilot, http://openpilot.org
5	MAV_AUTOPILOT_GENERIC_WAYPOINTS_ONLY	Generic autopilot only supporting simple way
6	MAV_AUTOPILOT_GENERIC_WAYPOINTS_AND_SIMPLE_NAVIGATION_ONLY	Generic autopilot supp waypoints and other si navigation commands
7	MAV_AUTOPILOT_GENERIC_MISSION_FULL	Generic autopilot supp the full mission comma set
8	MAV_AUTOPILOT_INVALID	No valid autopilot, e.g. GCS or other MAVLink component
9	MAV_AUTOPILOT_PPZ	PPZ UAV - http://nongnu.org/papa
10	MAV_AUTOPILOT_UDB	UAV Dev Board
11	MAV_AUTOPILOT_FP	FlexiPilot
12	MAV_AUTOPILOT_PX4	PX4 Autopilot - http://pixhawk.ethz.ch/
13	MAV_AUTOPILOT_SMACCMPILOT	SMACCMPilot - http://smaccmpilot.org
14	MAV_AUTOPILOT_AUTOQUAD	AutoQuad -- http://autoquad.org
15	MAV_AUTOPILOT_ARMAZILA	Armazila -- http://armazila.com
16	MAV_AUTOPILOT_AEROB	Aerob -- http://aerob.ru
17	MAV_AUTOPILOT_ASLUAV	ASLUAV autopilot -- http://www.asl.ethz.ch
18	MAV_AUTOPILOT_SMARTAP	SmartAP Autopilot - http://sky-drones.com
19	MAV_AUTOPILOT_AIRRAILS	AirRails - http://uaventure.com

MAV_TYPE

CMD ID	Field Name	Description
0	MAV_TYPE_GENERIC	Generic micro air vehicle.
1	MAV_TYPE_FIXED_WING	Fixed wing aircraft.
2	MAV_TYPE_QUADROTOR	Quadrotor
3	MAV_TYPE_COAXIAL	Coaxial helicopter
4	MAV_TYPE_HELICOPTER	Normal helicopter with tail rotor.
5	MAV_TYPE_ANTENNA_TRACKER	Ground installation
6	MAV_TYPE_GCS	Operator control unit / ground control station
7	MAV_TYPE_AIRSHIP	Airship, controlled
8	MAV_TYPE_FREE_BALLOON	Free balloon, uncontrolled
9	MAV_TYPE_ROCKET	Rocket
10	MAV_TYPE_GROUND_ROVER	Ground rover
11	MAV_TYPE_SURFACE_BOAT	Surface vessel, boat, ship
12	MAV_TYPE_SUBMARINE	Submarine
13	MAV_TYPE_HEXAROTOR	Hexarotor
14	MAV_TYPE_OCTOROTOR	Octorotor
15	MAV_TYPE_TRICOPTER	Tricopter
16	MAV_TYPE_FLAPPING_WING	Flapping wing
17	MAV_TYPE_KITE	Kite
18	MAV_TYPE_ONBOARD_CONTROLLER	Onboard companion controller
19	MAV_TYPE_VTOL_DUOROTOR	Two-rotor VTOL using control surfaces in vertical operation in addition. Tailsitter.
20	MAV_TYPE_VTOL_QUADROTOR	Quad-rotor VTOL using a V-shaped quad config in vertical operation. Tailsitter.
21	MAV_TYPE_VTOL_TILTROTOR	Tiltrotor VTOL
22	MAV_TYPE_VTOL_RESERVED2	VTOL reserved 2
23	MAV_TYPE_VTOL_RESERVED3	VTOL reserved 3
24	MAV_TYPE_VTOL_RESERVED4	VTOL reserved 4
25	MAV_TYPE_VTOL_RESERVED5	VTOL reserved 5
26	MAV_TYPE_GIMBAL	Onboard gimbal
27	MAV_TYPE_ADSB	Onboard ADSB peripheral
28	MAV_TYPE_PARAFOIL	Steerable, nonrigid airfoil
29	MAV_TYPE_DODECAROTOR	Dodecarotor
30	MAV_TYPE_CAMERA	Camera
31	MAV_TYPE_CHARGING_STATION	Charging station

FIRMWARE_VERSION_TYPE

These values define the type of firmware release. These values indicate the first version or release of this type. For example the first alpha release would be 64, the second would be 65.

CMD ID	Field Name	Description
0	FIRMWARE_VERSION_TYPE_DEV	development release
64	FIRMWARE_VERSION_TYPE_ALPHA	alpha release
128	FIRMWARE_VERSION_TYPE_BETA	beta release
192	FIRMWARE_VERSION_TYPE_RC	release candidate
255	FIRMWARE_VERSION_TYPE_OFFICIAL	official stable release

HL_FAILURE_FLAG

Flags to report failure cases over the high latency telemtry.

CMD ID	Field Name	Description
1	HL_FAILURE_FLAG_GPS	GPS failure.
2	HL_FAILURE_FLAG_DIFFERENTIAL_PRESSURE	Differential pressure sensor failure.
4	HL_FAILURE_FLAG_ABSOLUTE_PRESSURE	Absolute pressure sensor failure.
8	HL_FAILURE_FLAG_3D_ACCEL	Accelerometer sensor failure.
16	HL_FAILURE_FLAG_3D_GYRO	Gyroscope sensor failure.
32	HL_FAILURE_FLAG_3D_MAG	Magnetometer sensor failure.
64	HL_FAILURE_FLAG_TERRAIN	Terrain subsystem failure.
128	HL_FAILURE_FLAG_BATTERY	Battery failure/critical low battery.
256	HL_FAILURE_FLAG_RC_RECEIVER	RC receiver failure/no rc connection.
512	HL_FAILURE_FLAG_OFFBOARD_LINK	Offboard link failure.
1024	HL_FAILURE_FLAG_ENGINE	Engine failure.
2048	HL_FAILURE_FLAG_GEOFENCE	Geofence violation.
4096	HL_FAILURE_FLAG_ESTIMATOR	Estimator failure, for example measurement rejection or large variances.
8192	HL_FAILURE_FLAG_MISSION	Mission failure.

MAV_MODE_FLAG

These flags encode the MAV mode.

CMD ID	Field Name	Description
128	MAV_MODE_FLAG_SAFETY_ARMED	0b10000000 MAV safety set to armed. Motors are enabled / running / can start. Ready to fly. Additional note: this flag is to be ignore when sent in the command MAV_CMD_DO_SET_MODE and MAV_CMD_COMPONENT_ARM_DISARM shall be used instead. The flag can still be used to report the armed state.
64	MAV_MODE_FLAG_MANUAL_INPUT_ENABLED	0b01000000 remote control input is enabled.
32	MAV_MODE_FLAG_HIL_ENABLED	0b00100000 hardware in the loop simulation. All motors / actuators are blocked, but internal software is full operational.
16	MAV_MODE_FLAG_STABILIZE_ENABLED	0b00010000 system stabilizes electronically its attitude (and optionally position). It needs however further control inputs to move around.
8	MAV_MODE_FLAG_GUIDED_ENABLED	0b00001000 guided mode enabled, system flies waypoints / mission items.
4	MAV_MODE_FLAG_AUTO_ENABLED	0b00000100 autonomous mode enabled, system finds its own goal positions. Guided flag can be set or not, depends on the actual implementation.
2	MAV_MODE_FLAG_TEST_ENABLED	0b00000010 system has a test mode enabled. This flag is intended for temporary system tests and should not be used for stable implementations.
1	MAV_MODE_FLAG_CUSTOM_MODE_ENABLED	0b00000001 Reserved for future use.

MAV_MODE_FLAG_DECODE_POSITION

These values encode the bit positions of the decode position. These values can be used to read the value of a flag bit by combining the base_mode variable with AND with the flag position value. The result will be either 0 or 1, depending on if the flag is set or not.

CMD ID	Field Name	Description
128	MAV_MODE_FLAG_DECODE_POSITION_SAFETY	First bit: 10000000
64	MAV_MODE_FLAG_DECODE_POSITION_MANUAL	Second bit: 01000000
32	MAV_MODE_FLAG_DECODE_POSITION_HIL	Third bit: 00100000
16	MAV_MODE_FLAG_DECODE_POSITION_STABILIZE	Fourth bit: 00010000
8	MAV_MODE_FLAG_DECODE_POSITION_GUIDED	Fifth bit: 00001000
4	MAV_MODE_FLAG_DECODE_POSITION_AUTO	Sixt bit: 00000100
2	MAV_MODE_FLAG_DECODE_POSITION_TEST	Seventh bit: 00000010
1	MAV_MODE_FLAG_DECODE_POSITION_CUSTOM_MODE	Eighth bit: 00000001

MAV_GOTO

Override command, pauses current mission execution and moves immediately to a position

CMD ID	Field Name	Description
0	MAV_GOTO_DO_HOLD	Hold at the current position.
1	MAV_GOTO_DO_CONTINUE	Continue with the next item in mission execution.
2	MAV_GOTO_HOLD_AT_CURRENT_POSITION	Hold at the current position of the system
3	MAV_GOTO_HOLD_AT_SPECIFIED_POSITION	Hold at the position specified in the parameters of the DO_HOLD action

MAV_MODE

These defines are predefined OR-combined mode flags. There is no need to use values from this enum, but it simplifies the use of the mode flags. Note that manual input is enabled in all modes as a safety override.

CMD ID	Field Name	Description
0	MAV_MODE_PREFLIGHT	System is not ready to fly, booting, calibrating, etc. No flag is set.
80	MAV_MODE_STABILIZE_DISARMED	System is allowed to be active, under assisted RC control.
208	MAV_MODE_STABILIZE_ARMED	System is allowed to be active, under assisted RC control.
64	MAV_MODE_MANUAL_DISARMED	System is allowed to be active, under manual (RC) control, no stabilization
192	MAV_MODE_MANUAL_ARMED	System is allowed to be active, under manual (RC) control, no stabilization
88	MAV_MODE_GUIDED_DISARMED	System is allowed to be active, under autonomous control, manual setpoint
216	MAV_MODE_GUIDED_ARMED	System is allowed to be active, under autonomous control, manual setpoint
92	MAV_MODE_AUTO_DISARMED	System is allowed to be active, under autonomous control and navigation (the trajectory is decided onboard and not pre-programmed by waypoints)
220	MAV_MODE_AUTO_ARMED	System is allowed to be active, under autonomous control and navigation (the trajectory is decided onboard and not pre-programmed by waypoints)
66	MAV_MODE_TEST_DISARMED	UNDEFINED mode. This solely depends on the autopilot - use with caution, intended for developers only.
194	MAV_MODE_TEST_ARMED	UNDEFINED mode. This solely depends on the autopilot - use with caution, intended for developers only.

MAV_STATE

CMD ID	Field Name	Description
0	MAV_STATE_UNINIT	Uninitialized system, state is unknown.
	MAV_STATE_BOOT	System is booting up.
	MAV_STATE_CALIBRATING	System is calibrating and not flight-ready.
	MAV_STATE_STANDBY	System is grounded and on standby. It can be launched any time.
	MAV_STATE_ACTIVE	System is active and might be already airborne. Motors are engaged.
	MAV_STATE_CRITICAL	System is in a non-normal flight mode. It can however still navigate.
	MAV_STATE_EMERGENCY	System is in a non-normal flight mode. It lost control over parts or over the whole airframe. It is in mayday and going down.
	MAV_STATE_POWEROFF	System just initialized its power-down sequence, will shut down now.
	MAV_STATE_FLIGHT_TERMINATION	System is terminating itself.

MAV_COMPONENT

CMD ID	Field Name	Description
0	MAV_COMP_ID_ALL	
1	MAV_COMP_ID_AUTOPILOT1	
100	MAV_COMP_ID_CAMERA	
101	MAV_COMP_ID_CAMERA2	
102	MAV_COMP_ID_CAMERA3	
103	MAV_COMP_ID_CAMERA4	
104	MAV_COMP_ID_CAMERA5	
105	MAV_COMP_ID_CAMERA6	
140	MAV_COMP_ID_SERVO1	
141	MAV_COMP_ID_SERVO2	
142	MAV_COMP_ID_SERVO3	
143	MAV_COMP_ID_SERVO4	
144	MAV_COMP_ID_SERVO5	
145	MAV_COMP_ID_SERVO6	
146	MAV_COMP_ID_SERVO7	
147	MAV_COMP_ID_SERVO8	
148	MAV_COMP_ID_SERVO9	
149	MAV_COMP_ID_SERVO10	
150	MAV_COMP_ID_SERVO11	
151	MAV_COMP_ID_SERVO12	
152	MAV_COMP_ID_SERVO13	

153	MAV_COMP_ID_SERVO14	
154	MAV_COMP_ID_GIMBAL	
155	MAV_COMP_ID_LOG	
156	MAV_COMP_ID_ADSB	
157	MAV_COMP_ID_OSD	On Screen Display (OSD) devices for video links
158	MAV_COMP_ID_PERIPHERAL	Generic autopilot peripheral component ID. Meant for devices that do not implement the parameter sub-protocol
159	MAV_COMP_ID_QX1_GIMBAL	
180	MAV_COMP_ID_MAPPER	
190	MAV_COMP_ID_MISSIONPLANNER	
195	MAV_COMP_ID_PATHPLANNER	
200	MAV_COMP_ID_IMU	
201	MAV_COMP_ID_IMU_2	
202	MAV_COMP_ID_IMU_3	
220	MAV_COMP_ID_GPS	
221	MAV_COMP_ID_GPS2	
240	MAV_COMP_ID_UDP_BRIDGE	
241	MAV_COMP_ID_UART_BRIDGE	
250	MAV_COMP_ID_SYSTEM_CONTROL	

MAV_SYS_STATUS_SENSOR

These encode the sensors whose status is sent as part of the SYS_STATUS message.

CMD ID	Field Name	Description
1	MAV_SYS_STATUS_SENSOR_3D_GYRO	0x01 3D gyro
2	MAV_SYS_STATUS_SENSOR_3D_ACCEL	0x02 3D accelerometer
4	MAV_SYS_STATUS_SENSOR_3D_MAG	0x04 3D magnetometer
8	MAV_SYS_STATUS_SENSOR_ABSOLUTE_PRESSURE	0x08 absolute pressure
16	MAV_SYS_STATUS_SENSOR_DIFFERENTIAL_PRESSURE	0x10 differential pressure
32	MAV_SYS_STATUS_SENSOR_GPS	0x20 GPS
64	MAV_SYS_STATUS_SENSOR_OPTICAL_FLOW	0x40 optical flow
128	MAV_SYS_STATUS_SENSOR_VISION_POSITION	0x80 computer vision position
256	MAV_SYS_STATUS_SENSOR_LASER_POSITION	0x100 laser based position
512	MAV_SYS_STATUS_SENSOR_EXTERNAL_GROUND_TRUTH	0x200 external ground truth (Vicon or Leica)
1024	MAV_SYS_STATUS_SENSOR_ANGULAR_RATE_CONTROL	0x400 3D angular rate control
2048	MAV_SYS_STATUS_SENSOR_ATTITUDE_STABILIZATION	0x800 attitude stabilization
4096	MAV_SYS_STATUS_SENSOR_YAW_POSITION	0x1000 yaw position
8192	MAV_SYS_STATUS_SENSOR_Z_ALTITUDE_CONTROL	0x2000 z/altitude control
16384	MAV_SYS_STATUS_SENSOR_XY_POSITION_CONTROL	0x4000 x/y position control
32768	MAV_SYS_STATUS_SENSOR_MOTOR_OUTPUTS	0x8000 motor outputs / control
65536	MAV_SYS_STATUS_SENSOR_RC_RECEIVER	0x10000 rc receiver
131072	MAV_SYS_STATUS_SENSOR_3D_GYRO2	0x20000 2nd 3D gyro
262144	MAV_SYS_STATUS_SENSOR_3D_ACCEL2	0x40000 2nd 3D accelerometer
524288	MAV_SYS_STATUS_SENSOR_3D_MAG2	0x80000 2nd 3D magnetometer
1048576	MAV_SYS_STATUS_GEOFENCE	0x100000 geofence
2097152	MAV_SYS_STATUS_AHRS	0x200000 AHRS subsystem health
4194304	MAV_SYS_STATUS_TERRAIN	0x400000 Terrain subsystem health
8388608	MAV_SYS_STATUS_REVERSE_MOTOR	0x800000 Motors are reversed
16777216	MAV_SYS_STATUS_LOGGING	0x1000000 Logging
33554432	MAV_SYS_STATUS_SENSOR_BATTERY	0x2000000 Battery

MAV_FRAME

CMD ID	Field Name	Description
0	MAV_FRAME_GLOBAL	Global coordinate frame, WGS84 coordinate system. First value / x: latitude, second value / y: longitude, third value / z: positive altitude over mean sea level (MSL).
1	MAV_FRAME_LOCAL_NED	Local coordinate frame, Z-down (x: north, y: east, z: down).
2	MAV_FRAME_MISSION	NOT a coordinate frame, indicates a mission command.
		Global coordinate frame, WGS84 coordinate system,

3	MAV_FRAME_GLOBAL_RELATIVE_ALT	relative altitude over ground with respect to the home position. First value / x: latitude, second value / y: longitude, third value / z: positive altitude with 0 being at the altitude of the home location.
4	MAV_FRAME_LOCAL_ENU	Local coordinate frame, Z-up (x: east, y: north, z: up).
5	MAV_FRAME_GLOBAL_INT	Global coordinate frame, WGS84 coordinate system. First value / x: latitude in degrees*1.0e-7, second value / y: longitude in degrees*1.0e-7, third value / z: positive altitude over mean sea level (MSL).
6	MAV_FRAME_GLOBAL_RELATIVE_ALT_INT	Global coordinate frame, WGS84 coordinate system, relative altitude over ground with respect to the home position. First value / x: latitude in degrees*10e-7, second value / y: longitude in degrees*10e-7, third value / z: positive altitude with 0 being at the altitude of the home location.
7	MAV_FRAME_LOCAL_OFFSET_NED	Offset to the current local frame. Anything expressed in this frame should be added to the current local frame position.
8	MAV_FRAME_BODY_NED	Setpoint in body NED frame. This makes sense if all position control is externalized - e.g. useful to command 2 m/s^2 acceleration to the right.
9	MAV_FRAME_BODY_OFFSET_NED	Offset in body NED frame. This makes sense if adding setpoints to the current flight path, to avoid an obstacle - e.g. useful to command 2 m/s^2 acceleration to the east.
10	MAV_FRAME_GLOBAL_TERRAIN_ALT	Global coordinate frame with above terrain level altitude. WGS84 coordinate system, relative altitude over terrain with respect to the waypoint coordinate. First value / x: latitude in degrees, second value / y: longitude in degrees, third value / z: positive altitude in meters with 0 being at ground level in terrain model.
11	MAV_FRAME_GLOBAL_TERRAIN_ALT_INT	Global coordinate frame with above terrain level altitude. WGS84 coordinate system, relative altitude over terrain with respect to the waypoint coordinate. First value / x: latitude in degrees*10e-7, second value / y: longitude in degrees*10e-7, third value / z: positive altitude in meters with 0 being at ground level in terrain model.
12	MAV_FRAME_BODY_FRD	Body fixed frame of reference, Z-down (x: forward, y: right, z: down).
13	MAV_FRAME_BODY_FLU	Body fixed frame of reference, Z-up (x: forward, y: left, z: up).
14	MAV_FRAME_MOCAP_NED	Odometry local coordinate frame of data given by a motion capture system, Z-down (x: north, y: east, z: down).
15	MAV_FRAME_MOCAP_ENU	Odometry local coordinate frame of data given by a motion capture system, Z-up (x: east, y: north, z: up).
16	MAV_FRAME_VISION_NED	Odometry local coordinate frame of data given by a vision estimation system, Z-down (x: north, y: east, z: down).
17	MAV_FRAME_VISION_ENU	Odometry local coordinate frame of data given by a vision estimation system, Z-up (x: east, y: north, z: up).
18	MAV_FRAME_ESTIM_NED	Odometry local coordinate frame of data given by an estimator running onboard the vehicle, Z-down (x: north, y: east, z: down).
19	MAV_FRAME_ESTIM_ENU	Odometry local coordinate frame of data given by an estimator running onboard the vehicle, Z-up (x: east, y:

		noth, z: up).

MAVLINK_DATA_STREAM_TYPE

CMD ID	Field Name	Description
	MAVLINK_DATA_STREAM_IMG_JPEG	
	MAVLINK_DATA_STREAM_IMG_BMP	
	MAVLINK_DATA_STREAM_IMG_RAW8U	
	MAVLINK_DATA_STREAM_IMG_RAW32U	
	MAVLINK_DATA_STREAM_IMG_PGM	
	MAVLINK_DATA_STREAM_IMG_PNG	

FENCE_ACTION

CMD ID	Field Name	Description
0	FENCE_ACTION_NONE	Disable fenced mode
1	FENCE_ACTION_GUIDED	Switched to guided mode to return point (fence point 0)
2	FENCE_ACTION_REPORT	Report fence breach, but don't take action
3	FENCE_ACTION_GUIDED_THR_PASS	Switched to guided mode to return point (fence point 0) with manual throttle control
4	FENCE_ACTION_RTL	Switch to RTL (return to launch) mode and head for the return point.

FENCE_BREACH

CMD ID	Field Name	Description
0	FENCE_BREACH_NONE	No last fence breach
1	FENCE_BREACH_MINALT	Breached minimum altitude
2	FENCE_BREACH_MAXALT	Breached maximum altitude
3	FENCE_BREACH_BOUNDARY	Breached fence boundary

MAV_MOUNT_MODE

Enumeration of possible mount operation modes

CMD ID	Field Name	Description
0	MAV_MOUNT_MODE_RETRACT	Load and keep safe position (Roll,Pitch,Yaw) from permant memory and stop stabilization
1	MAV_MOUNT_MODE_NEUTRAL	Load and keep neutral position (Roll,Pitch,Yaw) from permanent memory.
2	MAV_MOUNT_MODE_MAVLINK_TARGETING	Load neutral position and start MAVLink Roll,Pitch,Yaw control with stabilization
3	MAV_MOUNT_MODE_RC_TARGETING	Load neutral position and start RC Roll,Pitch,Yaw control with stabilization
4	MAV_MOUNT_MODE_GPS_POINT	Load neutral position and start to point to Lat,Lon,Alt

UAVCAN_NODE_HEALTH

Generalized UAVCAN node health

CMD ID	Field Name	Description
0	UAVCAN_NODE_HEALTH_OK	The node is functioning properly.
1	UAVCAN_NODE_HEALTH_WARNING	A critical parameter went out of range or the node has encountered a minor failure.
2	UAVCAN_NODE_HEALTH_ERROR	The node has encountered a major failure.
3	UAVCAN_NODE_HEALTH_CRITICAL	The node has suffered a fatal malfunction.

UAVCAN_NODE_MODE

Generalized UAVCAN node mode

CMD ID	Field Name	Description
0	UAVCAN_NODE_MODE_OPERATIONAL	The node is performing its primary functions.
1	UAVCAN_NODE_MODE_INITIALIZATION	The node is initializing; this mode is entered immediately after startup.
2	UAVCAN_NODE_MODE_MAINTENANCE	The node is under maintenance.
3	UAVCAN_NODE_MODE_SOFTWARE_UPDATE	The node is in the process of updating its software.
7	UAVCAN_NODE_MODE_OFFLINE	The node is no longer available online.

MAV_CMD

Commands to be executed by the MAV. They can be executed on user request, or as part of a mission script. If the action is used in a mission, the parameter mapping to the waypoint/mission message is as follows: Param 1, Param 2, Param 3, Param 4, X: Param 5, Y:Param 6, Z:Param 7. This command list is similar what ARINC 424 is for commercial aircraft: A data format how to interpret waypoint/mission data.

CMD ID	Field Name	Description
16	MAV_CMD_NAV_WAYPOINT	Navigate to waypoint.
	Mission Param #1	Hold time in decimal seconds. (ignored by wing, time to stay at waypoint for rotary w

	Mission Param #2	Acceptance radius in meters (if the sphere this radius is hit, the waypoint counts as r
	Mission Param #3	0 to pass through the WP, if > 0 radius in to pass by WP. Positive value for clockwis negative value for counter-clockwise orbit trajectory control.
	Mission Param #4	Desired yaw angle at waypoint (rotary win for unchanged.
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Altitude
17	MAV_CMD_NAV_LOITER_UNLIM	Loiter around this waypoint an unlimited a of time
	Mission Param #1	Empty
	Mission Param #2	Empty
	Mission Param #3	Radius around waypoint, in meters. If pos loiter clockwise, else counter-clockwise
	Mission Param #4	Desired yaw angle.
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Altitude
18	MAV_CMD_NAV_LOITER_TURNS	Loiter around this waypoint for X turns
	Mission Param #1	Turns
	Mission Param #2	Empty
	Mission Param #3	Radius around waypoint, in meters. If pos loiter clockwise, else counter-clockwise
	Mission Param #4	Forward moving aircraft this sets exit xtra location: 0 for center of loiter wp, 1 for exi location. Else, this is desired yaw angle
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Altitude
19	MAV_CMD_NAV_LOITER_TIME	Loiter around this waypoint for X seconds
	Mission Param #1	Seconds (decimal)
	Mission Param #2	Empty
	Mission Param #3	Radius around waypoint, in meters. If pos loiter clockwise, else counter-clockwise
	Mission Param #4	Forward moving aircraft this sets exit xtra location: 0 for center of loiter wp, 1 for exi

		location. Else, this is desired yaw angle
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Altitude
20	MAV_CMD_NAV_RETURN_TO_LAUNCH	Return to launch location
	Mission Param #1	Empty
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
21	MAV_CMD_NAV_LAND	Land at location
	Mission Param #1	Abort Alt
	Mission Param #2	Precision land mode. (0 = normal landing, opportunistic precision landing, 2 = requir precsion landing)
	Mission Param #3	Empty
	Mission Param #4	Desired yaw angle. NaN for unchanged.
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Altitude (ground level)
22	MAV_CMD_NAV_TAKEOFF	Takeoff from ground / hand
	Mission Param #1	Minimum pitch (if airspeed sensor present desired pitch without sensor
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Yaw angle (if magnetometer present), ign without magnetometer. NaN for unchange
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Altitude
23	MAV_CMD_NAV_LAND_LOCAL	Land at local position (local frame only)
	Mission Param #1	Landing target number (if available)

	Mission Param #2	Maximum accepted offset from desired la position [m] - computed magnitude from s coordinates: d = sqrt(x^2 + y^2 + z^2), wh gives the maximum accepted distance be the desired landing position and the positi where the vehicle is about to land
	Mission Param #3	Landing descend rate [ms^-1]
	Mission Param #4	Desired yaw angle [rad]
	Mission Param #5	Y-axis position [m]
	Mission Param #6	X-axis position [m]
	Mission Param #7	Z-axis / ground level position [m]
24	MAV_CMD_NAV_TAKEOFF_LOCAL	Takeoff from local position (local frame on
	Mission Param #1	Minimum pitch (if airspeed sensor present desired pitch without sensor [rad]
	Mission Param #2	Empty
	Mission Param #3	Takeoff ascend rate [ms^-1]
	Mission Param #4	Yaw angle [rad] (if magnetometer or anoth estimation source present), ignored witho these
	Mission Param #5	Y-axis position [m]
	Mission Param #6	X-axis position [m]
	Mission Param #7	Z-axis position [m]
25	MAV_CMD_NAV_FOLLOW	Vehicle following, i.e. this waypoint repres position of a moving vehicle
	Mission Param #1	Following logic to use (e.g. loitering or sin following) - depends on specific autopilot implementation
	Mission Param #2	Ground speed of vehicle to be followed
	Mission Param #3	Radius around waypoint, in meters. If pos loiter clockwise, else counter-clockwise
	Mission Param #4	Desired yaw angle.
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Altitude
30	MAV_CMD_NAV_CONTINUE_AND_CHANGE_ALT	Continue on the current course and climb/descend to specified altitude. When altitude is reached continue to the next co (i.e., don't proceed to the next command u desired altitude is reached.
		Climb or Descend (0 = Neutral, command completes when within 5m of this comma

	Mission Param #1	altitude, 1 = Climbing, command complete at or above this command's altitude, 2 = Descending, command completes when a below this command's altitude.
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Desired altitude in meters
31	MAV_CMD_NAV_LOITER_TO_ALT	Begin loiter at the specified Latitude and Longitude. If Lat=Lon=0, then loiter at the position. Don't consider the navigation co complete (don't leave loiter) until the altitu been reached. Additionally, if the Heading Required parameter is non-zero the aircra not leave the loiter until heading toward th waypoint.
	Mission Param #1	Heading Required (0 = False)
	Mission Param #2	Radius in meters. If positive loiter clockwis negative counter-clockwise, 0 means no c to standard loiter.
	Mission Param #3	Empty
	Mission Param #4	Forward moving aircraft this sets exit xtrac location: 0 for center of loiter wp, 1 for exi location
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Altitude
32	MAV_CMD_DO_FOLLOW	Being following a target
	Mission Param #1	System ID (the system ID of the FOLLOW_TARGET beacon). Send 0 to d follow-me and return to the default positio mode
	Mission Param #2	RESERVED
	Mission Param #3	RESERVED
	Mission Param #4	altitude flag: 0: Keep current altitude, 1: ke altitude difference to target, 2: go to a fixe altitude above home
	Mission Param #5	altitude
	Mission Param #6	RESERVED
	Mission Param #7	TTL in seconds in which the MAV should g default position hold mode after a messag timeout

33	MAV_CMD_DO_FOLLOW_REPOSITION	Reposition the MAV after a follow target command has been sent
	Mission Param #1	Camera q1 (where 0 is on the ray from th camera to the tracking device)
	Mission Param #2	Camera q2
	Mission Param #3	Camera q3
	Mission Param #4	Camera q4
	Mission Param #5	altitude offset from target (m)
	Mission Param #6	X offset from target (m)
	Mission Param #7	Y offset from target (m)
80	MAV_CMD_NAV_ROI	THIS INTERFACE IS DEPRECATED AS JANUARY 2018. Please use MAV_CMD_DO_SET_ROI_* messages ir Sets the region of interest (ROI) for a sens or the vehicle itself. This can then be used vehicles control system to control the vehi attitude and the attitude of various sensor as cameras.
	Mission Param #1	Region of intereset mode. (see MAV_ROI
	Mission Param #2	Waypoint index/ target ID. (see MAV_ROI
	Mission Param #3	ROI index (allows a vehicle to manage mu ROI's)
	Mission Param #4	Empty
	Mission Param #5	x the location of the fixed ROI (see MAV_
	Mission Param #6	y
	Mission Param #7	z
81	MAV_CMD_NAV_PATHPLANNING	Control autonomous path planning on the
	Mission Param #1	0: Disable local obstacle avoidance / local planning (without resetting map), 1: Enabl path planning, 2: Enable and reset local p planning
	Mission Param #2	0: Disable full path planning (without reset map), 1: Enable, 2: Enable and reset map/occupancy grid, 3: Enable and reset route, but not occupancy grid
	Mission Param #3	Empty
	Mission Param #4	Yaw angle at goal, in compass degrees, [(
	Mission Param #5	Latitude/X of goal
	Mission Param #6	Longitude/Y of goal
	Mission Param #7	Altitude/Z of goal
82	MAV_CMD_NAV_SPLINE_WAYPOINT	Navigate to waypoint using a spline path.

	Mission Param #1	Hold time in decimal seconds. (ignored by wing, time to stay at waypoint for rotary w
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Latitude/X of goal
	Mission Param #6	Longitude/Y of goal
	Mission Param #7	Altitude/Z of goal
84	MAV_CMD_NAV_VTOL_TAKEOFF	Takeoff from ground using VTOL mode
	Mission Param #1	Empty
	Mission Param #2	Front transition heading, see VTOL_TRANSITION_HEADING enum.
	Mission Param #3	Empty
	Mission Param #4	Yaw angle in degrees. NaN for unchange
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Altitude
85	MAV_CMD_NAV_VTOL_LAND	Land using VTOL mode
	Mission Param #1	Empty
	Mission Param #2	Empty
	Mission Param #3	Approach altitude (with the same referenc Altitude field). NaN if unspecified.
	Mission Param #4	Yaw angle in degrees. NaN for unchange
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Altitude (ground level)
92	MAV_CMD_NAV_GUIDED_ENABLE	hand control over to an external controller
	Mission Param #1	On / Off (> 0.5f on)
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty

93	MAV_CMD_NAV_DELAY	Delay the next navigation command a nur seconds or until a specified time
	Mission Param #1	Delay in seconds (decimal, -1 to enable ti day fields)
	Mission Param #2	hour (24h format, UTC, -1 to ignore)
	Mission Param #3	minute (24h format, UTC, -1 to ignore)
	Mission Param #4	second (24h format, UTC)
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
94	MAV_CMD_NAV_PAYLOAD_PLACE	Descend and place payload. Vehicle desc until it detects a hanging payload has reac ground, the gripper is opened to release t payload
	Mission Param #1	Maximum distance to descend (meters)
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Latitude (deg * 1E7)
	Mission Param #6	Longitude (deg * 1E7)
	Mission Param #7	Altitude (meters)
95	MAV_CMD_NAV_LAST	NOP - This command is only used to mark upper limit of the NAV/ACTION command enumeration
	Mission Param #1	Empty
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
112	MAV_CMD_CONDITION_DELAY	Delay mission state machine.
	Mission Param #1	Delay in seconds (decimal)
	Mission Param #2	Empty
	Mission Param #3	Empty

	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
113	MAV_CMD_CONDITION_CHANGE_ALT	Ascend/descend at rate. Delay mission st machine until desired altitude reached.
	Mission Param #1	Descent / Ascend rate (m/s)
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Finish Altitude
114	MAV_CMD_CONDITION_DISTANCE	Delay mission state machine until within d distance of next NAV point.
	Mission Param #1	Distance (meters)
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
115	MAV_CMD_CONDITION_YAW	Reach a certain target angle.
	Mission Param #1	target angle: [0-360], 0 is north
	Mission Param #2	speed during yaw change:[deg per secon
	Mission Param #3	direction: negative: counter clockwise, pos clockwise [-1,1]
	Mission Param #4	relative offset or absolute angle: [1,0]
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
159	MAV_CMD_CONDITION_LAST	NOP - This command is only used to marl upper limit of the CONDITION commands enumeration

	Mission Param #1	Empty
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
176	MAV_CMD_DO_SET_MODE	Set system mode.
	Mission Param #1	Mode, as defined by ENUM MAV_MODE
	Mission Param #2	Custom mode - this is system specific, ple refer to the individual autopilot specificatio details.
	Mission Param #3	Custom sub mode - this is system specific refer to the individual autopilot specificatio details.
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
177	MAV_CMD_DO_JUMP	Jump to the desired command in the miss Repeat this action only the specified numb times
	Mission Param #1	Sequence number
	Mission Param #2	Repeat count
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
178	MAV_CMD_DO_CHANGE_SPEED	Change speed and/or throttle set points.
	Mission Param #1	Speed type (0=Airspeed, 1=Ground Spee
	Mission Param #2	Speed (m/s, -1 indicates no change)
	Mission Param #3	Throttle (Percent, -1 indicates no change
	Mission Param #4	absolute or relative [0,1]
	Mission Param #5	Empty
	Mission Param #6	Empty

	Mission Param #7	Empty
179	MAV_CMD_DO_SET_HOME	Changes the home location either to the c location or a specified location.
	Mission Param #1	Use current (1=use current location, 0=us specified location)
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Altitude
180	MAV_CMD_DO_SET_PARAMETER	Set a system parameter. Caution! Use of t command requires knowledge of the num enumeration value of the parameter.
	Mission Param #1	Parameter number
	Mission Param #2	Parameter value
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
181	MAV_CMD_DO_SET_RELAY	Set a relay to a condition.
	Mission Param #1	Relay number
	Mission Param #2	Setting (1=on, 0=off, others possible depe on system hardware)
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
182	MAV_CMD_DO_REPEAT_RELAY	Cycle a relay on and off for a desired num cyles with a desired period.
	Mission Param #1	Relay number
	Mission Param #2	Cycle count

	Mission Param #3	Cycle time (seconds, decimal)
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
183	MAV_CMD_DO_SET_SERVO	Set a servo to a desired PWM value.
	Mission Param #1	Servo number
	Mission Param #2	PWM (microseconds, 1000 to 2000 typica
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
184	MAV_CMD_DO_REPEAT_SERVO	Cycle a between its nominal setting and a PWM for a desired number of cycles with desired period.
	Mission Param #1	Servo number
	Mission Param #2	PWM (microseconds, 1000 to 2000 typica
	Mission Param #3	Cycle count
	Mission Param #4	Cycle time (seconds)
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
185	MAV_CMD_DO_FLIGHTTERMINATION	Terminate flight immediately
	Mission Param #1	Flight termination activated if > 0.5
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
186	MAV_CMD_DO_CHANGE_ALTITUDE	Change altitude set point.
	Mission Param #1	Altitude in meters

	Mission Param #2	Mav frame of new altitude (see MAV_FRA
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
189	MAV_CMD_DO_LAND_START	Mission command to perform a landing. T used as a marker in a mission to tell the a where a sequence of mission items that represents a landing starts. It may also be via a COMMAND_LONG to trigger a landi which case the nearest (geographically) la sequence in the mission will be used. The Latitude/Longitude is optional, and may be 0 if not needed. If specified then it will be help find the closest landing sequence.
	Mission Param #1	Empty
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Empty
190	MAV_CMD_DO_RALLY_LAND	Mission command to perform a landing fro rally point.
	Mission Param #1	Break altitude (meters)
	Mission Param #2	Landing speed (m/s)
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
191	MAV_CMD_DO_GO_AROUND	Mission command to safely abort an auto landing.
	Mission Param #1	Altitude (meters)
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty

	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
192	MAV_CMD_DO_REPOSITION	Reposition the vehicle to a specific WGS8 position.
	Mission Param #1	Ground speed, less than 0 (-1) for default
	Mission Param #2	Bitmask of option flags, see the MAV_DO_REPOSITION_FLAGS enum.
	Mission Param #3	Reserved
	Mission Param #4	Yaw heading, NaN for unchanged. For pla indicates loiter direction (0: clockwise, 1: c clockwise)
	Mission Param #5	Latitude (deg * 1E7)
	Mission Param #6	Longitude (deg * 1E7)
	Mission Param #7	Altitude (meters)
193	MAV_CMD_DO_PAUSE_CONTINUE	If in a GPS controlled position mode, hold current position or continue.
	Mission Param #1	0: Pause current mission or reposition cor hold current position. 1: Continue mission VTOL capable vehicle should enter hover (multicopter and VTOL planes). A plane sl loiter with the default loiter radius.
	Mission Param #2	Reserved
	Mission Param #3	Reserved
	Mission Param #4	Reserved
	Mission Param #5	Reserved
	Mission Param #6	Reserved
	Mission Param #7	Reserved
194	MAV_CMD_DO_SET_REVERSE	Set moving direction to forward or reverse
	Mission Param #1	Direction (0=Forward, 1=Reverse)
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty

195	MAV_CMD_DO_SET_ROI_LOCATION	Sets the region of interest (ROI) to a locat This can then be used by the vehicles cor system to control the vehicle attitude and attitude of various sensors such as camer
	Mission Param #1	Empty
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Altitude
196	MAV_CMD_DO_SET_ROI_WPNEXT_OFFSET	Sets the region of interest (ROI) to be tow waypoint, with optional pitch/roll/yaw offse can then be used by the vehicles control s to control the vehicle attitude and the attitu various sensors such as cameras.
	Mission Param #1	Empty
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	pitch offset from next waypoint
	Mission Param #6	roll offset from next waypoint
	Mission Param #7	yaw offset from next waypoint
197	MAV_CMD_DO_SET_ROI_NONE	Cancels any previous ROI command retu vehicle/sensors to default flight characteri This can then be used by the vehicles cor system to control the vehicle attitude and attitude of various sensors such as camer
	Mission Param #1	Empty
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
200	MAV_CMD_DO_CONTROL_VIDEO	Control onboard camera system.
	Mission Param #1	Camera ID (-1 for all)
	Mission Param #2	Transmission: 0: disabled, 1: enabled

	Mission Param #2	compressed, 2: enabled raw
	Mission Param #3	Transmission mode: 0: video stream, >0: images every n seconds (decimal)
	Mission Param #4	Recording: 0: disabled, 1: enabled compr 2: enabled raw
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
201	MAV_CMD_DO_SET_ROI	THIS INTERFACE IS DEPRECATED AS JANUARY 2018. Please use MAV_CMD_DO_SET_ROI_* messages ir Sets the region of interest (ROI) for a sens or the vehicle itself. This can then be used vehicles control system to control the vehi attitude and the attitude of various sensor as cameras.
	Mission Param #1	Region of intereset mode. (see MAV_ROI
	Mission Param #2	Waypoint index/ target ID. (see MAV_ROI
	Mission Param #3	ROI index (allows a vehicle to manage mu ROI's)
	Mission Param #4	Empty
	Mission Param #5	MAV_ROI_WPNEXT: pitch offset from nex waypoint, MAV_ROI_LOCATION: latitude
	Mission Param #6	MAV_ROI_WPNEXT: roll offset from next waypoint, MAV_ROI_LOCATION: longitud
	Mission Param #7	MAV_ROI_WPNEXT: yaw offset from nex waypoint, MAV_ROI_LOCATION: altitude
202	MAV_CMD_DO_DIGICAM_CONFIGURE	THIS INTERFACE IS DEPRECATED sinc 01. Please use PARAM_EXT_XXX messa and the camera definition format describe https://mavlink.io/en/protocol/camera_def.
	Mission Param #1	Modes: P, TV, AV, M, Etc
	Mission Param #2	Shutter speed: Divisor number for one se
	Mission Param #3	Aperture: F stop number
	Mission Param #4	ISO number e.g. 80, 100, 200, Etc
	Mission Param #5	Exposure type enumerator
	Mission Param #6	Command Identity
	Mission Param #7	Main engine cut-off time before camera tri seconds/10 (0 means no cut-off)
203	MAV_CMD_DO_DIGICAM_CONTROL	THIS INTERFACE IS DEPRECATED sinc 01. Please use PARAM_EXT_XXX messa and the camera definition format describe https://mavlink.io/en/protocol/camera_def.

	Mission Param #1	Session control e.g. show/hide lens
	Mission Param #2	Zoom's absolute position
	Mission Param #3	Zooming step value to offset zoom from th current position
	Mission Param #4	Focus Locking, Unlocking or Re-locking
	Mission Param #5	Shooting Command
	Mission Param #6	Command Identity
	Mission Param #7	Test shot identifier. If set to 1, image will o captured, but not counted towards interna count.
204	MAV_CMD_DO_MOUNT_CONFIGURE	Mission command to configure a camera antenna mount
	Mission Param #1	Mount operation mode (see MAV_MOUNT_MODE enum)
	Mission Param #2	stabilize roll? (1 = yes, 0 = no)
	Mission Param #3	stabilize pitch? (1 = yes, 0 = no)
	Mission Param #4	stabilize yaw? (1 = yes, 0 = no)
	Mission Param #5	roll input (0 = angle, 1 = angular rate)
	Mission Param #6	pitch input (0 = angle, 1 = angular rate)
	Mission Param #7	yaw input (0 = angle, 1 = angular rate)
205	MAV_CMD_DO_MOUNT_CONTROL	Mission command to control a camera or mount
	Mission Param #1	pitch depending on mount mode (degrees degrees/second depending on pitch input
	Mission Param #2	roll depending on mount mode (degrees c degrees/second depending on roll input).
	Mission Param #3	yaw depending on mount mode (degrees degrees/second depending on yaw input)
	Mission Param #4	alt in meters depending on mount mode.
	Mission Param #5	latitude in degrees * 1E7, set if appropriat mode.
	Mission Param #6	longitude in degrees * 1E7, set if appropri mount mode.
	Mission Param #7	MAV_MOUNT_MODE enum value
206	MAV_CMD_DO_SET_CAM_TRIGG_DIST	Mission command to set camera trigger d for this flight. The camera is trigerred eacl this distance is exceeded. This command also be used to set the shutter integration the camera.
	Mission Param #1	Camera trigger distance (meters). 0 to sto triggering.

	Mission Param #1	triggering.
	Mission Param #2	Camera shutter integration time (milliseco or 0 to ignore
	Mission Param #3	Trigger camera once immediately. (0 = no 1 = trigger)
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
207	MAV_CMD_DO_FENCE_ENABLE	Mission command to enable the geofence
	Mission Param #1	enable? (0=disable, 1=enable, 2=disable_floor_only)
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
208	MAV_CMD_DO_PARACHUTE	Mission command to trigger a parachute
	Mission Param #1	action (0=disable, 1=enable, 2=release, fo systems see PARACHUTE_ACTION enu general message set.)
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
209	MAV_CMD_DO_MOTOR_TEST	Mission command to perform motor test
	Mission Param #1	motor number (a number from 1 to max n of motors on the vehicle)
	Mission Param #2	throttle type (0=throttle percentage, 1=PW 2=pilot throttle channel pass-through. See MOTOR_TEST_THROTTLE_TYPE enum
	Mission Param #3	throttle
	Mission Param #4	timeout (in seconds)
		motor count (number of motors to test to t

		them; 0=1 motor, 1=1 motor, 2=2 motors..
	Mission Param #6	motor test order (See MOTOR_TEST_OR enum)
	Mission Param #7	Empty
210	MAV_CMD_DO_INVERTED_FLIGHT	Change to/from inverted flight
	Mission Param #1	inverted (0=normal, 1=inverted)
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
213	MAV_CMD_NAV_SET_YAW_SPEED	Sets a desired vehicle turn angle and spe change
	Mission Param #1	yaw angle to adjust steering by in centide
	Mission Param #2	speed - normalized to 0 .. 1
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
214	MAV_CMD_DO_SET_CAM_TRIGG_INTERVAL	Mission command to set camera trigger ir for this flight. If triggering is enabled, the c is triggered each time this interval expires command can also be used to set the shu integration time for the camera.
	Mission Param #1	Camera trigger cycle time (milliseconds). to ignore.
	Mission Param #2	Camera shutter integration time (milliseco Should be less than trigger cycle time. -1 ignore.
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty

220	MAV_CMD_DO_MOUNT_CONTROL_QUAT	Mission command to control a camera or mount, using a quaternion as reference.
	Mission Param #1	q1 - quaternion param #1, w (1 in null-rota
	Mission Param #2	q2 - quaternion param #2, x (0 in null-rota
	Mission Param #3	q3 - quaternion param #3, y (0 in null-rota
	Mission Param #4	q4 - quaternion param #4, z (0 in null-rota
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
221	MAV_CMD_DO_GUIDED_MASTER	set id of master controller
	Mission Param #1	System ID
	Mission Param #2	Component ID
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
222	MAV_CMD_DO_GUIDED_LIMITS	set limits for external control
	Mission Param #1	timeout - maximum time (in seconds) that controller will be allowed to control vehicle means no timeout
	Mission Param #2	absolute altitude min (in meters, AMSL) - vehicle moves below this alt, the comman aborted and the mission will continue. 0 m lower altitude limit
	Mission Param #3	absolute altitude max (in meters)- if vehicl moves above this alt, the command will b aborted and the mission will continue. 0 m upper altitude limit
	Mission Param #4	horizontal move limit (in meters, AMSL) - vehicle moves more than this distance fro location at the moment the command was executed, the command will be aborted a mission will continue. 0 means no horizon altitude limit
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
		Control vehicle engine. This is interpreted vehicles engine controller to change the ta

		internal combustion engines
	Mission Param #1	0: Stop engine, 1:Start Engine
	Mission Param #2	0: Warm start, 1:Cold start. Controls use c where applicable
	Mission Param #3	Height delay (meters). This is for commar engine start only after the vehicle has gai specified height. Used in VTOL vehicles d takeoff to start engine after the aircraft is c ground. Zero for no delay.
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
240	MAV_CMD_DO_LAST	NOP - This command is only used to marl upper limit of the DO commands in the enumeration
	Mission Param #1	Empty
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
241	MAV_CMD_PREFLIGHT_CALIBRATION	Trigger calibration. This command will be accepted if in pre-flight mode. Except for Temperature Calibration, only one sensor be set in a single message and all others be zero.
	Mission Param #1	1: gyro calibration, 3: gyro temperature ca
	Mission Param #2	1: magnetometer calibration
	Mission Param #3	1: ground pressure calibration
	Mission Param #4	1: radio RC calibration, 2: RC trim calibrat
	Mission Param #5	1: accelerometer calibration, 2: board leve calibration, 3: accelerometer temperature calibration, 4: simple accelerometer calibr
	Mission Param #6	1: APM: compass/motor interference calib (PX4: airspeed calibration, deprecated), 2 airspeed calibration
	Mission Param #7	1: ESC calibration, 3: barometer temperat calibration

242	MAV_CMD_PREFLIGHT_SET_SENSOR_OFFSETS	Set sensor offsets. This command will be accepted if in pre-flight mode.
	Mission Param #1	Sensor to adjust the offsets for: 0: gyros, accelerometer, 2: magnetometer, 3: baror optical flow, 5: second magnetometer, 6: t magnetometer
	Mission Param #2	X axis offset (or generic dimension 1), in t sensor's raw units
	Mission Param #3	Y axis offset (or generic dimension 2), in t sensor's raw units
	Mission Param #4	Z axis offset (or generic dimension 3), in t sensor's raw units
	Mission Param #5	Generic dimension 4, in the sensor's raw
	Mission Param #6	Generic dimension 5, in the sensor's raw
	Mission Param #7	Generic dimension 6, in the sensor's raw
243	MAV_CMD_PREFLIGHT_UAVCAN	Trigger UAVCAN config. This command w only accepted if in pre-flight mode.
	Mission Param #1	1: Trigger actuator ID assignment and dire mapping.
	Mission Param #2	Reserved
	Mission Param #3	Reserved
	Mission Param #4	Reserved
	Mission Param #5	Reserved
	Mission Param #6	Reserved
	Mission Param #7	Reserved
245	MAV_CMD_PREFLIGHT_STORAGE	Request storage of different parameter va and logs. This command will be only acce in pre-flight mode.
	Mission Param #1	Parameter storage: 0: READ FROM FLASH/EEPROM, 1: WRITE CURRENT FLASH/EEPROM, 2: Reset to defaults
	Mission Param #2	Mission storage: 0: READ FROM FLASH/EEPROM, 1: WRITE CURRENT FLASH/EEPROM, 2: Reset to defaults
	Mission Param #3	Onboard logging: 0: Ignore, 1: Start defau logging, -1: Stop logging, > 1: start loggin rate of param 3 in Hz (e.g. set to 1000 for Hz logging)
	Mission Param #4	Reserved
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty

246	MAV_CMD_PREFLIGHT_REBOOT_SHUTDOWN	Request the reboot or shutdown of systen components.
	Mission Param #1	0: Do nothing for autopilot, 1: Reboot autc Shutdown autopilot, 3: Reboot autopilot a it in the bootloader until upgraded.
	Mission Param #2	0: Do nothing for onboard computer, 1: Re onboard computer, 2: Shutdown onboard computer, 3: Reboot onboard computer a it in the bootloader until upgraded.
	Mission Param #3	WIP: 0: Do nothing for camera, 1: Reboot onboard camera, 2: Shutdown onboard ca 3: Reboot onboard camera and keep it in bootloader until upgraded
	Mission Param #4	WIP: 0: Do nothing for mount (e.g. gimbal Reboot mount, 2: Shutdown mount, 3: Re mount and keep it in the bootloader until upgraded
	Mission Param #5	Reserved, send 0
	Mission Param #6	Reserved, send 0
	Mission Param #7	WIP: ID (e.g. camera ID -1 for all IDs)
252	MAV_CMD_OVERRIDE_GOTO	Hold / continue the current action
	Mission Param #1	MAV_GOTO_DO_HOLD: hold MAV_GOTO_DO_CONTINUE: continue v item in mission plan
	Mission Param #2	MAV_GOTO_HOLD_AT_CURRENT_POS Hold at current position MAV_GOTO_HOLD_AT_SPECIFIED_PO hold at specified position
	Mission Param #3	MAV_FRAME coordinate frame of hold po
	Mission Param #4	Desired yaw angle in degrees
	Mission Param #5	Latitude / X position
	Mission Param #6	Longitude / Y position
	Mission Param #7	Altitude / Z position
300	MAV_CMD_MISSION_START	start running a mission
	Mission Param #1	first_item: the first mission item to run
	Mission Param #2	last_item: the last mission item to run (afte item is run, the mission ends)
400	MAV_CMD_COMPONENT_ARM_DISARM	Arms / Disarms a component
	Mission Param #1	1 to arm, 0 to disarm
410	MAV_CMD_GET_HOME_POSITION	Request the home position from the vehic

410	MAV_CMD_GET_HOME_POSITION	Request the home position from the vehic
	Mission Param #1	Reserved
	Mission Param #2	Reserved
	Mission Param #3	Reserved
	Mission Param #4	Reserved
	Mission Param #5	Reserved
	Mission Param #6	Reserved
	Mission Param #7	Reserved
500	MAV_CMD_START_RX_PAIR	Starts receiver pairing
	Mission Param #1	0:Spektrum
	Mission Param #2	RC type (see RC_TYPE enum)
510	MAV_CMD_GET_MESSAGE_INTERVAL	Request the interval between messages f particular MAVLink message ID
	Mission Param #1	The MAVLink message ID
511	MAV_CMD_SET_MESSAGE_INTERVAL	Request the interval between messages f particular MAVLink message ID. This inte replaces REQUEST_DATA_STREAM
	Mission Param #1	The MAVLink message ID
	Mission Param #2	The interval between two messages, in microseconds. Set to -1 to disable and 0 t request default rate.
519	MAV_CMD_REQUEST_PROTOCOL_VERSION	Request MAVLink protocol version compa
	Mission Param #1	1: Request supported protocol versions b nodes on the network
	Mission Param #2	Reserved (all remaining params)
520	MAV_CMD_REQUEST_AUTOPILOT_CAPABILITIES	Request autopilot capabilities
	Mission Param #1	1: Request autopilot version
	Mission Param #2	Reserved (all remaining params)
521	MAV_CMD_REQUEST_CAMERA_INFORMATION	Request camera information (CAMERA_INFORMATION).
	Mission Param #1	0: No action 1: Request camera capabiliti
	Mission Param #2	Reserved (all remaining params)

522	MAV_CMD_REQUEST_CAMERA_SETTINGS	Request camera settings (CAMERA_SET
	Mission Param #1	0: No Action 1: Request camera settings
	Mission Param #2	Reserved (all remaining params)
525	MAV_CMD_REQUEST_STORAGE_INFORMATION	WIP: Request storage information (STORAGE_INFORMATION). Use the command's target_component to target a component's storage.
	Mission Param #1	Storage ID (0 for all, 1 for first, 2 for secor
	Mission Param #2	0: No Action 1: Request storage informati
	Mission Param #3	Reserved (all remaining params)
526	MAV_CMD_STORAGE_FORMAT	WIP: Format a storage medium. Once for complete, a STORAGE_INFORMATION r is sent. Use the command's target_compo target a specific component's storage.
	Mission Param #1	Storage ID (1 for first, 2 for second, etc.)
	Mission Param #2	0: No action 1: Format storage
	Mission Param #3	Reserved (all remaining params)
527	MAV_CMD_REQUEST_CAMERA_CAPTURE_STATUS	Request camera capture status (CAMERA_CAPTURE_STATUS)
	Mission Param #1	0: No Action 1: Request camera capture s
	Mission Param #2	Reserved (all remaining params)
528	MAV_CMD_REQUEST_FLIGHT_INFORMATION	WIP: Request flight information (FLIGHT_INFORMATION)
	Mission Param #1	1: Request flight information
	Mission Param #2	Reserved (all remaining params)
529	MAV_CMD_RESET_CAMERA_SETTINGS	Reset all camera settings to Factory Defa
	Mission Param #1	0: No Action 1: Reset all settings
	Mission Param #2	Reserved (all remaining params)
530	MAV_CMD_SET_CAMERA_MODE	Set camera running mode. Use NAN for r values.
	Mission Param #1	Reserved (Set to 0)
	Mission Param #2	Camera mode (see CAMERA_MODE enu

	Mission Param #2	Camera mode (see CAMERA_MODE enu
	Mission Param #3	Reserved (all remaining params)
2000	MAV_CMD_IMAGE_START_CAPTURE	Start image capture sequence. Sends CAMERA_IMAGE_CAPTURED after eac capture. Use NAN for reserved values.
	Mission Param #1	Reserved (Set to 0)
	Mission Param #2	Duration between two consecutive picture seconds)
	Mission Param #3	Number of images to capture total - 0 for unlimited capture
	Mission Param #4	Reserved (all remaining params)
2001	MAV_CMD_IMAGE_STOP_CAPTURE	Stop image capture sequence Use NAN f reserved values.
	Mission Param #1	Reserved (Set to 0)
	Mission Param #2	Reserved (all remaining params)
2002	MAV_CMD_REQUEST_CAMERA_IMAGE_CAPTURE	WIP: Re-request a CAMERA_IMAGE_CA packet. Use NAN for reserved values.
	Mission Param #1	Sequence number for missing CAMERA_IMAGE_CAPTURE packet
	Mission Param #2	Reserved (all remaining params)
2003	MAV_CMD_DO_TRIGGER_CONTROL	Enable or disable on-board camera trigge system.
	Mission Param #1	Trigger enable/disable (0 for disable, 1 for -1 to ignore
	Mission Param #2	1 to reset the trigger sequence, -1 or 0 to
	Mission Param #3	1 to pause triggering, but without switchin camera off or retracting it. -1 to ignore
2500	MAV_CMD_VIDEO_START_CAPTURE	Starts video capture (recording). Use NAN reserved values.
	Mission Param #1	Reserved (Set to 0)
	Mission Param #2	Frequency CAMERA_CAPTURE_STATU messages should be sent while recording no messages, otherwise frequency in Hz)
	Mission Param #3	Reserved (all remaining params)
2501	MAV_CMD_VIDEO_STOP_CAPTURE	Stop the current video capture (recording) NAN for reserved values.

	Mission Param #1	Reserved (Set to 0)
	Mission Param #2	Reserved (all remaining params)
2502	MAV_CMD_VIDEO_START_STREAMING	WIP: Start video streaming
	Mission Param #1	Camera ID (0 for all cameras, 1 for first, 2 second, etc.)
	Mission Param #2	Reserved
2503	MAV_CMD_VIDEO_STOP_STREAMING	WIP: Stop the current video streaming
	Mission Param #1	Camera ID (0 for all cameras, 1 for first, 2 second, etc.)
	Mission Param #2	Reserved
2504	MAV_CMD_REQUEST_VIDEO_STREAM_INFORMATION	WIP: Request video stream information (VIDEO_STREAM_INFORMATION)
	Mission Param #1	Camera ID (0 for all cameras, 1 for first, 2 second, etc.)
	Mission Param #2	0: No Action 1: Request video stream info
	Mission Param #3	Reserved (all remaining params)
2510	MAV_CMD_LOGGING_START	Request to start streaming logging data o MAVLink (see also LOGGING_DATA mes
	Mission Param #1	Format: 0: ULog
	Mission Param #2	Reserved (set to 0)
	Mission Param #3	Reserved (set to 0)
	Mission Param #4	Reserved (set to 0)
	Mission Param #5	Reserved (set to 0)
	Mission Param #6	Reserved (set to 0)
	Mission Param #7	Reserved (set to 0)
2511	MAV_CMD_LOGGING_STOP	Request to stop streaming log data over M
	Mission Param #1	Reserved (set to 0)
	Mission Param #2	Reserved (set to 0)
	Mission Param #3	Reserved (set to 0)
	Mission Param #4	Reserved (set to 0)
	Mission Param #5	Reserved (set to 0)
	Mission Param #6	Reserved (set to 0)
	Mission Param #7	Reserved (set to 0)

	Mission Param #7	Reserved (set to 0)
2520	MAV_CMD_AIRFRAME_CONFIGURATION	
	Mission Param #1	Landing gear ID (default: 0, -1 for all)
	Mission Param #2	Landing gear position (Down: 0, Up: 1, NA no change)
	Mission Param #3	Reserved, set to NAN
	Mission Param #4	Reserved, set to NAN
	Mission Param #5	Reserved, set to NAN
	Mission Param #6	Reserved, set to NAN
	Mission Param #7	Reserved, set to NAN
2600	MAV_CMD_CONTROL_HIGH_LATENCY	Request to start/stop transmitting over the latency telemetry
	Mission Param #1	Control transmittion over high latency tele (0: stop, 1: start)
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
2800	MAV_CMD_PANORAMA_CREATE	Create a panorama at the current position
	Mission Param #1	Viewing angle horizontal of the panorama degrees, +- 0.5 the total angle)
	Mission Param #2	Viewing angle vertical of panorama (in de
	Mission Param #3	Speed of the horizontal rotation (in degree second)
	Mission Param #4	Speed of the vertical rotation (in degrees second)
3000	MAV_CMD_DO_VTOL_TRANSITION	Request VTOL transition
	Mission Param #1	The target VTOL state, as defined by ENU MAV_VTOL_STATE. Only MAV_VTOL_STATE_MC and MAV_VTOL_STATE_FW can be used.
4000	MAV_CMD_SET_GUIDED_SUBMODE_STANDARD	This command sets the submode to stand guided when vehicle is in guided mode. T vehicle holds position and altitude and the

4001	MAV_CMD_SET_GUIDED_SUBMODE_CIRCLE	This command sets submode circle when is in guided mode. Vehicle flies along a ci facing the center of the circle. The user ca the velocity along the circle and change th radius. If no input is given the vehicle will position.
	Mission Param #1	Radius of desired circle in CIRCLE_MOD
	Mission Param #2	User defined
	Mission Param #3	User defined
	Mission Param #4	User defined
	Mission Param #5	Unscaled target latitude of center of circle CIRCLE_MODE
	Mission Param #6	Unscaled target longitude of center of circ CIRCLE_MODE
4501	MAV_CMD_CONDITION_GATE	WIP: Delay mission state machine until ga been reached.
	Mission Param #1	Geometry: 0: orthogonal to path between previous and next waypoint.
	Mission Param #2	Altitude: 0: ignore altitude
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Altitude
3001	MAV_CMD_ARM_AUTHORIZATION_REQUEST	Request authorization to arm the vehicle t external entity, the arm authorizer is respo request all data that is needs from the veh before authorize or deny the request. If ap the progress of command_ack message s be set with period of time that this authori valid in seconds or in case it was denied i be set with one of the reasons in ARM_AUTH_DENIED_REASON.
	Mission Param #1	Vehicle system id, this way ground station request arm authorization on behalf of any
5000	MAV_CMD_NAV_FENCE_RETURN_POINT	Fence return point. There can only be one return point.
	Mission Param #1	Reserved
	Mission Param #2	Reserved
	Mission Param #3	Reserved
	Mission Param #4	Reserved

	Mission Param #4	Reserved
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Altitude
5001	MAV_CMD_NAV_FENCE_POLYGON_VERTEX_INCLUSION	Fence vertex for an inclusion polygon (the polygon must not be self-intersecting). Th vehicle must stay within this area. Minimu vertices required.
	Mission Param #1	Polygon vertex count
	Mission Param #2	Reserved
	Mission Param #3	Reserved
	Mission Param #4	Reserved
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Reserved
5002	MAV_CMD_NAV_FENCE_POLYGON_VERTEX_EXCLUSION	Fence vertex for an exclusion polygon (th polygon must not be self-intersecting). Th vehicle must stay outside this area. Minim vertices required.
	Mission Param #1	Polygon vertex count
	Mission Param #2	Reserved
	Mission Param #3	Reserved
	Mission Param #4	Reserved
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Reserved
5003	MAV_CMD_NAV_FENCE_CIRCLE_INCLUSION	Circular fence area. The vehicle must stay this area.
	Mission Param #1	radius in meters
	Mission Param #2	Reserved
	Mission Param #3	Reserved
	Mission Param #4	Reserved
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Reserved
		Circular fence area. The vehicle must stay

	Mission Param #1	radius in meters
	Mission Param #2	Reserved
	Mission Param #3	Reserved
	Mission Param #4	Reserved
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Reserved
5100	MAV_CMD_NAV_RALLY_POINT	Rally point. You can have multiple rally po defined.
	Mission Param #1	Reserved
	Mission Param #2	Reserved
	Mission Param #3	Reserved
	Mission Param #4	Reserved
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Altitude
5200	MAV_CMD_UAVCAN_GET_NODE_INFO	Commands the vehicle to respond with a sequence of messages UAVCAN_NODE_ one message per every UAVCAN node th online. Note that some of the response m can be lost, which the receiver can detect by checking whether every received UAVCAN_NODE_STATUS has a matchin message UAVCAN_NODE_INFO receive earlier; if not, this command should be se in order to request re-transmission of the information messages.
	Mission Param #1	Reserved (set to 0)
	Mission Param #2	Reserved (set to 0)
	Mission Param #3	Reserved (set to 0)
	Mission Param #4	Reserved (set to 0)
	Mission Param #5	Reserved (set to 0)
	Mission Param #6	Reserved (set to 0)
	Mission Param #7	Reserved (set to 0)
30001	MAV_CMD_PAYLOAD_PREPARE_DEPLOY	Deploy payload on a Lat / Lon / Alt positio includes the navigation to reach the requi release position and velocity.
	Mission Param #1	Operation mode. 0: prepare single payloa (overwriting previous requests), but do no execute it. 1: execute payload deploy imm

	Mission Param #1	(overwriting previous requests), but do no execute it. 1: execute payload deploy imm (rejecting further deploy commands during execution, but allowing abort). 2: add payl deploy to existing deployment list.
	Mission Param #2	Desired approach vector in degrees comp heading (0..360). A negative value indicat system can define the approach vector at
	Mission Param #3	Desired ground speed at release time. Th be overriden by the airframe in case it nee meet minimum airspeed. A negative value indicates the system can define the groun at will.
	Mission Param #4	Minimum altitude clearance to the release position in meters. A negative value indica system can define the clearance at will.
	Mission Param #5	Latitude unscaled for MISSION_ITEM or i degrees for MISSION_ITEM_INT
	Mission Param #6	Longitude unscaled for MISSION_ITEM o degrees for MISSION_ITEM_INT
	Mission Param #7	Altitude, in meters AMSL
30002	MAV_CMD_PAYLOAD_CONTROL_DEPLOY	Control the payload deployment.
	Mission Param #1	Operation mode. 0: Abort deployment, co normal mission. 1: switch to payload depl mode. 100: delete first payload deployme request. 101: delete all payload deployme requests.
	Mission Param #2	Reserved
	Mission Param #3	Reserved
	Mission Param #4	Reserved
	Mission Param #5	Reserved
	Mission Param #6	Reserved
	Mission Param #7	Reserved
31000	MAV_CMD_WAYPOINT_USER_1	User defined waypoint item. Ground Stati show the Vehicle as flying through this ite
	Mission Param #1	User defined
	Mission Param #2	User defined
	Mission Param #3	User defined
	Mission Param #4	User defined
	Mission Param #5	Latitude unscaled
	Mission Param #6	Longitude unscaled
	Mission Param #7	Altitude, in meters AMSL
		User defined waypoint item. Ground Stati

	Mission Param #1	User defined
	Mission Param #2	User defined
	Mission Param #3	User defined
	Mission Param #4	User defined
	Mission Param #5	Latitude unscaled
	Mission Param #6	Longitude unscaled
	Mission Param #7	Altitude, in meters AMSL
31002	MAV_CMD_WAYPOINT_USER_3	User defined waypoint item. Ground Stati show the Vehicle as flying through this ite
	Mission Param #1	User defined
	Mission Param #2	User defined
	Mission Param #3	User defined
	Mission Param #4	User defined
	Mission Param #5	Latitude unscaled
	Mission Param #6	Longitude unscaled
	Mission Param #7	Altitude, in meters AMSL
31003	MAV_CMD_WAYPOINT_USER_4	User defined waypoint item. Ground Stati show the Vehicle as flying through this ite
	Mission Param #1	User defined
	Mission Param #2	User defined
	Mission Param #3	User defined
	Mission Param #4	User defined
	Mission Param #5	Latitude unscaled
	Mission Param #6	Longitude unscaled
	Mission Param #7	Altitude, in meters AMSL
31004	MAV_CMD_WAYPOINT_USER_5	User defined waypoint item. Ground Stati show the Vehicle as flying through this ite
	Mission Param #1	User defined
	Mission Param #2	User defined
	Mission Param #3	User defined
	Mission Param #4	User defined
	Mission Param #5	Latitude unscaled
	Mission Param #6	Longitude unscaled
	Mission Param #7	Altitude, in meters AMSL

	Mission Param #7	Altitude, in meters AMSL
31005	MAV_CMD_SPATIAL_USER_1	User defined spatial item. Ground Station show the Vehicle as flying through this ite Example: ROI item.
	Mission Param #1	User defined
	Mission Param #2	User defined
	Mission Param #3	User defined
	Mission Param #4	User defined
	Mission Param #5	Latitude unscaled
	Mission Param #6	Longitude unscaled
	Mission Param #7	Altitude, in meters AMSL
31006	MAV_CMD_SPATIAL_USER_2	User defined spatial item. Ground Station show the Vehicle as flying through this ite Example: ROI item.
	Mission Param #1	User defined
	Mission Param #2	User defined
	Mission Param #3	User defined
	Mission Param #4	User defined
	Mission Param #5	Latitude unscaled
	Mission Param #6	Longitude unscaled
	Mission Param #7	Altitude, in meters AMSL
31007	MAV_CMD_SPATIAL_USER_3	User defined spatial item. Ground Station show the Vehicle as flying through this ite Example: ROI item.
	Mission Param #1	User defined
	Mission Param #2	User defined
	Mission Param #3	User defined
	Mission Param #4	User defined
	Mission Param #5	Latitude unscaled
	Mission Param #6	Longitude unscaled
	Mission Param #7	Altitude, in meters AMSL
31008	MAV_CMD_SPATIAL_USER_4	User defined spatial item. Ground Station show the Vehicle as flying through this ite Example: ROI item.
	Mission Param #1	User defined

	Mission Param #3	User defined
	Mission Param #4	User defined
	Mission Param #5	Latitude unscaled
	Mission Param #6	Longitude unscaled
	Mission Param #7	Altitude, in meters AMSL
31009	MAV_CMD_SPATIAL_USER_5	User defined spatial item. Ground Station show the Vehicle as flying through this ite Example: ROI item.
	Mission Param #1	User defined
	Mission Param #2	User defined
	Mission Param #3	User defined
	Mission Param #4	User defined
	Mission Param #5	Latitude unscaled
	Mission Param #6	Longitude unscaled
	Mission Param #7	Altitude, in meters AMSL
31010	MAV_CMD_USER_1	User defined command. Ground Station w show the Vehicle as flying through this ite Example: MAV_CMD_DO_SET_PARAME item.
	Mission Param #1	User defined
	Mission Param #2	User defined
	Mission Param #3	User defined
	Mission Param #4	User defined
	Mission Param #5	User defined
	Mission Param #6	User defined
	Mission Param #7	User defined
31011	MAV_CMD_USER_2	User defined command. Ground Station w show the Vehicle as flying through this ite Example: MAV_CMD_DO_SET_PARAME item.
	Mission Param #1	User defined
	Mission Param #2	User defined
	Mission Param #3	User defined
	Mission Param #4	User defined
	Mission Param #5	User defined
	Mission Param #6	User defined
	Mission Param #7	User defined

	Mission Param #7	User defined
31012	MAV_CMD_USER_3	User defined command. Ground Station w show the Vehicle as flying through this ite Example: MAV_CMD_DO_SET_PARAME item.
	Mission Param #1	User defined
	Mission Param #2	User defined
	Mission Param #3	User defined
	Mission Param #4	User defined
	Mission Param #5	User defined
	Mission Param #6	User defined
	Mission Param #7	User defined
31013	MAV_CMD_USER_4	User defined command. Ground Station w show the Vehicle as flying through this ite Example: MAV_CMD_DO_SET_PARAME item.
	Mission Param #1	User defined
	Mission Param #2	User defined
	Mission Param #3	User defined
	Mission Param #4	User defined
	Mission Param #5	User defined
	Mission Param #6	User defined
	Mission Param #7	User defined
31014	MAV_CMD_USER_5	User defined command. Ground Station w show the Vehicle as flying through this ite Example: MAV_CMD_DO_SET_PARAME item.
	Mission Param #1	User defined
	Mission Param #2	User defined
	Mission Param #3	User defined
	Mission Param #4	User defined
	Mission Param #5	User defined
	Mission Param #6	User defined
	Mission Param #7	User defined

MAV_DATA_STREAM

THIS INTERFACE IS DEPRECATED AS OF JULY 2015. Please use MESSAGE_INTERVAL instead. A data stream is not a fixed set of messages, but rather a recommendation to the autopilot software. Individual autopilots may or may not obey the recommended messages.

CMD ID	Field Name	Description
0	MAV_DATA_STREAM_ALL	Enable all data streams
1	MAV_DATA_STREAM_RAW_SENSORS	Enable IMU_RAW, GPS_RAW, GPS_STATUS packets.
2	MAV_DATA_STREAM_EXTENDED_STATUS	Enable GPS_STATUS, CONTROL_STATUS, AUX_STATUS
3	MAV_DATA_STREAM_RC_CHANNELS	Enable RC_CHANNELS_SCALED, RC_CHANNELS_RAW, SERVO_OUTPUT_RAW
4	MAV_DATA_STREAM_RAW_CONTROLLER	Enable ATTITUDE_CONTROLLER_OUTPUT, POSITION_CONTROLLER_OUTPUT, NAV_CONTROLLER_OUTPUT.
6	MAV_DATA_STREAM_POSITION	Enable LOCAL_POSITION, GLOBAL_POSITION/GLOBAL_POSITION_INT messages.
10	MAV_DATA_STREAM_EXTRA1	Dependent on the autopilot
11	MAV_DATA_STREAM_EXTRA2	Dependent on the autopilot
12	MAV_DATA_STREAM_EXTRA3	Dependent on the autopilot

MAV_ROI

THIS INTERFACE IS DEPRECATED AS OF JANUARY 2018. Please use MAV_CMD_DO_SET_ROI_* messages instead. The ROI (region of interest) for the vehicle. This can be be used by the vehicle for camera/vehicle attitude alignment (see MAV_CMD_NAV_ROI).

CMD ID	Field Name	Description
0	MAV_ROI_NONE	No region of interest.
1	MAV_ROI_WPNEXT	Point toward next waypoint, with optional pitch/roll/yaw offset.
2	MAV_ROI_WPINDEX	Point toward given waypoint.
3	MAV_ROI_LOCATION	Point toward fixed location.
4	MAV_ROI_TARGET	Point toward of given id.

MAV_CMD_ACK

ACK / NACK / ERROR values as a result of MAV_CMDs and for mission item transmission.

CMD ID	Field Name	Description
	MAV_CMD_ACK_OK	Command / mission item is ok.
	MAV_CMD_ACK_ERR_FAIL	Generic error message if none of the other reasons fails or if no detailed error reporting is implemented.
	MAV_CMD_ACK_ERR_ACCESS_DENIED	The system is refusing to accept this command from this source / communication partner.
	MAV_CMD_ACK_ERR_NOT_SUPPORTED	Command or mission item is not supported, other commands would be accepted.
	MAV_CMD_ACK_ERR_COORDINATE_FRAME_NOT_SUPPORTED	The coordinate frame of this command / mission item is not supported.
	MAV_CMD_ACK_ERR_COORDINATES_OUT_OF_RANGE	The coordinate frame of this command is ok, but he coordinate values exceed the safety limits of this system. This is a generic error, please use the more specific error messages below if possible.
	MAV_CMD_ACK_ERR_X_LAT_OUT_OF_RANGE	The X or latitude value is out of range.
	MAV_CMD_ACK_ERR_Y_LON_OUT_OF_RANGE	The Y or longitude value is out of range.
	MAV_CMD_ACK_ERR_Z_ALT_OUT_OF_RANGE	The Z or altitude value is out of range.

MAV_PARAM_TYPE

Specifies the datatype of a MAVLink parameter.

CMD ID	Field Name	Description
1	MAV_PARAM_TYPE_UINT8	8-bit unsigned integer
2	MAV_PARAM_TYPE_INT8	8-bit signed integer
3	MAV_PARAM_TYPE_UINT16	16-bit unsigned integer
4	MAV_PARAM_TYPE_INT16	16-bit signed integer
5	MAV_PARAM_TYPE_UINT32	32-bit unsigned integer
6	MAV_PARAM_TYPE_INT32	32-bit signed integer
7	MAV_PARAM_TYPE_UINT64	64-bit unsigned integer
8	MAV_PARAM_TYPE_INT64	64-bit signed integer
9	MAV_PARAM_TYPE_REAL32	32-bit floating-point
10	MAV_PARAM_TYPE_REAL64	64-bit floating-point

MAV_PARAM_EXT_TYPE

Specifies the datatype of a MAVLink extended parameter.

CMD ID	Field Name	Description
1	MAV_PARAM_EXT_TYPE_UINT8	8-bit unsigned integer
2	MAV_PARAM_EXT_TYPE_INT8	8-bit signed integer
3	MAV_PARAM_EXT_TYPE_UINT16	16-bit unsigned integer
4	MAV_PARAM_EXT_TYPE_INT16	16-bit signed integer
5	MAV_PARAM_EXT_TYPE_UINT32	32-bit unsigned integer
6	MAV_PARAM_EXT_TYPE_INT32	32-bit signed integer
7	MAV_PARAM_EXT_TYPE_UINT64	64-bit unsigned integer
8	MAV_PARAM_EXT_TYPE_INT64	64-bit signed integer
9	MAV_PARAM_EXT_TYPE_REAL32	32-bit floating-point
10	MAV_PARAM_EXT_TYPE_REAL64	64-bit floating-point
11	MAV_PARAM_EXT_TYPE_CUSTOM	Custom Type

MAV_RESULT

result from a mavlink command

CMD ID	Field Name	Description
0	MAV_RESULT_ACCEPTED	Command ACCEPTED and EXECUTED
1	MAV_RESULT_TEMPORARILY_REJECTED	Command TEMPORARY REJECTED/DENIED
2	MAV_RESULT_DENIED	Command PERMANENTLY DENIED
3	MAV_RESULT_UNSUPPORTED	Command UNKNOWN/UNSUPPORTED
4	MAV_RESULT_FAILED	Command executed, but failed
5	MAV_RESULT_IN_PROGRESS	WIP: Command being executed

MAV_MISSION_RESULT

result in a mavlink mission ack

CMD ID	Field Name	Description
0	MAV_MISSION_ACCEPTED	mission accepted OK
1	MAV_MISSION_ERROR	generic error / not accepting mission commands at all right now
2	MAV_MISSION_UNSUPPORTED_FRAME	coordinate frame is not supported
3	MAV_MISSION_UNSUPPORTED	command is not supported
4	MAV_MISSION_NO_SPACE	mission item exceeds storage space
5	MAV_MISSION_INVALID	one of the parameters has an invalid value
6	MAV_MISSION_INVALID_PARAM1	param1 has an invalid value
7	MAV_MISSION_INVALID_PARAM2	param2 has an invalid value
8	MAV_MISSION_INVALID_PARAM3	param3 has an invalid value
9	MAV_MISSION_INVALID_PARAM4	param4 has an invalid value
10	MAV_MISSION_INVALID_PARAM5_X	x/param5 has an invalid value
11	MAV_MISSION_INVALID_PARAM6_Y	y/param6 has an invalid value
12	MAV_MISSION_INVALID_PARAM7	param7 has an invalid value
13	MAV_MISSION_INVALID_SEQUENCE	received waypoint out of sequence
14	MAV_MISSION_DENIED	not accepting any mission commands from this communication partner

MAV_SEVERITY

Indicates the severity level, generally used for status messages to indicate their relative urgency. Based on RFC-5424 using expanded definitions at: http://www.kiwisyslog.com/kb/info:-syslog-message-levels/.

CMD ID	Field Name	Description
0	MAV_SEVERITY_EMERGENCY	System is unusable. This is a "panic" condition.
1	MAV_SEVERITY_ALERT	Action should be taken immediately. Indicates error in non-critical systems.
2	MAV_SEVERITY_CRITICAL	Action must be taken immediately. Indicates failure in a primary system.
3	MAV_SEVERITY_ERROR	Indicates an error in secondary/redundant systems.
4	MAV_SEVERITY_WARNING	Indicates about a possible future error if this is not resolved within a given timeframe. Example would be a low battery warning.
5	MAV_SEVERITY_NOTICE	An unusual event has occured, though not an error condition. This should be investigated for the root cause.
6	MAV_SEVERITY_INFO	Normal operational messages. Useful for logging. No action is required for these messages.
7	MAV_SEVERITY_DEBUG	Useful non-operational messages that can assist in debugging. These should not occur during normal operation.

MAV_POWER_STATUS

Power supply status flags (bitmask)

CMD ID	Field Name	Description
1	MAV_POWER_STATUS_BRICK_VALID	main brick power supply valid
2	MAV_POWER_STATUS_SERVO_VALID	main servo power supply valid for FMU
4	MAV_POWER_STATUS_USB_CONNECTED	USB power is connected
8	MAV_POWER_STATUS_PERIPH_OVERCURRENT	peripheral supply is in over-current state
16	MAV_POWER_STATUS_PERIPH_HIPOWER_OVERCURRENT	hi-power peripheral supply is in over-current state
32	MAV_POWER_STATUS_CHANGED	Power status has changed since boot

SERIAL_CONTROL_DEV

SERIAL_CONTROL device types

CMD ID	Field Name	Description
0	SERIAL_CONTROL_DEV_TELEM1	First telemetry port
1	SERIAL_CONTROL_DEV_TELEM2	Second telemetry port
2	SERIAL_CONTROL_DEV_GPS1	First GPS port
3	SERIAL_CONTROL_DEV_GPS2	Second GPS port
10	SERIAL_CONTROL_DEV_SHELL	system shell

SERIAL_CONTROL_FLAG

SERIAL_CONTROL flags (bitmask)

CMD ID	Field Name	Description
1	SERIAL_CONTROL_FLAG_REPLY	Set if this is a reply
2	SERIAL_CONTROL_FLAG_RESPOND	Set if the sender wants the receiver to send a response as another SERIAL_CONTROL message
4	SERIAL_CONTROL_FLAG_EXCLUSIVE	Set if access to the serial port should be removed from whatever driver is currently using it, giving exclusive access to the SERIAL_CONTROL protocol. The port can be handed back by sending a request without this flag set
8	SERIAL_CONTROL_FLAG_BLOCKING	Block on writes to the serial port
16	SERIAL_CONTROL_FLAG_MULTI	Send multiple replies until port is drained

MAV_DISTANCE_SENSOR

Enumeration of distance sensor types

CMD ID	Field Name	Description
0	MAV_DISTANCE_SENSOR_LASER	Laser rangefinder, e.g. LightWare SF02/F or PulsedLight units
1	MAV_DISTANCE_SENSOR_ULTRASOUND	Ultrasound rangefinder, e.g. MaxBotix units
2	MAV_DISTANCE_SENSOR_INFRARED	Infrared rangefinder, e.g. Sharp units
3	MAV_DISTANCE_SENSOR_RADAR	Radar type, e.g. uLanding units
4	MAV_DISTANCE_SENSOR_UNKNOWN	Broken or unknown type, e.g. analog units

MAV_SENSOR_ORIENTATION

Enumeration of sensor orientation, according to its rotations

CMD ID	Field Name	Description
0	MAV_SENSOR_ROTATION_NONE	Roll: 0, Pitch: 0, Yaw: 0
1	MAV_SENSOR_ROTATION_YAW_45	Roll: 0, Pitch: 0, Yaw: 45
2	MAV_SENSOR_ROTATION_YAW_90	Roll: 0, Pitch: 0, Yaw: 90
3	MAV_SENSOR_ROTATION_YAW_135	Roll: 0, Pitch: 0, Yaw: 135
4	MAV_SENSOR_ROTATION_YAW_180	Roll: 0, Pitch: 0, Yaw: 180
5	MAV_SENSOR_ROTATION_YAW_225	Roll: 0, Pitch: 0, Yaw: 225
6	MAV_SENSOR_ROTATION_YAW_270	Roll: 0, Pitch: 0, Yaw: 270
7	MAV_SENSOR_ROTATION_YAW_315	Roll: 0, Pitch: 0, Yaw: 315
8	MAV_SENSOR_ROTATION_ROLL_180	Roll: 180, Pitch: 0, Yaw: 0
9	MAV_SENSOR_ROTATION_ROLL_180_YAW_45	Roll: 180, Pitch: 0, Yaw: 45
10	MAV_SENSOR_ROTATION_ROLL_180_YAW_90	Roll: 180, Pitch: 0, Yaw: 90
11	MAV_SENSOR_ROTATION_ROLL_180_YAW_135	Roll: 180, Pitch: 0, Yaw: 135
12	MAV_SENSOR_ROTATION_PITCH_180	Roll: 0, Pitch: 180, Yaw: 0
13	MAV_SENSOR_ROTATION_ROLL_180_YAW_225	Roll: 180, Pitch: 0, Yaw: 225
14	MAV_SENSOR_ROTATION_ROLL_180_YAW_270	Roll: 180, Pitch: 0, Yaw: 270
15	MAV_SENSOR_ROTATION_ROLL_180_YAW_315	Roll: 180, Pitch: 0, Yaw: 315
16	MAV_SENSOR_ROTATION_ROLL_90	Roll: 90, Pitch: 0, Yaw: 0
17	MAV_SENSOR_ROTATION_ROLL_90_YAW_45	Roll: 90, Pitch: 0, Yaw: 45
18	MAV_SENSOR_ROTATION_ROLL_90_YAW_90	Roll: 90, Pitch: 0, Yaw: 90
19	MAV_SENSOR_ROTATION_ROLL_90_YAW_135	Roll: 90, Pitch: 0, Yaw: 135
20	MAV_SENSOR_ROTATION_ROLL_270	Roll: 270, Pitch: 0, Yaw: 0
21	MAV_SENSOR_ROTATION_ROLL_270_YAW_45	Roll: 270, Pitch: 0, Yaw: 45
22	MAV_SENSOR_ROTATION_ROLL_270_YAW_90	Roll: 270, Pitch: 0, Yaw: 90
23	MAV_SENSOR_ROTATION_ROLL_270_YAW_135	Roll: 270, Pitch: 0, Yaw: 135
24	MAV_SENSOR_ROTATION_PITCH_90	Roll: 0, Pitch: 90, Yaw: 0
25	MAV_SENSOR_ROTATION_PITCH_270	Roll: 0, Pitch: 270, Yaw: 0

26	MAV_SENSOR_ROTATION_PITCH_180_YAW_90	Roll: 0, Pitch: 180, Yaw: 90
27	MAV_SENSOR_ROTATION_PITCH_180_YAW_270	Roll: 0, Pitch: 180, Yaw: 270
28	MAV_SENSOR_ROTATION_ROLL_90_PITCH_90	Roll: 90, Pitch: 90, Yaw: 0
29	MAV_SENSOR_ROTATION_ROLL_180_PITCH_90	Roll: 180, Pitch: 90, Yaw: 0
30	MAV_SENSOR_ROTATION_ROLL_270_PITCH_90	Roll: 270, Pitch: 90, Yaw: 0
31	MAV_SENSOR_ROTATION_ROLL_90_PITCH_180	Roll: 90, Pitch: 180, Yaw: 0
32	MAV_SENSOR_ROTATION_ROLL_270_PITCH_180	Roll: 270, Pitch: 180, Yaw: 0
33	MAV_SENSOR_ROTATION_ROLL_90_PITCH_270	Roll: 90, Pitch: 270, Yaw: 0
34	MAV_SENSOR_ROTATION_ROLL_180_PITCH_270	Roll: 180, Pitch: 270, Yaw: 0
35	MAV_SENSOR_ROTATION_ROLL_270_PITCH_270	Roll: 270, Pitch: 270, Yaw: 0
36	MAV_SENSOR_ROTATION_ROLL_90_PITCH_180_YAW_90	Roll: 90, Pitch: 180, Yaw: 90
37	MAV_SENSOR_ROTATION_ROLL_90_YAW_270	Roll: 90, Pitch: 0, Yaw: 270
38	MAV_SENSOR_ROTATION_ROLL_315_PITCH_315_YAW_315	Roll: 315, Pitch: 315, Yaw: 315

MAV_PROTOCOL_CAPABILITY

Bitmask of (optional) autopilot capabilities (64 bit). If a bit is set, the autopilot supports this capability.

CMD ID	Field Name	Description
1	MAV_PROTOCOL_CAPABILITY_MISSION_FLOAT	Autopilot supports MISSION float message type.
2	MAV_PROTOCOL_CAPABILITY_PARAM_FLOAT	Autopilot supports the new param float message type.
4	MAV_PROTOCOL_CAPABILITY_MISSION_INT	Autopilot supports MISSION_INT scaled intege message type.
8	MAV_PROTOCOL_CAPABILITY_COMMAND_INT	Autopilot supports COMMAND_INT scaled inte message type.
16	MAV_PROTOCOL_CAPABILITY_PARAM_UNION	Autopilot supports the new param union message type.
32	MAV_PROTOCOL_CAPABILITY_FTP	Autopilot supports the new FILE_TRANSFER_PROTOC message type.
64	MAV_PROTOCOL_CAPABILITY_SET_ATTITUDE_TARGET	Autopilot supports command attitude offboard.
128	MAV_PROTOCOL_CAPABILITY_SET_POSITION_TARGET_LOCAL_NED	Autopilot supports command position and velocity targets local NED frame.
256	MAV_PROTOCOL_CAPABILITY_SET_POSITION_TARGET_GLOBAL_INT	Autopilot supports command position and velocity targets global scaled integers.
512	MAV_PROTOCOL_CAPABILITY_TERRAIN	Autopilot supports terrain protocol / data handling.
1024	MAV_PROTOCOL_CAPABILITY_SET_ACTUATOR_TARGET	Autopilot supports direct actuator control.
2048	MAV_PROTOCOL_CAPABILITY_FLIGHT_TERMINATION	Autopilot supports the flight termination command.
4096	MAV_PROTOCOL_CAPABILITY_COMPASS_CALIBRATION	Autopilot supports onboard compass calibration.
8192	MAV_PROTOCOL_CAPABILITY_MAVLINK2	Autopilot supports mavlink version 2.
16384	MAV_PROTOCOL_CAPABILITY_MISSION_FENCE	Autopilot supports mission fence protocol.
32768	MAV_PROTOCOL_CAPABILITY_MISSION_RALLY	Autopilot supports mission ra point protocol.
65536	MAV_PROTOCOL_CAPABILITY_FLIGHT_INFORMATION	Autopilot supports the flight information protocol.

MAV_MISSION_TYPE

Type of mission items being requested/sent in mission protocol.

CMD ID	Field Name	Description
0	MAV_MISSION_TYPE_MISSION	Items are mission commands for main mission.
1	MAV_MISSION_TYPE_FENCE	Specifies GeoFence area(s). Items are MAV_CMD_FENCE_ GeoFence items.
2	MAV_MISSION_TYPE_RALLY	Specifies the rally points for the vehicle. Rally points are alternative RTL points. Items are MAV_CMD_RALLY_POINT rally point items.
255	MAV_MISSION_TYPE_ALL	Only used in MISSION_CLEAR_ALL to clear all mission types.

MAV_ESTIMATOR_TYPE

Enumeration of estimator types

CMD ID	Field Name	Description
1	MAV_ESTIMATOR_TYPE_NAIVE	This is a naive estimator without any real covariance feedback.
2	MAV_ESTIMATOR_TYPE_VISION	Computer vision based estimate. Might be up to scale.
3	MAV_ESTIMATOR_TYPE_VIO	Visual-inertial estimate.
4	MAV_ESTIMATOR_TYPE_GPS	Plain GPS estimate.
5	MAV_ESTIMATOR_TYPE_GPS_INS	Estimator integrating GPS and inertial sensing.

MAV_BATTERY_TYPE

Enumeration of battery types

CMD ID	Field Name	Description
0	MAV_BATTERY_TYPE_UNKNOWN	Not specified.
1	MAV_BATTERY_TYPE_LIPO	Lithium polymer battery
2	MAV_BATTERY_TYPE_LIFE	Lithium-iron-phosphate battery
3	MAV_BATTERY_TYPE_LION	Lithium-ION battery
4	MAV_BATTERY_TYPE_NIMH	Nickel metal hydride battery

MAV_BATTERY_FUNCTION

Enumeration of battery functions

CMD ID	Field Name	Description
0	MAV_BATTERY_FUNCTION_UNKNOWN	Battery function is unknown
1	MAV_BATTERY_FUNCTION_ALL	Battery supports all flight systems
2	MAV_BATTERY_FUNCTION_PROPULSION	Battery for the propulsion system
3	MAV_BATTERY_FUNCTION_AVIONICS	Avionics battery
4	MAV_BATTERY_TYPE_PAYLOAD	Payload battery

MAV_BATTERY_CHARGE_STATE

Enumeration for low battery states.

CMD ID	Field Name	Description
0	MAV_BATTERY_CHARGE_STATE_UNDEFINED	Low battery state is not provided
1	MAV_BATTERY_CHARGE_STATE_OK	Battery is not in low state. Normal operation.
2	MAV_BATTERY_CHARGE_STATE_LOW	Battery state is low, warn and monitor close.
3	MAV_BATTERY_CHARGE_STATE_CRITICAL	Battery state is critical, return or abort immediately.
4	MAV_BATTERY_CHARGE_STATE_EMERGENCY	Battery state is too low for ordinary abort sequence. Perform fastest possible emergency stop to prevent damage.
5	MAV_BATTERY_CHARGE_STATE_FAILED	Battery failed, damage unavoidable.
6	MAV_BATTERY_CHARGE_STATE_UNHEALTHY	Battery is diagnosed to be defective or an error occurred, usage is discouraged / prohibited.

MAV_VTOL_STATE

Enumeration of VTOL states

CMD ID	Field Name	Description
0	MAV_VTOL_STATE_UNDEFINED	MAV is not configured as VTOL
1	MAV_VTOL_STATE_TRANSITION_TO_FW	VTOL is in transition from multicopter to fixed-wing
2	MAV_VTOL_STATE_TRANSITION_TO_MC	VTOL is in transition from fixed-wing to multicopter
3	MAV_VTOL_STATE_MC	VTOL is in multicopter state
4	MAV_VTOL_STATE_FW	VTOL is in fixed-wing state

MAV_LANDED_STATE

Enumeration of landed detector states

CMD ID	Field Name	Description
0	MAV_LANDED_STATE_UNDEFINED	MAV landed state is unknown
1	MAV_LANDED_STATE_ON_GROUND	MAV is landed (on ground)
2	MAV_LANDED_STATE_IN_AIR	MAV is in air
3	MAV_LANDED_STATE_TAKEOFF	MAV currently taking off
4	MAV_LANDED_STATE_LANDING	MAV currently landing

ADSB_ALTITUDE_TYPE

Enumeration of the ADSB altimeter types

CMD ID	Field Name	Description
0	ADSB_ALTITUDE_TYPE_PRESSURE_QNH	Altitude reported from a Baro source using QNH reference
1	ADSB_ALTITUDE_TYPE_GEOMETRIC	Altitude reported from a GNSS source

ADSB_EMITTER_TYPE

ADSB classification for the type of vehicle emitting the transponder signal

CMD ID	Field Name	Description
0	ADSB_EMITTER_TYPE_NO_INFO	
1	ADSB_EMITTER_TYPE_LIGHT	
2	ADSB_EMITTER_TYPE_SMALL	
3	ADSB_EMITTER_TYPE_LARGE	
4	ADSB_EMITTER_TYPE_HIGH_VORTEX_LARGE	
5	ADSB_EMITTER_TYPE_HEAVY	
6	ADSB_EMITTER_TYPE_HIGHLY_MANUV	
7	ADSB_EMITTER_TYPE_ROTOCRAFT	
8	ADSB_EMITTER_TYPE_UNASSIGNED	
9	ADSB_EMITTER_TYPE_GLIDER	
10	ADSB_EMITTER_TYPE_LIGHTER_AIR	
11	ADSB_EMITTER_TYPE_PARACHUTE	
12	ADSB_EMITTER_TYPE_ULTRA_LIGHT	
13	ADSB_EMITTER_TYPE_UNASSIGNED2	
14	ADSB_EMITTER_TYPE_UAV	
15	ADSB_EMITTER_TYPE_SPACE	
16	ADSB_EMITTER_TYPE_UNASSGINED3	
17	ADSB_EMITTER_TYPE_EMERGENCY_SURFACE	
18	ADSB_EMITTER_TYPE_SERVICE_SURFACE	
19	ADSB_EMITTER_TYPE_POINT_OBSTACLE	

ADSB_FLAGS

These flags indicate status such as data validity of each data source. Set = data valid

CMD ID	Field Name	Description
1	ADSB_FLAGS_VALID_COORDS	
2	ADSB_FLAGS_VALID_ALTITUDE	
4	ADSB_FLAGS_VALID_HEADING	
8	ADSB_FLAGS_VALID_VELOCITY	
16	ADSB_FLAGS_VALID_CALLSIGN	
32	ADSB_FLAGS_VALID_SQUAWK	
64	ADSB_FLAGS_SIMULATED	

MAV_DO_REPOSITION_FLAGS

Bitmask of options for the MAV_CMD_DO_REPOSITION

CMD ID	Field Name	Description
1	MAV_DO_REPOSITION_FLAGS_CHANGE_MODE	The aircraft should immediately transition into guided. This should not be set for follow me applications

ESTIMATOR_STATUS_FLAGS

Flags in EKF_STATUS message

CMD ID	Field Name	Description
1	ESTIMATOR_ATTITUDE	True if the attitude estimate is good
2	ESTIMATOR_VELOCITY_HORIZ	True if the horizontal velocity estimate is good
4	ESTIMATOR_VELOCITY_VERT	True if the vertical velocity estimate is good
8	ESTIMATOR_POS_HORIZ_REL	True if the horizontal position (relative) estimate is good
16	ESTIMATOR_POS_HORIZ_ABS	True if the horizontal position (absolute) estimate is good
32	ESTIMATOR_POS_VERT_ABS	True if the vertical position (absolute) estimate is good
64	ESTIMATOR_POS_VERT_AGL	True if the vertical position (above ground) estimate is good
128	ESTIMATOR_CONST_POS_MODE	True if the EKF is in a constant position mode and is not using external measurements (eg GPS or optical flow)
256	ESTIMATOR_PRED_POS_HORIZ_REL	True if the EKF has sufficient data to enter a mode that will provide a (relative) position estimate
512	ESTIMATOR_PRED_POS_HORIZ_ABS	True if the EKF has sufficient data to enter a mode that will provide a (absolute) position estimate
1024	ESTIMATOR_GPS_GLITCH	True if the EKF has detected a GPS glitch
2048	ESTIMATOR_ACCEL_ERROR	True if the EKF has detected bad accelerometer data

MOTOR_TEST_ORDER

CMD ID	Field Name	Description
0	MOTOR_TEST_ORDER_DEFAULT	default autopilot motor test method
1	MOTOR_TEST_ORDER_SEQUENCE	motor numbers are specified as their index in a predefined vehicle-specific sequence
2	MOTOR_TEST_ORDER_BOARD	motor numbers are specified as the output as labeled on the board

MOTOR_TEST_THROTTLE_TYPE

CMD ID	Field Name	Description
0	MOTOR_TEST_THROTTLE_PERCENT	throttle as a percentage from 0 ~ 100
1	MOTOR_TEST_THROTTLE_PWM	throttle as an absolute PWM value (normally in range of 1000~2000)
2	MOTOR_TEST_THROTTLE_PILOT	throttle pass-through from pilot's transmitter
3	MOTOR_TEST_COMPASS_CAL	per-motor compass calibration test

GPS_INPUT_IGNORE_FLAGS

CMD ID	Field Name	Description
1	GPS_INPUT_IGNORE_FLAG_ALT	ignore altitude field
2	GPS_INPUT_IGNORE_FLAG_HDOP	ignore hdop field
4	GPS_INPUT_IGNORE_FLAG_VDOP	ignore vdop field
8	GPS_INPUT_IGNORE_FLAG_VEL_HORIZ	ignore horizontal velocity field (vn and ve)
16	GPS_INPUT_IGNORE_FLAG_VEL_VERT	ignore vertical velocity field (vd)
32	GPS_INPUT_IGNORE_FLAG_SPEED_ACCURACY	ignore speed accuracy field
64	GPS_INPUT_IGNORE_FLAG_HORIZONTAL_ACCURACY	ignore horizontal accuracy field
128	GPS_INPUT_IGNORE_FLAG_VERTICAL_ACCURACY	ignore vertical accuracy field

MAV_COLLISION_ACTION

Possible actions an aircraft can take to avoid a collision.

CMD ID	Field Name	Description
0	MAV_COLLISION_ACTION_NONE	Ignore any potential collisions
1	MAV_COLLISION_ACTION_REPORT	Report potential collision
2	MAV_COLLISION_ACTION_ASCEND_OR_DESCEND	Ascend or Descend to avoid threat
3	MAV_COLLISION_ACTION_MOVE_HORIZONTALLY	Move horizontally to avoid threat
4	MAV_COLLISION_ACTION_MOVE_PERPENDICULAR	Aircraft to move perpendicular to the collision's velocity vector
5	MAV_COLLISION_ACTION_RTL	Aircraft to fly directly back to its launch point
6	MAV_COLLISION_ACTION_HOVER	Aircraft to stop in place

MAV_COLLISION_THREAT_LEVEL

Aircraft-rated danger from this threat.

CMD ID	Field Name	Description
0	MAV_COLLISION_THREAT_LEVEL_NONE	Not a threat
1	MAV_COLLISION_THREAT_LEVEL_LOW	Craft is mildly concerned about this threat
2	MAV_COLLISION_THREAT_LEVEL_HIGH	Craft is panicing, and may take actions to avoid threat

MAV_COLLISION_SRC

Source of information about this collision.

CMD ID	Field Name	Description
0	MAV_COLLISION_SRC_ADSB	ID field references ADSB_VEHICLE packets
1	MAV_COLLISION_SRC_MAVLINK_GPS_GLOBAL_INT	ID field references MAVLink SRC ID

GPS_FIX_TYPE

Type of GPS fix

CMD ID	Field Name	Description
0	GPS_FIX_TYPE_NO_GPS	No GPS connected
1	GPS_FIX_TYPE_NO_FIX	No position information, GPS is connected
2	GPS_FIX_TYPE_2D_FIX	2D position
3	GPS_FIX_TYPE_3D_FIX	3D position
4	GPS_FIX_TYPE_DGPS	DGPS/SBAS aided 3D position
5	GPS_FIX_TYPE_RTK_FLOAT	RTK float, 3D position
6	GPS_FIX_TYPE_RTK_FIXED	RTK Fixed, 3D position
7	GPS_FIX_TYPE_STATIC	Static fixed, typically used for base stations
8	GPS_FIX_TYPE_PPP	PPP, 3D position.

LANDING_TARGET_TYPE

Type of landing target

CMD ID	Field Name	Description
0	LANDING_TARGET_TYPE_LIGHT_BEACON	Landing target signaled by light beacon (ex: IR-LOCK)
1	LANDING_TARGET_TYPE_RADIO_BEACON	Landing target signaled by radio beacon (ex: ILS, NDB)
2	LANDING_TARGET_TYPE_VISION_FIDUCIAL	Landing target represented by a fiducial marker (ex: ARTag)
3	LANDING_TARGET_TYPE_VISION_OTHER	Landing target represented by a pre-defined visual shape/feature (ex: X-marker, H-marker, square)

VTOL_TRANSITION_HEADING

Direction of VTOL transition

CMD ID	Field Name	Description
0	VTOL_TRANSITION_HEADING_VEHICLE_DEFAULT	Respect the heading configuration of the vehicle.
1	VTOL_TRANSITION_HEADING_NEXT_WAYPOINT	Use the heading pointing towards the next waypoint.
2	VTOL_TRANSITION_HEADING_TAKEOFF	Use the heading on takeoff (while sitting on the ground).
3	VTOL_TRANSITION_HEADING_SPECIFIED	Use the specified heading in parameter 4.
4	VTOL_TRANSITION_HEADING_ANY	Use the current heading when reaching takeoff altitude (potentially facing the wind when weather-vaning is active).

CAMERA_CAP_FLAGS

Camera capability flags (Bitmap).

CMD ID	Field Name	Description
1	CAMERA_CAP_FLAGS_CAPTURE_VIDEO	Camera is able to record video.
2	CAMERA_CAP_FLAGS_CAPTURE_IMAGE	Camera is able to capture images.
4	CAMERA_CAP_FLAGS_HAS_MODES	Camera has separate Video and Image/Photo modes (MAV_CMD_SET_CAMERA_MOD
8	CAMERA_CAP_FLAGS_CAN_CAPTURE_IMAGE_IN_VIDEO_MODE	Camera can capture images while video mode
16	CAMERA_CAP_FLAGS_CAN_CAPTURE_VIDEO_IN_IMAGE_MODE	Camera can capture videos while i Photo/Image mode
32	CAMERA_CAP_FLAGS_HAS_IMAGE_SURVEY_MODE	Camera has image survey mode (MAV_CMD_SET_CAMERA_MOD

PARAM_ACK

Result from a PARAM_EXT_SET message.

CMD ID	Field Name	Description
0	PARAM_ACK_ACCEPTED	Parameter value ACCEPTED and SET
1	PARAM_ACK_VALUE_UNSUPPORTED	Parameter value UNKNOWN/UNSUPPORTED
2	PARAM_ACK_FAILED	Parameter failed to set
3	PARAM_ACK_IN_PROGRESS	Parameter value received but not yet validated or set. A subsequent PARAM_EXT_ACK will follow once operation is completed with the actual result. These are for parameters that may take longer to set. Instead of waiting for an ACK and potentially timing out, you will immediately receive this response to let you know it was received.

CAMERA_MODE

Camera Modes.

CMD ID	Field Name	Description
0	CAMERA_MODE_IMAGE	Camera is in image/photo capture mode.
1	CAMERA_MODE_VIDEO	Camera is in video capture mode.
2	CAMERA_MODE_IMAGE_SURVEY	Camera is in image survey capture mode. It allows for camera controller to do specific settings for surveys.

MAV_ARM_AUTH_DENIED_REASON

CMD ID	Field Name	Description
0	MAV_ARM_AUTH_DENIED_REASON_GENERIC	Not a specific reason
1	MAV_ARM_AUTH_DENIED_REASON_NONE	Authorizer will send the error as string to GCS
2	MAV_ARM_AUTH_DENIED_REASON_INVALID_WAYPOINT	At least one waypoint have a invalid value
3	MAV_ARM_AUTH_DENIED_REASON_TIMEOUT	Timeout in the authorizer process(in case it depends on network)
4	MAV_ARM_AUTH_DENIED_REASON_AIRSPACE_IN_USE	Airspace of the mission in use by another vehicle, second result parameter can have the waypoint id that caused it to be denied.
5	MAV_ARM_AUTH_DENIED_REASON_BAD_WEATHER	Weather is not good to fly

RTK_BASELINE_COORDINATE_SYSTEM

RTK GPS baseline coordinate system, used for RTK corrections

CMD ID	Field Name	Description
0	RTK_BASELINE_COORDINATE_SYSTEM_ECEF	Earth-centered, Earth-fixed
1	RTK_BASELINE_COORDINATE_SYSTEM_NED	North, East, Down

RC_TYPE

RC type

CMD ID	Field Name	Description
0	RC_TYPE_SPEKTRUM_DSM2	Spektrum DSM2
1	RC_TYPE_SPEKTRUM_DSMX	Spektrum DSMX

MAVLink Messages

HEARTBEAT (#0)

The heartbeat message shows that a system is present and responding. The type of the MAV and Autopilot hardware allow the receiving system to treat further messages from this system appropriate (e.g. by laying out the user interface based on the autopilot).

Field Name	Type	Description
type	uint8_t	Type of the MAV (quadrotor, helicopter, etc., up to 15 types, defined in MAV_TYPE ENUM) (Enum:MAV_TYPE)
autopilot	uint8_t	Autopilot type / class. defined in MAV_AUTOPILOT ENUM (Enum:MAV_AUTOPILOT)
base_mode	uint8_t	System mode bitfield, as defined by MAV_MODE_FLAG enum (Enum:MAV_MODE_FLAG)
custom_mode	uint32_t	A bitfield for use for autopilot-specific flags
system_status	uint8_t	System status flag, as defined by MAV_STATE enum (Enum:MAV_STATE)
mavlink_version	uint8_t_mavlink_version	MAVLink version, not writable by user, gets added by protocol because of magic data type: uint8_t_mavlink_version

SYS_STATUS (#1)

The general system state. If the system is following the MAVLink standard, the system state is mainly defined by three orthogonal states/modes: The system mode, which is either LOCKED (motors shut down and locked), MANUAL (system under RC control), GUIDED (system with autonomous position control, position setpoint controlled manually) or AUTO (system guided by path/waypoint planner). The NAV_MODE defined the current flight state: LIFTOFF (often an open-loop maneuver), LANDING, WAYPOINTS or VECTOR. This represents the internal navigation state machine. The system status shows whether the system is currently active or not and if an emergency occured. During the CRITICAL and EMERGENCY states the MAV is still considered to be active, but should start emergency procedures autonomously. After a failure occured it should first move from active to critical to allow manual intervention and then move to emergency after a certain timeout.

Field Name	Type	Description
onboard_control_sensors_present	uint32_t	Bitmask showing which onboard controllers and sensors are present. Value of 0: not present. Value of 1: present. Indices defined by ENUM MAV_SYS_STATUS_SENSOR (Enum:MAV_SYS_STATUS_SENSOR)
onboard_control_sensors_enabled	uint32_t	Bitmask showing which onboard controllers and sensors are enabled: Value of 0: not enabled. Value of 1: enabled. Indices defined by ENUM MAV_SYS_STATUS_SENSOR (Enum:MAV_SYS_STATUS_SENSOR)
onboard_control_sensors_health	uint32_t	Bitmask showing which onboard controllers and sensors are operational or have an error: Value of 0: not enabled. Value of 1: enabled. Indices defined by ENUM MAV_SYS_STATUS_SENSOR (Enum:MAV_SYS_STATUS_SENSOR)
load	uint16_t	Maximum usage in percent of the mainloop time, (0%: 0, 100%: 1000) should be always below 1000 (Units: d%)
voltage_battery	uint16_t	Battery voltage, in millivolts (1 = 1 millivolt) (Units: mV)
current_battery	int16_t	Battery current, in 10*milliamperes (1 = 10 milliampere), -1: autopilot does not measure the current (Units: cA)
battery_remaining	int8_t	Remaining battery energy: (0%: 0, 100%: 100), -1: autopilot estimate the remaining battery (Units: %)
drop_rate_comm	uint16_t	Communication drops in percent, (0%: 0, 100%: 10'000), (UART, I2C, SPI, CAN), dropped packets on all links (packets that were corrupted on reception on the MAV) (Units: c%)
errors_comm	uint16_t	Communication errors (UART, I2C, SPI, CAN), dropped packets on all links (packets that were corrupted on reception on the MAV)
errors_count1	uint16_t	Autopilot-specific errors
errors_count2	uint16_t	Autopilot-specific errors
errors_count3	uint16_t	Autopilot-specific errors
errors_count4	uint16_t	Autopilot-specific errors

SYSTEM_TIME (#2)

The system time is the time of the master clock, typically the computer clock of the main onboard computer.

Field Name	Type	Description
time_unix_usec	uint64_t	Timestamp of the master clock in microseconds since UNIX epoch. (Units: us)
time_boot_ms	uint32_t	Timestamp of the component clock since boot time in milliseconds. (Units: ms)

PING (#4)

A ping message either requesting or responding to a ping. This allows to measure the system latencies, including serial port, radio modem and UDP connections.

Field Name	Type	Description
time_usec	uint64_t	Unix timestamp in microseconds or since system boot if smaller than MAVLink epoch (1.1.2009) (Units: us)
seq	uint32_t	PING sequence
target_system	uint8_t	0: request ping from all receiving systems, if greater than 0: message is a ping response and number is the system id of the requesting system
target_component	uint8_t	0: request ping from all receiving components, if greater than 0: message is a ping response and number is the system id of the requesting system

CHANGE_OPERATOR_CONTROL (#5)

Request to control this MAV

Field Name	Type	Description
target_system	uint8_t	System the GCS requests control for
control_request	uint8_t	0: request control of this MAV, 1: Release control of this MAV
version	uint8_t	0: key as plaintext, 1-255: future, different hashing/encryption variants. The GCS should in general use the safest mode possible initially and then gradually move down the encryption level if it gets a NACK message indicating an encryption mismatch. (Units: rad)
passkey	char[25]	Password / Key, depending on version plaintext or encrypted. 25 or less characters, NULL terminated. The characters may involve A-Z, a-z, 0-9, and "!?,.-"

CHANGE_OPERATOR_CONTROL_ACK (#6)

Accept / deny control of this MAV

Field Name	Type	Description
gcs_system_id	uint8_t	ID of the GCS this message
control_request	uint8_t	0: request control of this MAV, 1: Release control of this MAV
ack	uint8_t	0: ACK, 1: NACK: Wrong passkey, 2: NACK: Unsupported passkey encryption method, 3: NACK: Already under control

AUTH_KEY (#7)

Emit an encrypted signature / key identifying this system. PLEASE NOTE: This protocol has been kept simple, so transmitting the key requires an encrypted channel for true safety.

Field Name	Type	Description
key	char[32]	key

SET_MODE (#11)

THIS INTERFACE IS DEPRECATED. USE COMMAND_LONG with MAV_CMD_DO_SET_MODE INSTEAD. Set the system mode, as defined by enum MAV_MODE. There is no target component id as the mode is by definition for the overall aircraft, not only for one component.

Field Name	Type	Description
target_system	uint8_t	The system setting the mode
base_mode	uint8_t	The new base mode (Enum:MAV_MODE)
custom_mode	uint32_t	The new autopilot-specific mode. This field can be ignored by an autopilot.

PARAM_REQUEST_READ (#20)

Request to read the onboard parameter with the param_id string id. Onboard parameters are stored as key[const char*] -> value[float]. This allows to send a parameter to any other component (such as the GCS) without the need of previous knowledge of possible parameter names. Thus the same GCS can store different parameters for different autopilots. See also https://mavlink.io/en/protocol/parameter.html for a full documentation of QGroundControl and IMU code.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
param_id	char[16]	Onboard parameter id, terminated by NULL if the length is less than 16 human-readable chars and WITHOUT null termination (NULL) byte if the length is exactly 16 chars - applications have to provide 16+1 bytes storage if the ID is stored as string
param_index	int16_t	Parameter index. Send -1 to use the param ID field as identifier (else the param id will be ignored)

PARAM_REQUEST_LIST (#21)

Request all parameters of this component. After this request, all parameters are emitted.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID

PARAM_VALUE (#22)

Emit the value of a onboard parameter. The inclusion of param_count and param_index in the message allows the recipient to keep track of received parameters and allows him to re-request missing parameters after a loss or timeout.

Field Name	Type	Description
param_id	char[16]	Onboard parameter id, terminated by NULL if the length is less than 16 human-readable chars and WITHOUT null termination (NULL) byte if the length is exactly 16 chars - applications have to provide 16+1 bytes storage if the ID is stored as string
param_value	float	Onboard parameter value
param_type	uint8_t	Onboard parameter type: see the MAV_PARAM_TYPE enum for supported data types. (Enum:MAV_PARAM_TYPE)
param_count	uint16_t	Total number of onboard parameters
param_index	uint16_t	Index of this onboard parameter

PARAM_SET (#23)

Set a parameter value TEMPORARILY to RAM. It will be reset to default on system reboot. Send the ACTION MAV_ACTION_STORAGE_WRITE to PERMANENTLY write the RAM contents to EEPROM. IMPORTANT: The receiving component should acknowledge the new parameter value by sending a param_value message to all communication partners. This will also ensure that multiple GCS all have an up-to-date list of all parameters. If the sending GCS did not receive a PARAM_VALUE message within its timeout time, it should re-send the PARAM_SET message.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
param_id	char[16]	Onboard parameter id, terminated by NULL if the length is less than 16 human-readable chars and WITHOUT null termination (NULL) byte if the length is exactly 16 chars - applications have to provide 16+1 bytes storage if the ID is stored as string
param_value	float	Onboard parameter value
param_type	uint8_t	Onboard parameter type: see the MAV_PARAM_TYPE enum for supported data types. (Enum:MAV_PARAM_TYPE)

GPS_RAW_INT (#24)

The global position, as returned by the Global Positioning System (GPS). This is NOT the global position estimate of the system, but rather a RAW sensor value. See message GLOBAL_POSITION for the global position estimate. Coordinate frame is right-handed, Z-axis up (GPS frame).

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds since UNIX epoch or microseconds since system boot) (Units: us)
fix_type	uint8_t	See the GPS_FIX_TYPE enum. (Enum:GPS_FIX_TYPE)
lat	int32_t	Latitude (WGS84, EGM96 ellipsoid), in degrees * 1E7 (Units: degE7)
lon	int32_t	Longitude (WGS84, EGM96 ellipsoid), in degrees * 1E7 (Units: degE7)
alt	int32_t	Altitude (AMSL, NOT WGS84), in meters * 1000 (positive for up). Note that virtually all GPS modules provide the AMSL altitude in addition to the WGS84 altitude. (Units: mm)
eph	uint16_t	GPS HDOP horizontal dilution of position (unitless). If unknown, set to: UINT16_MAX
epv	uint16_t	GPS VDOP vertical dilution of position (unitless). If unknown, set to: UINT16_MAX
vel	uint16_t	GPS ground speed (m/s * 100). If unknown, set to: UINT16_MAX (Units: cm/s)
cog	uint16_t	Course over ground (NOT heading, but direction of movement) in degrees * 100, 0.0..359.99 degrees. If unknown, set to: UINT16_MAX (Units: cdeg)
satellites_visible	uint8_t	Number of satellites visible. If unknown, set to 255
alt_ellipsoid **	int32_t	Altitude (above WGS84, EGM96 ellipsoid), in meters * 1000 (positive for up). (Units: mm)
h_acc **	uint32_t	Position uncertainty in meters * 1000 (positive for up). (Units: mm)
v_acc **	uint32_t	Altitude uncertainty in meters * 1000 (positive for up). (Units: mm)
vel_acc **	uint32_t	Speed uncertainty in meters * 1000 (positive for up). (Units: mm)
hdg_acc **	uint32_t	Heading / track uncertainty in degrees * 1e5. (Units: degE5)

GPS_STATUS (#25)

The positioning status, as reported by GPS. This message is intended to display status information about each satellite visible to the receiver. See message GLOBAL_POSITION for the global position estimate. This message can contain information for up to 20 satellites.

Field Name	Type	Description
satellites_visible	uint8_t	Number of satellites visible
satellite_prn	uint8_t[20]	Global satellite ID
satellite_used	uint8_t[20]	0: Satellite not used, 1: used for localization
satellite_elevation	uint8_t[20]	Elevation (0: right on top of receiver, 90: on the horizon) of satellite (Units: deg)
satellite_azimuth	uint8_t[20]	Direction of satellite, 0: 0 deg, 255: 360 deg. (Units: deg)
satellite_snr	uint8_t[20]	Signal to noise ratio of satellite (Units: dB)

SCALED_IMU (#26)

The RAW IMU readings for the usual 9DOF sensor setup. This message should contain the scaled values to the described units

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
xacc	int16_t	X acceleration (mg) (Units: mG)
yacc	int16_t	Y acceleration (mg) (Units: mG)
zacc	int16_t	Z acceleration (mg) (Units: mG)
xgyro	int16_t	Angular speed around X axis (millirad /sec) (Units: mrad/s)
ygyro	int16_t	Angular speed around Y axis (millirad /sec) (Units: mrad/s)
zgyro	int16_t	Angular speed around Z axis (millirad /sec) (Units: mrad/s)
xmag	int16_t	X Magnetic field (milli tesla) (Units: mT)
ymag	int16_t	Y Magnetic field (milli tesla) (Units: mT)
zmag	int16_t	Z Magnetic field (milli tesla) (Units: mT)

RAW_IMU (#27)

The RAW IMU readings for the usual 9DOF sensor setup. This message should always contain the true raw values without any scaling to allow data capture and system debugging.

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds since UNIX epoch or microseconds since system boot) (Units: us)
xacc	int16_t	X acceleration (raw)
yacc	int16_t	Y acceleration (raw)
zacc	int16_t	Z acceleration (raw)
xgyro	int16_t	Angular speed around X axis (raw)
ygyro	int16_t	Angular speed around Y axis (raw)
zgyro	int16_t	Angular speed around Z axis (raw)
xmag	int16_t	X Magnetic field (raw)
ymag	int16_t	Y Magnetic field (raw)
zmag	int16_t	Z Magnetic field (raw)

RAW_PRESSURE (#28)

The RAW pressure readings for the typical setup of one absolute pressure and one differential pressure sensor. The sensor values should be the raw, UNSCALED ADC values.

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds since UNIX epoch or microseconds since system boot) (Units: us)
press_abs	int16_t	Absolute pressure (raw)
press_diff1	int16_t	Differential pressure 1 (raw, 0 if nonexistant)
press_diff2	int16_t	Differential pressure 2 (raw, 0 if nonexistant)
temperature	int16_t	Raw Temperature measurement (raw)

SCALED_PRESSURE (#29)

The pressure readings for the typical setup of one absolute and differential pressure sensor. The units are as specified in each field.

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
press_abs	float	Absolute pressure (hectopascal) (Units: hPa)
press_diff	float	Differential pressure 1 (hectopascal) (Units: hPa)
temperature	int16_t	Temperature measurement (0.01 degrees celsius) (Units: cdegC)

ATTITUDE (#30)

The attitude in the aeronautical frame (right-handed, Z-down, X-front, Y-right).

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
roll	float	Roll angle (rad, -pi..+pi) (Units: rad)
pitch	float	Pitch angle (rad, -pi..+pi) (Units: rad)
yaw	float	Yaw angle (rad, -pi..+pi) (Units: rad)
rollspeed	float	Roll angular speed (rad/s) (Units: rad/s)
pitchspeed	float	Pitch angular speed (rad/s) (Units: rad/s)
yawspeed	float	Yaw angular speed (rad/s) (Units: rad/s)

ATTITUDE_QUATERNION (#31)

The attitude in the aeronautical frame (right-handed, Z-down, X-front, Y-right), expressed as quaternion. Quaternion order is w, x, y, z and a zero rotation would be expressed as (1 0 0 0).

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
q1	float	Quaternion component 1, w (1 in null-rotation)
q2	float	Quaternion component 2, x (0 in null-rotation)
q3	float	Quaternion component 3, y (0 in null-rotation)
q4	float	Quaternion component 4, z (0 in null-rotation)
rollspeed	float	Roll angular speed (rad/s) (Units: rad/s)
pitchspeed	float	Pitch angular speed (rad/s) (Units: rad/s)
yawspeed	float	Yaw angular speed (rad/s) (Units: rad/s)

LOCAL_POSITION_NED (#32)

The filtered local position (e.g. fused computer vision and accelerometers). Coordinate frame is right-handed, Z-axis down (aeronautical frame, NED / north-east-down convention)

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
x	float	X Position (Units: m)
y	float	Y Position (Units: m)
z	float	Z Position (Units: m)
vx	float	X Speed (Units: m/s)
vy	float	Y Speed (Units: m/s)
vz	float	Z Speed (Units: m/s)

GLOBAL_POSITION_INT (#33)

The filtered global position (e.g. fused GPS and accelerometers). The position is in GPS-frame (right-handed, Z-up). It is designed as scaled integer message since the resolution of float is not sufficient.

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
lat	int32_t	Latitude, expressed as degrees * 1E7 (Units: degE7)
lon	int32_t	Longitude, expressed as degrees * 1E7 (Units: degE7)
alt	int32_t	Altitude in meters, expressed as * 1000 (millimeters), AMSL (not WGS84 - note that virtually all GPS modules provide the AMSL as well) (Units: mm)
relative_alt	int32_t	Altitude above ground in meters, expressed as * 1000 (millimeters) (Units: mm)
vx	int16_t	Ground X Speed (Latitude, positive north), expressed as m/s * 100 (Units: cm/s)
vy	int16_t	Ground Y Speed (Longitude, positive east), expressed as m/s * 100 (Units: cm/s)
vz	int16_t	Ground Z Speed (Altitude, positive up), expressed as m/s * 100 (Units: cm/s)
hdg	uint16_t	Vehicle heading (yaw angle) in degrees * 100, 0.0..359.99 degrees. If unknown, set to: UINT16_MAX (Units: cdeg)

RC_CHANNELS_SCALED (#34)

The scaled values of the RC channels received. (-100%) -10000, (0%) 0, (100%) 10000. Channels that are inactive should be set to UINT16_MAX.

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
port	uint8_t	Servo output port (set of 8 outputs = 1 port). Most MAVs will just use one, but this allows for more than 8 servos.
chan1_scaled	int16_t	RC channel 1 value scaled, (-100%) -10000, (0%) 0, (100%) 10000, (invalid) INT16_MAX.
chan2_scaled	int16_t	RC channel 2 value scaled, (-100%) -10000, (0%) 0, (100%) 10000, (invalid) INT16_MAX.
chan3_scaled	int16_t	RC channel 3 value scaled, (-100%) -10000, (0%) 0, (100%) 10000, (invalid) INT16_MAX.
chan4_scaled	int16_t	RC channel 4 value scaled, (-100%) -10000, (0%) 0, (100%) 10000, (invalid) INT16_MAX.
chan5_scaled	int16_t	RC channel 5 value scaled, (-100%) -10000, (0%) 0, (100%) 10000, (invalid) INT16_MAX.
chan6_scaled	int16_t	RC channel 6 value scaled, (-100%) -10000, (0%) 0, (100%) 10000, (invalid) INT16_MAX.
chan7_scaled	int16_t	RC channel 7 value scaled, (-100%) -10000, (0%) 0, (100%) 10000, (invalid) INT16_MAX.
chan8_scaled	int16_t	RC channel 8 value scaled, (-100%) -10000, (0%) 0, (100%) 10000, (invalid) INT16_MAX.
rssi	uint8_t	Receive signal strength indicator, 0: 0%, 100: 100%, 255: invalid/unknown. (Units: %)

RC_CHANNELS_RAW (#35)

The RAW values of the RC channels received. The standard PPM modulation is as follows: 1000 microseconds: 0%, 2000 microseconds: 100%. Individual receivers/transmitters might violate this specification.

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
port	uint8_t	Servo output port (set of 8 outputs = 1 port). Most MAVs will just use one, but this allows for more than 8 servos.
chan1_raw	uint16_t	RC channel 1 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan2_raw	uint16_t	RC channel 2 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan3_raw	uint16_t	RC channel 3 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan4_raw	uint16_t	RC channel 4 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan5_raw	uint16_t	RC channel 5 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan6_raw	uint16_t	RC channel 6 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan7_raw	uint16_t	RC channel 7 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan8_raw	uint16_t	RC channel 8 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
rssi	uint8_t	Receive signal strength indicator, 0: 0%, 100: 100%, 255: invalid/unknown. (Units: %)

SERVO_OUTPUT_RAW (#36)

The RAW values of the servo outputs (for RC input from the remote, use the RC_CHANNELS messages). The standard PPM modulation is as follows: 1000 microseconds: 0%, 2000 microseconds: 100%.

Field Name	Type	Description
time_usec	uint32_t	Timestamp (microseconds since system boot) (Units: us)
port	uint8_t	Servo output port (set of 8 outputs = 1 port). Most MAVs will just use one, but this allows to encode more than 8 servos.
servo1_raw	uint16_t	Servo output 1 value, in microseconds (Units: us)
servo2_raw	uint16_t	Servo output 2 value, in microseconds (Units: us)
servo3_raw	uint16_t	Servo output 3 value, in microseconds (Units: us)
servo4_raw	uint16_t	Servo output 4 value, in microseconds (Units: us)
servo5_raw	uint16_t	Servo output 5 value, in microseconds (Units: us)
servo6_raw	uint16_t	Servo output 6 value, in microseconds (Units: us)
servo7_raw	uint16_t	Servo output 7 value, in microseconds (Units: us)
servo8_raw	uint16_t	Servo output 8 value, in microseconds (Units: us)
servo9_raw **	uint16_t	Servo output 9 value, in microseconds (Units: us)
servo10_raw **	uint16_t	Servo output 10 value, in microseconds (Units: us)
servo11_raw **	uint16_t	Servo output 11 value, in microseconds (Units: us)
servo12_raw **	uint16_t	Servo output 12 value, in microseconds (Units: us)
servo13_raw **	uint16_t	Servo output 13 value, in microseconds (Units: us)
servo14_raw **	uint16_t	Servo output 14 value, in microseconds (Units: us)
servo15_raw **	uint16_t	Servo output 15 value, in microseconds (Units: us)
servo16_raw **	uint16_t	Servo output 16 value, in microseconds (Units: us)

MISSION_REQUEST_PARTIAL_LIST (#37)

Request a partial list of mission items from the system/component. https://mavlink.io/en/protocol/mission.html. If start and end index are the same, just send one waypoint.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
start_index	int16_t	Start index, 0 by default
end_index	int16_t	End index, -1 by default (-1: send list to end). Else a valid index of the list
mission_type **	uint8_t	Mission type, see MAV_MISSION_TYPE (Enum:MAV_MISSION_TYPE)

MISSION_WRITE_PARTIAL_LIST (#38)

This message is sent to the MAV to write a partial list. If start index == end index, only one item will be transmitted / updated. If the start index is NOT 0 and above the current list size, this request should be REJECTED!

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
start_index	int16_t	Start index, 0 by default and smaller / equal to the largest index of the current onboard list.
end_index	int16_t	End index, equal or greater than start index.
mission_type **	uint8_t	Mission type, see MAV_MISSION_TYPE (Enum:MAV_MISSION_TYPE)

MISSION_ITEM (#39)

Message encoding a mission item. This message is emitted to announce the presence of a mission item and to set a mission item on the system. The mission item can be either in x, y, z meters (type: LOCAL) or x:lat, y:lon, z:altitude. Local frame is Z-down, right handed (NED), global frame is Z-up, right handed (ENU). See also https://mavlink.io/en/protocol/mission.html.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
seq	uint16_t	Sequence
frame	uint8_t	The coordinate system of the waypoint, as defined by MAV_FRAME enum (Enum:MAV_FRAME)
command	uint16_t	The scheduled action for the waypoint, as defined by MAV_CMD enum (Enum:MAV_CMD)
current	uint8_t	false:0, true:1
autocontinue	uint8_t	autocontinue to next wp
param1	float	PARAM1, see MAV_CMD enum
param2	float	PARAM2, see MAV_CMD enum
param3	float	PARAM3, see MAV_CMD enum
param4	float	PARAM4, see MAV_CMD enum
x	float	PARAM5 / local: x position, global: latitude
y	float	PARAM6 / y position: global: longitude
z	float	PARAM7 / z position: global: altitude (relative or absolute, depending on frame.
mission_type **	uint8_t	Mission type, see MAV_MISSION_TYPE (Enum:MAV_MISSION_TYPE)

MISSION_REQUEST (#40)

Request the information of the mission item with the sequence number seq. The response of the system to this message should be a MISSION_ITEM message. https://mavlink.io/en/protocol/mission.html

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
seq	uint16_t	Sequence
mission_type **	uint8_t	Mission type, see MAV_MISSION_TYPE (Enum:MAV_MISSION_TYPE)

MISSION_SET_CURRENT (#41)

Set the mission item with sequence number seq as current item. This means that the MAV will continue to this mission item on the shortest path (not following the mission items in-between).

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
seq	uint16_t	Sequence

MISSION_CURRENT (#42)

Message that announces the sequence number of the current active mission item. The MAV will fly towards this mission item.

Field Name	Type	Description
seq	uint16_t	Sequence

MISSION_REQUEST_LIST (#43)

Request the overall list of mission items from the system/component.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
mission_type **	uint8_t	Mission type, see MAV_MISSION_TYPE (Enum:MAV_MISSION_TYPE)

MISSION_COUNT (#44)

This message is emitted as response to MISSION_REQUEST_LIST by the MAV and to initiate a write transaction. The GCS can then request the individual mission item based on the knowledge of the total number of waypoints.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
count	uint16_t	Number of mission items in the sequence
mission_type **	uint8_t	Mission type, see MAV_MISSION_TYPE (Enum:MAV_MISSION_TYPE)

MISSION_CLEAR_ALL (#45)

Delete all mission items at once.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
mission_type **	uint8_t	Mission type, see MAV_MISSION_TYPE (Enum:MAV_MISSION_TYPE)

MISSION_ITEM_REACHED (#46)

A certain mission item has been reached. The system will either hold this position (or circle on the orbit) or (if the autocontinue on the WP was set) continue to the next waypoint.

Field Name	Type	Description
seq	uint16_t	Sequence

MISSION_ACK (#47)

Ack message during waypoint handling. The type field states if this message is a positive ack (type=0) or if an error happened (type=non-zero).

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
type	uint8_t	See MAV_MISSION_RESULT enum (Enum:MAV_MISSION_RESULT)
mission_type **	uint8_t	Mission type, see MAV_MISSION_TYPE (Enum:MAV_MISSION_TYPE)

SET_GPS_GLOBAL_ORIGIN (#48)

As local waypoints exist, the global waypoint reference allows to transform between the local coordinate frame and the global (GPS) coordinate frame. This can be necessary when e.g. in- and outdoor settings are connected and the MAV should move from in- to outdoor.

Field Name	Type	Description
target_system	uint8_t	System ID
latitude	int32_t	Latitude (WGS84), in degrees * 1E7 (Units: degE7)
longitude	int32_t	Longitude (WGS84), in degrees * 1E7 (Units: degE7)
altitude	int32_t	Altitude (AMSL), in meters * 1000 (positive for up) (Units: mm)
time_usec **	uint64_t	Timestamp (microseconds since UNIX epoch or microseconds since system boot) (Units: us)

GPS_GLOBAL_ORIGIN (#49)

Once the MAV sets a new GPS-Local correspondence, this message announces the origin (0,0,0) position

Field Name	Type	Description
latitude	int32_t	Latitude (WGS84), in degrees * 1E7 (Units: degE7)
longitude	int32_t	Longitude (WGS84), in degrees * 1E7 (Units: degE7)
altitude	int32_t	Altitude (AMSL), in meters * 1000 (positive for up) (Units: mm)
time_usec **	uint64_t	Timestamp (microseconds since UNIX epoch or microseconds since system boot) (Units: us)

PARAM_MAP_RC (#50)

Bind a RC channel to a parameter. The parameter should change accoding to the RC channel value.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
param_id	char[16]	Onboard parameter id, terminated by NULL if the length is less than 16 human-readable chars and WITHOUT null termination (NULL) byte if the length is exactly 16 chars - applications have to provide 16+1 bytes storage if the ID is stored as string
param_index	int16_t	Parameter index. Send -1 to use the param ID field as identifier (else the param id will be ignored), send -2 to disable any existing map for this rc_channel_index.
parameter_rc_channel_index	uint8_t	Index of parameter RC channel. Not equal to the RC channel id. Typically correpsonds to a potentiometer-knob on the RC.
param_value0	float	Initial parameter value
scale	float	Scale, maps the RC range [-1, 1] to a parameter value
param_value_min	float	Minimum param value. The protocol does not define if this overwrites an onboard minimum value. (Depends on implementation)
param_value_max	float	Maximum param value. The protocol does not define if this overwrites an onboard maximum value. (Depends on implementation)

MISSION_REQUEST_INT (#51)

Request the information of the mission item with the sequence number seq. The response of the system to this message should be a MISSION_ITEM_INT message. https://mavlink.io/en/protocol/mission.html

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
seq	uint16_t	Sequence
mission_type **	uint8_t	Mission type, see MAV_MISSION_TYPE (Enum:MAV_MISSION_TYPE)

SAFETY_SET_ALLOWED_AREA (#54)

Set a safety zone (volume), which is defined by two corners of a cube. This message can be used to tell the MAV which setpoints/waypoints to accept and which to reject. Safety areas are often enforced by national or competition regulations.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
frame	uint8_t	Coordinate frame, as defined by MAV_FRAME enum. Can be either global, GPS, right-handed with Z axis up or local, right handed, Z axis down. (Enum:MAV_FRAME)
p1x	float	x position 1 / Latitude 1 (Units: m)
p1y	float	y position 1 / Longitude 1 (Units: m)
p1z	float	z position 1 / Altitude 1 (Units: m)
p2x	float	x position 2 / Latitude 2 (Units: m)
p2y	float	y position 2 / Longitude 2 (Units: m)
p2z	float	z position 2 / Altitude 2 (Units: m)

SAFETY_ALLOWED_AREA (#55)

Read out the safety zone the MAV currently assumes.

Field Name	Type	Description
frame	uint8_t	Coordinate frame, as defined by MAV_FRAME enum. Can be either global, GPS, right-handed with Z axis up or local, right handed, Z axis down. (Enum:MAV_FRAME)
p1x	float	x position 1 / Latitude 1 (Units: m)
p1y	float	y position 1 / Longitude 1 (Units: m)
p1z	float	z position 1 / Altitude 1 (Units: m)
p2x	float	x position 2 / Latitude 2 (Units: m)
p2y	float	y position 2 / Longitude 2 (Units: m)
p2z	float	z position 2 / Altitude 2 (Units: m)

ATTITUDE_QUATERNION_COV (#61)

The attitude in the aeronautical frame (right-handed, Z-down, X-front, Y-right), expressed as quaternion. Quaternion order is w, x, y, z and a zero rotation would be expressed as (1 0 0 0).

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds since system boot or since UNIX epoch) (Units: us)
q	float[4]	Quaternion components, w, x, y, z (1 0 0 0 is the null-rotation)
rollspeed	float	Roll angular speed (rad/s) (Units: rad/s)
pitchspeed	float	Pitch angular speed (rad/s) (Units: rad/s)
yawspeed	float	Yaw angular speed (rad/s) (Units: rad/s)
covariance	float[9]	Attitude covariance

NAV_CONTROLLER_OUTPUT (#62)

The state of the fixed wing navigation and position controller.

Field Name	Type	Description
nav_roll	float	Current desired roll in degrees (Units: deg)
nav_pitch	float	Current desired pitch in degrees (Units: deg)
nav_bearing	int16_t	Current desired heading in degrees (Units: deg)
target_bearing	int16_t	Bearing to current waypoint/target in degrees (Units: deg)
wp_dist	uint16_t	Distance to active waypoint in meters (Units: m)
alt_error	float	Current altitude error in meters (Units: m)
aspd_error	float	Current airspeed error in meters/second (Units: m/s)
xtrack_error	float	Current crosstrack error on x-y plane in meters (Units: m)

GLOBAL_POSITION_INT_COV (#63)

The filtered global position (e.g. fused GPS and accelerometers). The position is in GPS-frame (right-handed, Z-up). It is designed as scaled integer message since the resolution of float is not sufficient. NOTE: This message is intended for onboard networks / companion computers and higher-bandwidth links and optimized for accuracy and completeness. Please use the GLOBAL_POSITION_INT message for a minimal subset.

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds since system boot or since UNIX epoch) (Units: us)
estimator_type	uint8_t	Class id of the estimator this estimate originated from. (Enum:MAV_ESTIMATOR_TYPE)
lat	int32_t	Latitude, expressed as degrees * 1E7 (Units: degE7)
lon	int32_t	Longitude, expressed as degrees * 1E7 (Units: degE7)
alt	int32_t	Altitude in meters, expressed as * 1000 (millimeters), above MSL (Units: mm)
relative_alt	int32_t	Altitude above ground in meters, expressed as * 1000 (millimeters) (Units: mm)
vx	float	Ground X Speed (Latitude), expressed as m/s (Units: m/s)
vy	float	Ground Y Speed (Longitude), expressed as m/s (Units: m/s)
vz	float	Ground Z Speed (Altitude), expressed as m/s (Units: m/s)
covariance	float[36]	Covariance matrix (first six entries are the first ROW, next six entries are the second row, etc.)

LOCAL_POSITION_NED_COV (#64)

The filtered local position (e.g. fused computer vision and accelerometers). Coordinate frame is right-handed, Z-axis down (aeronautical frame, NED / north-east-down convention)

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds since system boot or since UNIX epoch) (Units: us)
estimator_type	uint8_t	Class id of the estimator this estimate originated from. (Enum:MAV_ESTIMATOR_TYPE)
x	float	X Position (Units: m)
y	float	Y Position (Units: m)
z	float	Z Position (Units: m)
vx	float	X Speed (m/s) (Units: m/s)
vy	float	Y Speed (m/s) (Units: m/s)
vz	float	Z Speed (m/s) (Units: m/s)
ax	float	X Acceleration (m/s^2) (Units: m/s/s)
ay	float	Y Acceleration (m/s^2) (Units: m/s/s)
az	float	Z Acceleration (m/s^2) (Units: m/s/s)
covariance	float[45]	Covariance matrix upper right triangular (first nine entries are the first ROW, next eight entries are the second row, etc.)

RC_CHANNELS (#65)

The PPM values of the RC channels received. The standard PPM modulation is as follows: 1000 microseconds: 0%, 2000 microseconds: 100%. Individual receivers/transmitters might violate this specification.

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
chancount	uint8_t	Total number of RC channels being received. This can be larger than 18, indicating that more channels are available but not given in this message. This value should be 0 when no RC channels are available.
chan1_raw	uint16_t	RC channel 1 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan2_raw	uint16_t	RC channel 2 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan3_raw	uint16_t	RC channel 3 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan4_raw	uint16_t	RC channel 4 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan5_raw	uint16_t	RC channel 5 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan6_raw	uint16_t	RC channel 6 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan7_raw	uint16_t	RC channel 7 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan8_raw	uint16_t	RC channel 8 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan9_raw	uint16_t	RC channel 9 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan10_raw	uint16_t	RC channel 10 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan11_raw	uint16_t	RC channel 11 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan12_raw	uint16_t	RC channel 12 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan13_raw	uint16_t	RC channel 13 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan14_raw	uint16_t	RC channel 14 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan15_raw	uint16_t	RC channel 15 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan16_raw	uint16_t	RC channel 16 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan17_raw	uint16_t	RC channel 17 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
chan18_raw	uint16_t	RC channel 18 value, in microseconds. A value of UINT16_MAX implies the channel is unused. (Units: us)
rssi	uint8_t	Receive signal strength indicator, 0: 0%, 100: 100%, 255: invalid/unknown. (Units: %)

REQUEST_DATA_STREAM (#66)

THIS INTERFACE IS DEPRECATED. USE SET_MESSAGE_INTERVAL INSTEAD.

Field Name	Type	Description
target_system	uint8_t	The target requested to send the message stream.
target_component	uint8_t	The target requested to send the message stream.
req_stream_id	uint8_t	The ID of the requested data stream
req_message_rate	uint16_t	The requested message rate (Units: Hz)
start_stop	uint8_t	1 to start sending, 0 to stop sending.

DATA_STREAM (#67)

THIS INTERFACE IS DEPRECATED. USE MESSAGE_INTERVAL INSTEAD.

Field Name	Type	Description
stream_id	uint8_t	The ID of the requested data stream
message_rate	uint16_t	The message rate (Units: Hz)
on_off	uint8_t	1 stream is enabled, 0 stream is stopped.

MANUAL_CONTROL (#69)

This message provides an API for manually controlling the vehicle using standard joystick axes nomenclature, along with a joystick-like input device. Unused axes can be disabled an buttons are also transmit as boolean values of their

Field Name	Type	Description
target	uint8_t	The system to be controlled.
x	int16_t	X-axis, normalized to the range [-1000,1000]. A value of INT16_MAX indicates that this axis is invalid. Generally corresponds to forward(1000)-backward(-1000) movement on a joystick and the pitch of a vehicle.
y	int16_t	Y-axis, normalized to the range [-1000,1000]. A value of INT16_MAX indicates that this axis is invalid. Generally corresponds to left(-1000)-right(1000) movement on a joystick and the roll of a vehicle.
z	int16_t	Z-axis, normalized to the range [-1000,1000]. A value of INT16_MAX indicates that this axis is invalid. Generally corresponds to a separate slider movement with maximum being 1000 and minimum being -1000 on a joystick and the thrust of a vehicle. Positive values are positive thrust, negative values are negative thrust.
r	int16_t	R-axis, normalized to the range [-1000,1000]. A value of INT16_MAX indicates that this axis is invalid. Generally corresponds to a twisting of the joystick, with counter-clockwise being 1000 and clockwise being -1000, and the yaw of a vehicle.
buttons	uint16_t	A bitfield corresponding to the joystick buttons' current state, 1 for pressed, 0 for released. The lowest bit corresponds to Button 1.

RC_CHANNELS_OVERRIDE (#70)

The RAW values of the RC channels sent to the MAV to override info received from the RC radio. A value of UINT16_MAX means no change to that channel. A value of 0 means control of that channel should be released back to the RC radio. The standard PPM modulation is as follows: 1000 microseconds: 0%, 2000 microseconds: 100%. Individual receivers/transmitters might violate this specification.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
chan1_raw	uint16_t	RC channel 1 value, in microseconds. A value of UINT16_MAX means to ignore this field. (Units: us)
chan2_raw	uint16_t	RC channel 2 value, in microseconds. A value of UINT16_MAX means to ignore this field. (Units: us)
chan3_raw	uint16_t	RC channel 3 value, in microseconds. A value of UINT16_MAX means to ignore this field. (Units: us)
chan4_raw	uint16_t	RC channel 4 value, in microseconds. A value of UINT16_MAX means to ignore this field. (Units: us)
chan5_raw	uint16_t	RC channel 5 value, in microseconds. A value of UINT16_MAX means to ignore this field. (Units: us)
chan6_raw	uint16_t	RC channel 6 value, in microseconds. A value of UINT16_MAX means to ignore this field. (Units: us)
chan7_raw	uint16_t	RC channel 7 value, in microseconds. A value of UINT16_MAX means to ignore this field. (Units: us)
chan8_raw	uint16_t	RC channel 8 value, in microseconds. A value of UINT16_MAX means to ignore this field. (Units: us)
chan9_raw **	uint16_t	RC channel 9 value, in microseconds. A value of 0 means to ignore this field. (Units: us)
chan10_raw **	uint16_t	RC channel 10 value, in microseconds. A value of 0 or UINT16_MAX means to ignore this field. (Units: us)
chan11_raw **	uint16_t	RC channel 11 value, in microseconds. A value of 0 or UINT16_MAX means to ignore this field. (Units: us)
chan12_raw **	uint16_t	RC channel 12 value, in microseconds. A value of 0 or UINT16_MAX means to ignore this field. (Units: us)
chan13_raw **	uint16_t	RC channel 13 value, in microseconds. A value of 0 or UINT16_MAX means to ignore this field. (Units: us)
chan14_raw **	uint16_t	RC channel 14 value, in microseconds. A value of 0 or UINT16_MAX means to ignore this field. (Units: us)
chan15_raw **	uint16_t	RC channel 15 value, in microseconds. A value of 0 or UINT16_MAX means to ignore this field. (Units: us)
chan16_raw **	uint16_t	RC channel 16 value, in microseconds. A value of 0 or UINT16_MAX means to ignore this field. (Units: us)
chan17_raw **	uint16_t	RC channel 17 value, in microseconds. A value of 0 or UINT16_MAX means to ignore this field. (Units: us)
chan18_raw **	uint16_t	RC channel 18 value, in microseconds. A value of 0 or UINT16_MAX means to ignore this field. (Units: us)

MISSION_ITEM_INT (#73)

Message encoding a mission item. This message is emitted to announce the presence of a mission item and to set a mission item on the system. The mission item can be either in x, y, z meters (type: LOCAL) or x:lat, y:lon, z:altitude. Local frame is Z-down, right handed (NED), global frame is Z-up, right handed (ENU). See also https://mavlink.io/en/protocol/mission.html.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
seq	uint16_t	Waypoint ID (sequence number). Starts at zero. Increases monotonically for each waypoint, no gaps in the sequence (0,1,2,3,4).
frame	uint8_t	The coordinate system of the waypoint, as defined by MAV_FRAME enum (Enum:MAV_FRAME)
command	uint16_t	The scheduled action for the waypoint, as defined by MAV_CMD enum (Enum:MAV_CMD)
current	uint8_t	false:0, true:1
autocontinue	uint8_t	autocontinue to next wp
param1	float	PARAM1, see MAV_CMD enum
param2	float	PARAM2, see MAV_CMD enum
param3	float	PARAM3, see MAV_CMD enum
param4	float	PARAM4, see MAV_CMD enum
x	int32_t	PARAM5 / local: x position in meters * 1e4, global: latitude in degrees * 10^7
y	int32_t	PARAM6 / y position: local: x position in meters * 1e4, global: longitude in degrees *10^7
z	float	PARAM7 / z position: global: altitude in meters (relative or absolute, depending on frame.
mission_type **	uint8_t	Mission type, see MAV_MISSION_TYPE (Enum:MAV_MISSION_TYPE)

VFR_HUD (#74)

Metrics typically displayed on a HUD for fixed wing aircraft

Field Name	Type	Description
airspeed	float	Current airspeed in m/s (Units: m/s)
groundspeed	float	Current ground speed in m/s (Units: m/s)
heading	int16_t	Current heading in degrees, in compass units (0..360, 0=north) (Units: deg)
throttle	uint16_t	Current throttle setting in integer percent, 0 to 100 (Units: %)
alt	float	Current altitude (MSL), in meters (Units: m)
climb	float	Current climb rate in meters/second (Units: m/s)

COMMAND_INT (#75)

Message encoding a command with parameters as scaled integers. Scaling depends on the actual command value.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
frame	uint8_t	The coordinate system of the COMMAND, as defined by MAV_FRAME enum (Enum:MAV_FRAME)
command	uint16_t	The scheduled action for the mission item, as defined by MAV_CMD enum (Enum:MAV_CMD)
current	uint8_t	false:0, true:1
autocontinue	uint8_t	autocontinue to next wp
param1	float	PARAM1, see MAV_CMD enum
param2	float	PARAM2, see MAV_CMD enum
param3	float	PARAM3, see MAV_CMD enum
param4	float	PARAM4, see MAV_CMD enum
x	int32_t	PARAM5 / local: x position in meters * 1e4, global: latitude in degrees * 10^7
y	int32_t	PARAM6 / local: y position in meters * 1e4, global: longitude in degrees * 10^7
z	float	PARAM7 / z position: global: altitude in meters (relative or absolute, depending on frame.

COMMAND_LONG (#76)

Send a command with up to seven parameters to the MAV

Field Name	Type	Description
target_system	uint8_t	System which should execute the command
target_component	uint8_t	Component which should execute the command, 0 for all components
command	uint16_t	Command ID, as defined by MAV_CMD enum. (Enum:MAV_CMD)
confirmation	uint8_t	0: First transmission of this command. 1-255: Confirmation transmissions (e.g. for kill command)
param1	float	Parameter 1, as defined by MAV_CMD enum.
param2	float	Parameter 2, as defined by MAV_CMD enum.
param3	float	Parameter 3, as defined by MAV_CMD enum.
param4	float	Parameter 4, as defined by MAV_CMD enum.
param5	float	Parameter 5, as defined by MAV_CMD enum.
param6	float	Parameter 6, as defined by MAV_CMD enum.
param7	float	Parameter 7, as defined by MAV_CMD enum.

COMMAND_ACK (#77)

Report status of a command. Includes feedback whether the command was executed.

Field Name	Type	Description
command	uint16_t	Command ID, as defined by MAV_CMD enum. (Enum:MAV_CMD)
result	uint8_t	See MAV_RESULT enum (Enum:MAV_RESULT)
progress **	uint8_t	WIP: Also used as result_param1, it can be set with a enum containing the errors reasons of why the command was denied or the progress percentage or 255 if unknown the progress when result is MAV_RESULT_IN_PROGRESS.
result_param2 **	int32_t	WIP: Additional parameter of the result, example: which parameter of MAV_CMD_NAV_WAYPOINT caused it to be denied.
target_system **	uint8_t	WIP: System which requested the command to be executed
target_component **	uint8_t	WIP: Component which requested the command to be executed

MANUAL_SETPOINT (#81)

Setpoint in roll, pitch, yaw and thrust from the operator

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp in milliseconds since system boot (Units: ms)
roll	float	Desired roll rate in radians per second (Units: rad/s)
pitch	float	Desired pitch rate in radians per second (Units: rad/s)
yaw	float	Desired yaw rate in radians per second (Units: rad/s)
thrust	float	Collective thrust, normalized to 0 .. 1
mode_switch	uint8_t	Flight mode switch position, 0.. 255
manual_override_switch	uint8_t	Override mode switch position, 0.. 255

SET_ATTITUDE_TARGET (#82)

Sets a desired vehicle attitude. Used by an external controller to command the vehicle (manual controller or other system).

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp in milliseconds since system boot (Units: ms)
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
type_mask	uint8_t	Mappings: If any of these bits are set, the corresponding input should be ignored: bit 1: body roll rate, bit 2: body pitch rate, bit 3: body yaw rate. bit 4-bit 6: reserved, bit 7: throttle, bit 8: attitude
q	float[4]	Attitude quaternion (w, x, y, z order, zero-rotation is 1, 0, 0, 0)
body_roll_rate	float	Body roll rate in radians per second (Units: rad/s)
body_pitch_rate	float	Body pitch rate in radians per second (Units: rad/s)
body_yaw_rate	float	Body yaw rate in radians per second (Units: rad/s)
thrust	float	Collective thrust, normalized to 0 .. 1 (-1 .. 1 for vehicles capable of reverse trust)

ATTITUDE_TARGET (#83)

Reports the current commanded attitude of the vehicle as specified by the autopilot. This should match the commands sent in a SET_ATTITUDE_TARGET message if the vehicle is being controlled this way.

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp in milliseconds since system boot (Units: ms)
type_mask	uint8_t	Mappings: If any of these bits are set, the corresponding input should be ignored: bit 1: body roll rate, bit 2: body pitch rate, bit 3: body yaw rate. bit 4-bit 7: reserved, bit 8: attitude
q	float[4]	Attitude quaternion (w, x, y, z order, zero-rotation is 1, 0, 0, 0)
body_roll_rate	float	Body roll rate in radians per second (Units: rad/s)
body_pitch_rate	float	Body pitch rate in radians per second (Units: rad/s)
body_yaw_rate	float	Body yaw rate in radians per second (Units: rad/s)
thrust	float	Collective thrust, normalized to 0 .. 1 (-1 .. 1 for vehicles capable of reverse trust)

SET_POSITION_TARGET_LOCAL_NED (#84)

Sets a desired vehicle position in a local north-east-down coordinate frame. Used by an external controller to command the vehicle (manual controller or other system).

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp in milliseconds since system boot (Units: ms)
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
coordinate_frame	uint8_t	Valid options are: MAV_FRAME_LOCAL_NED = 1, MAV_FRAME_LOCAL_OFFSET_NED = 7, MAV_FRAME_BODY_NED = 8, MAV_FRAME_BODY_OFFSET_NED = 9 (Enum:MAV_FRAME)
type_mask	uint16_t	Bitmask to indicate which dimensions should be ignored by the vehicle: a value of 0b0000000000000000 or 0b0000001000000000 indicates that none of the setpoint dimensions should be ignored. If bit 10 is set the floats afx afy afz should be interpreted as force instead of acceleration. Mapping: bit 1: x, bit 2: y, bit 3: z, bit 4: vx, bit 5: vy, bit 6: vz, bit 7: ax, bit 8: ay, bit 9: az, bit 10: is force setpoint, bit 11: yaw, bit 12: yaw rate
x	float	X Position in NED frame in meters (Units: m)
y	float	Y Position in NED frame in meters (Units: m)
z	float	Z Position in NED frame in meters (note, altitude is negative in NED) (Units: m)
vx	float	X velocity in NED frame in meter / s (Units: m/s)
vy	float	Y velocity in NED frame in meter / s (Units: m/s)
vz	float	Z velocity in NED frame in meter / s (Units: m/s)
afx	float	X acceleration or force (if bit 10 of type_mask is set) in NED frame in meter / s^2 or N (Units: m/s/s)
afy	float	Y acceleration or force (if bit 10 of type_mask is set) in NED frame in meter / s^2 or N (Units: m/s/s)
afz	float	Z acceleration or force (if bit 10 of type_mask is set) in NED frame in meter / s^2 or N (Units: m/s/s)
yaw	float	yaw setpoint in rad (Units: rad)
yaw_rate	float	yaw rate setpoint in rad/s (Units: rad/s)

POSITION_TARGET_LOCAL_NED (#85)

Reports the current commanded vehicle position, velocity, and acceleration as specified by the autopilot. This should match the commands sent in SET_POSITION_TARGET_LOCAL_NED if the vehicle is being controlled this way.

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp in milliseconds since system boot (Units: ms)
coordinate_frame	uint8_t	Valid options are: MAV_FRAME_LOCAL_NED = 1, MAV_FRAME_LOCAL_OFFSET_NED = 7, MAV_FRAME_BODY_NED = 8, MAV_FRAME_BODY_OFFSET_NED = 9 (Enum:MAV_FRAME)
type_mask	uint16_t	Bitmask to indicate which dimensions should be ignored by the vehicle: a value of 0b0000000000000000 or 0b0000001000000000 indicates that none of the setpoint dimensions should be ignored. If bit 10 is set the floats afx afy afz should be interpreted as force instead of acceleration. Mapping: bit 1: x, bit 2: y, bit 3: z, bit 4: vx, bit 5: vy, bit 6: vz, bit 7: ax, bit 8: ay, bit 9: az, bit 10: is force setpoint, bit 11: yaw, bit 12: yaw rate
x	float	X Position in NED frame in meters (Units: m)
y	float	Y Position in NED frame in meters (Units: m)
z	float	Z Position in NED frame in meters (note, altitude is negative in NED) (Units: m)
vx	float	X velocity in NED frame in meter / s (Units: m/s)
vy	float	Y velocity in NED frame in meter / s (Units: m/s)
vz	float	Z velocity in NED frame in meter / s (Units: m/s)
afx	float	X acceleration or force (if bit 10 of type_mask is set) in NED frame in meter / s^2 or N (Units: m/s/s)
afy	float	Y acceleration or force (if bit 10 of type_mask is set) in NED frame in meter / s^2 or N (Units: m/s/s)
afz	float	Z acceleration or force (if bit 10 of type_mask is set) in NED frame in meter / s^2 or N (Units: m/s/s)
yaw	float	yaw setpoint in rad (Units: rad)
yaw_rate	float	yaw rate setpoint in rad/s (Units: rad/s)

SET_POSITION_TARGET_GLOBAL_INT (#86)

Sets a desired vehicle position, velocity, and/or acceleration in a global coordinate system (WGS84). Used by an external controller to command the vehicle (manual controller or other system).

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp in milliseconds since system boot. The rationale for the timestamp in the setpoint is to allow the system to compensate for the transport delay of the setpoint. This allows the system to compensate processing latency. (Units: ms)
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
coordinate_frame	uint8_t	Valid options are: MAV_FRAME_GLOBAL_INT = 5, MAV_FRAME_GLOBAL_RELATIVE_ALT_INT = 6, MAV_FRAME_GLOBAL_TERRAIN_ALT_INT = 11 (Enum:MAV_FRAME)
type_mask	uint16_t	Bitmask to indicate which dimensions should be ignored by the vehicle: a value of 0b0000000000000000 or 0b0000001000000000 indicates that none of the setpoint dimensions should be ignored. If bit 10 is set the floats afx afy afz should be interpreted as force instead of acceleration. Mapping: bit 1: x, bit 2: y, bit 3: z, bit 4: vx, bit 5: vy, bit 6: vz, bit 7: ax, bit 8: ay, bit 9: az, bit 10: is force setpoint, bit 11: yaw, bit 12: yaw rate
lat_int	int32_t	X Position in WGS84 frame in 1e7 * degrees (Units: degE7)
lon_int	int32_t	Y Position in WGS84 frame in 1e7 * degrees (Units: degE7)
alt	float	Altitude in meters in AMSL altitude, not WGS84 if absolute or relative, above terrain if GLOBAL_TERRAIN_ALT_INT (Units: m)
vx	float	X velocity in NED frame in meter / s (Units: m/s)
vy	float	Y velocity in NED frame in meter / s (Units: m/s)
vz	float	Z velocity in NED frame in meter / s (Units: m/s)
afx	float	X acceleration or force (if bit 10 of type_mask is set) in NED frame in meter / s^2 or N (Units: m/s/s)
afy	float	Y acceleration or force (if bit 10 of type_mask is set) in NED frame in meter / s^2 or N (Units: m/s/s)
afz	float	Z acceleration or force (if bit 10 of type_mask is set) in NED frame in meter / s^2 or N (Units: m/s/s)
yaw	float	yaw setpoint in rad (Units: rad)
yaw_rate	float	yaw rate setpoint in rad/s (Units: rad/s)

POSITION_TARGET_GLOBAL_INT (#87)

Reports the current commanded vehicle position, velocity, and acceleration as specified by the autopilot. This should match the commands sent in SET_POSITION_TARGET_GLOBAL_INT if the vehicle is being controlled this way.

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp in milliseconds since system boot. The rationale for the timestamp in the setpoint is to allow the system to compensate for the transport delay of the setpoint. This allows the system to compensate processing latency. (Units: ms)
coordinate_frame	uint8_t	Valid options are: MAV_FRAME_GLOBAL_INT = 5, MAV_FRAME_GLOBAL_RELATIVE_ALT_INT = 6, MAV_FRAME_GLOBAL_TERRAIN_ALT_INT = 11 (Enum:MAV_FRAME)
type_mask	uint16_t	Bitmask to indicate which dimensions should be ignored by the vehicle: a value of 0b0000000000000000 or 0b0000001000000000 indicates that none of the setpoint dimensions should be ignored. If bit 10 is set the floats afx afy afz should be interpreted as force instead of acceleration. Mapping: bit 1: x, bit 2: y, bit 3: z, bit 4: vx, bit 5: vy, bit 6: vz, bit 7: ax, bit 8: ay, bit 9: az, bit 10: is force setpoint, bit 11: yaw, bit 12: yaw rate
lat_int	int32_t	X Position in WGS84 frame in 1e7 * degrees (Units: degE7)
lon_int	int32_t	Y Position in WGS84 frame in 1e7 * degrees (Units: degE7)
alt	float	Altitude in meters in AMSL altitude, not WGS84 if absolute or relative, above terrain if GLOBAL_TERRAIN_ALT_INT (Units: m)
vx	float	X velocity in NED frame in meter / s (Units: m/s)
vy	float	Y velocity in NED frame in meter / s (Units: m/s)
vz	float	Z velocity in NED frame in meter / s (Units: m/s)
afx	float	X acceleration or force (if bit 10 of type_mask is set) in NED frame in meter / s^2 or N (Units: m/s/s)
afy	float	Y acceleration or force (if bit 10 of type_mask is set) in NED frame in meter / s^2 or N (Units: m/s/s)
afz	float	Z acceleration or force (if bit 10 of type_mask is set) in NED frame in meter / s^2 or N (Units: m/s/s)
yaw	float	yaw setpoint in rad (Units: rad)
yaw_rate	float	yaw rate setpoint in rad/s (Units: rad/s)

LOCAL_POSITION_NED_SYSTEM_GLOBAL_OFFSET (#89)

The offset in X, Y, Z and yaw between the LOCAL_POSITION_NED messages of MAV X and the global coordinate frame in NED coordinates. Coordinate frame is right-handed, Z-axis down (aeronautical frame, NED / north-east-down convention)

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
x	float	X Position (Units: m)
y	float	Y Position (Units: m)
z	float	Z Position (Units: m)
roll	float	Roll (Units: rad)
pitch	float	Pitch (Units: rad)
yaw	float	Yaw (Units: rad)

HIL_STATE (#90)

DEPRECATED PACKET! Suffers from missing airspeed fields and singularities due to Euler angles. Please use HIL_STATE_QUATERNION instead. Sent from simulation to autopilot. This packet is useful for high throughput applications such as hardware in the loop simulations.

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds since UNIX epoch or microseconds since system boot) (Units: us)
roll	float	Roll angle (rad) (Units: rad)
pitch	float	Pitch angle (rad) (Units: rad)
yaw	float	Yaw angle (rad) (Units: rad)
rollspeed	float	Body frame roll / phi angular speed (rad/s) (Units: rad/s)
pitchspeed	float	Body frame pitch / theta angular speed (rad/s) (Units: rad/s)
yawspeed	float	Body frame yaw / psi angular speed (rad/s) (Units: rad/s)
lat	int32_t	Latitude, expressed as degrees * 1E7 (Units: degE7)
lon	int32_t	Longitude, expressed as degrees * 1E7 (Units: degE7)
alt	int32_t	Altitude in meters, expressed as * 1000 (millimeters) (Units: mm)
vx	int16_t	Ground X Speed (Latitude), expressed as m/s * 100 (Units: cm/s)
vy	int16_t	Ground Y Speed (Longitude), expressed as m/s * 100 (Units: cm/s)
vz	int16_t	Ground Z Speed (Altitude), expressed as m/s * 100 (Units: cm/s)
xacc	int16_t	X acceleration (mg) (Units: mG)
yacc	int16_t	Y acceleration (mg) (Units: mG)
zacc	int16_t	Z acceleration (mg) (Units: mG)

HIL_CONTROLS (#91)

Sent from autopilot to simulation. Hardware in the loop control outputs

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds since UNIX epoch or microseconds since system boot) (Units: us)
roll_ailerons	float	Control output -1 .. 1
pitch_elevator	float	Control output -1 .. 1
yaw_rudder	float	Control output -1 .. 1
throttle	float	Throttle 0 .. 1
aux1	float	Aux 1, -1 .. 1
aux2	float	Aux 2, -1 .. 1
aux3	float	Aux 3, -1 .. 1
aux4	float	Aux 4, -1 .. 1
mode	uint8_t	System mode (MAV_MODE) (Enum:MAV_MODE)
nav_mode	uint8_t	Navigation mode (MAV_NAV_MODE)

HIL_RC_INPUTS_RAW (#92)

Sent from simulation to autopilot. The RAW values of the RC channels received. The standard PPM modulation is as follows: 1000 microseconds: 0%, 2000 microseconds: 100%. Individual receivers/transmitters might violate this specification.

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds since UNIX epoch or microseconds since system boot) (Units: us)
chan1_raw	uint16_t	RC channel 1 value, in microseconds (Units: us)
chan2_raw	uint16_t	RC channel 2 value, in microseconds (Units: us)
chan3_raw	uint16_t	RC channel 3 value, in microseconds (Units: us)
chan4_raw	uint16_t	RC channel 4 value, in microseconds (Units: us)
chan5_raw	uint16_t	RC channel 5 value, in microseconds (Units: us)
chan6_raw	uint16_t	RC channel 6 value, in microseconds (Units: us)
chan7_raw	uint16_t	RC channel 7 value, in microseconds (Units: us)
chan8_raw	uint16_t	RC channel 8 value, in microseconds (Units: us)
chan9_raw	uint16_t	RC channel 9 value, in microseconds (Units: us)
chan10_raw	uint16_t	RC channel 10 value, in microseconds (Units: us)
chan11_raw	uint16_t	RC channel 11 value, in microseconds (Units: us)
chan12_raw	uint16_t	RC channel 12 value, in microseconds (Units: us)
rssi	uint8_t	Receive signal strength indicator, 0: 0%, 255: 100%

HIL_ACTUATOR_CONTROLS (#93)

Sent from autopilot to simulation. Hardware in the loop control outputs (replacement for HIL_CONTROLS)

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds since UNIX epoch or microseconds since system boot) (Units: us)
controls	float[16]	Control outputs -1 .. 1. Channel assignment depends on the simulated hardware.
mode	uint8_t	System mode (MAV_MODE), includes arming state. (Enum:MAV_MODE)
flags	uint64_t	Flags as bitfield, reserved for future use.

OPTICAL_FLOW (#100)

Optical flow from a flow sensor (e.g. optical mouse sensor)

Field Name	Type	Description
time_usec	uint64_t	Timestamp (UNIX) (Units: us)
sensor_id	uint8_t	Sensor ID
flow_x	int16_t	Flow in pixels * 10 in x-sensor direction (dezi-pixels) (Units: dpixels)
flow_y	int16_t	Flow in pixels * 10 in y-sensor direction (dezi-pixels) (Units: dpixels)
flow_comp_m_x	float	Flow in meters in x-sensor direction, angular-speed compensated (Units: m)
flow_comp_m_y	float	Flow in meters in y-sensor direction, angular-speed compensated (Units: m)
quality	uint8_t	Optical flow quality / confidence. 0: bad, 255: maximum quality
ground_distance	float	Ground distance in meters. Positive value: distance known. Negative value: Unknown distance (Units: m)
flow_rate_x **	float	Flow rate in radians/second about X axis (Units: rad/s)
flow_rate_y **	float	Flow rate in radians/second about Y axis (Units: rad/s)

GLOBAL_VISION_POSITION_ESTIMATE (#101)

Field Name	Type	Description
usec	uint64_t	Timestamp (microseconds, synced to UNIX time or since system boot) (Units: us)
x	float	Global X position (Units: m)
y	float	Global Y position (Units: m)
z	float	Global Z position (Units: m)
roll	float	Roll angle in rad (Units: rad)
pitch	float	Pitch angle in rad (Units: rad)
yaw	float	Yaw angle in rad (Units: rad)
covariance **	float[21]	Pose covariance matrix upper right triangular (first six entries are the first ROW, next five entries are the second ROW, etc.)

VISION_POSITION_ESTIMATE (#102)

Field Name	Type	Description
usec	uint64_t	Timestamp (microseconds, synced to UNIX time or since system boot) (Units: us)
x	float	Global X position (Units: m)
y	float	Global Y position (Units: m)
z	float	Global Z position (Units: m)
roll	float	Roll angle in rad (Units: rad)
pitch	float	Pitch angle in rad (Units: rad)
yaw	float	Yaw angle in rad (Units: rad)
covariance **	float[21]	Pose covariance matrix upper right triangular (first six entries are the first ROW, next five entries are the second ROW, etc.)

VISION_SPEED_ESTIMATE (#103)

Field Name	Type	Description
usec	uint64_t	Timestamp (microseconds, synced to UNIX time or since system boot) (Units: us)
x	float	Global X speed (Units: m/s)
y	float	Global Y speed (Units: m/s)
z	float	Global Z speed (Units: m/s)
covariance **	float[9]	Linear velocity covariance matrix (1st three entries - 1st row, etc.)

VICON_POSITION_ESTIMATE (#104)

Field Name	Type	Description
usec	uint64_t	Timestamp (microseconds, synced to UNIX time or since system boot) (Units: us)
x	float	Global X position (Units: m)
y	float	Global Y position (Units: m)
z	float	Global Z position (Units: m)
roll	float	Roll angle in rad (Units: rad)
pitch	float	Pitch angle in rad (Units: rad)
yaw	float	Yaw angle in rad (Units: rad)
covariance **	float[21]	Pose covariance matrix upper right triangular (first six entries are the first ROW, next five entries are the second ROW, etc.)

HIGHRES_IMU (#105)

The IMU readings in SI units in NED body frame

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds, synced to UNIX time or since system boot) (Units: us)
xacc	float	X acceleration (m/s^2) (Units: m/s/s)
yacc	float	Y acceleration (m/s^2) (Units: m/s/s)
zacc	float	Z acceleration (m/s^2) (Units: m/s/s)
xgyro	float	Angular speed around X axis (rad / sec) (Units: rad/s)
ygyro	float	Angular speed around Y axis (rad / sec) (Units: rad/s)
zgyro	float	Angular speed around Z axis (rad / sec) (Units: rad/s)
xmag	float	X Magnetic field (Gauss) (Units: gauss)
ymag	float	Y Magnetic field (Gauss) (Units: gauss)
zmag	float	Z Magnetic field (Gauss) (Units: gauss)
abs_pressure	float	Absolute pressure in millibar (Units: mbar)
diff_pressure	float	Differential pressure in millibar (Units: mbar)
pressure_alt	float	Altitude calculated from pressure
temperature	float	Temperature in degrees celsius (Units: degC)
fields_updated	uint16_t	Bitmask for fields that have updated since last message, bit 0 = xacc, bit 12: temperature

OPTICAL_FLOW_RAD (#106)

Optical flow from an angular rate flow sensor (e.g. PX4FLOW or mouse sensor)

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds, synced to UNIX time or since system boot) (Units: us)
sensor_id	uint8_t	Sensor ID
integration_time_us	uint32_t	Integration time in microseconds. Divide integrated_x and integrated_y by the integration time to obtain average flow. The integration time also indicates the. (Units: us)
integrated_x	float	Flow in radians around X axis (Sensor RH rotation about the X axis induces a positive flow. Sensor linear motion along the positive Y axis induces a negative flow.) (Units: rad)
integrated_y	float	Flow in radians around Y axis (Sensor RH rotation about the Y axis induces a positive flow. Sensor linear motion along the positive X axis induces a positive flow.) (Units: rad)
integrated_xgyro	float	RH rotation around X axis (rad) (Units: rad)
integrated_ygyro	float	RH rotation around Y axis (rad) (Units: rad)
integrated_zgyro	float	RH rotation around Z axis (rad) (Units: rad)
temperature	int16_t	Temperature * 100 in centi-degrees Celsius (Units: cdegC)
quality	uint8_t	Optical flow quality / confidence. 0: no valid flow, 255: maximum quality
time_delta_distance_us	uint32_t	Time in microseconds since the distance was sampled. (Units: us)
distance	float	Distance to the center of the flow field in meters. Positive value (including zero): distance known. Negative value: Unknown distance. (Units: m)

HIL_SENSOR (#107)

The IMU readings in SI units in NED body frame

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds, synced to UNIX time or since system boot) (Units: us)
xacc	float	X acceleration (m/s^2) (Units: m/s/s)
yacc	float	Y acceleration (m/s^2) (Units: m/s/s)
zacc	float	Z acceleration (m/s^2) (Units: m/s/s)
xgyro	float	Angular speed around X axis in body frame (rad / sec) (Units: rad/s)
ygyro	float	Angular speed around Y axis in body frame (rad / sec) (Units: rad/s)
zgyro	float	Angular speed around Z axis in body frame (rad / sec) (Units: rad/s)
xmag	float	X Magnetic field (Gauss) (Units: gauss)
ymag	float	Y Magnetic field (Gauss) (Units: gauss)
zmag	float	Z Magnetic field (Gauss) (Units: gauss)
abs_pressure	float	Absolute pressure in millibar (Units: mbar)
diff_pressure	float	Differential pressure (airspeed) in millibar (Units: mbar)
pressure_alt	float	Altitude calculated from pressure
temperature	float	Temperature in degrees celsius (Units: degC)
fields_updated	uint32_t	Bitmask for fields that have updated since last message, bit 0 = xacc, bit 12: temperature, bit 31: full reset of attitude/position/velocities/etc was performed in sim.

SIM_STATE (#108)

Status of simulation environment, if used

Field Name	Type	Description
q1	float	True attitude quaternion component 1, w (1 in null-rotation)
q2	float	True attitude quaternion component 2, x (0 in null-rotation)
q3	float	True attitude quaternion component 3, y (0 in null-rotation)
q4	float	True attitude quaternion component 4, z (0 in null-rotation)
roll	float	Attitude roll expressed as Euler angles, not recommended except for human-readable outputs
pitch	float	Attitude pitch expressed as Euler angles, not recommended except for human-readable outputs
yaw	float	Attitude yaw expressed as Euler angles, not recommended except for human-readable outputs
xacc	float	X acceleration m/s/s (Units: m/s/s)
yacc	float	Y acceleration m/s/s (Units: m/s/s)
zacc	float	Z acceleration m/s/s (Units: m/s/s)
xgyro	float	Angular speed around X axis rad/s (Units: rad/s)
ygyro	float	Angular speed around Y axis rad/s (Units: rad/s)
zgyro	float	Angular speed around Z axis rad/s (Units: rad/s)
lat	float	Latitude in degrees (Units: deg)
lon	float	Longitude in degrees (Units: deg)
alt	float	Altitude in meters (Units: m)
std_dev_horz	float	Horizontal position standard deviation
std_dev_vert	float	Vertical position standard deviation
vn	float	True velocity in m/s in NORTH direction in earth-fixed NED frame (Units: m/s)
ve	float	True velocity in m/s in EAST direction in earth-fixed NED frame (Units: m/s)
vd	float	True velocity in m/s in DOWN direction in earth-fixed NED frame (Units: m/s)

RADIO_STATUS (#109)

Status generated by radio and injected into MAVLink stream.

Field Name	Type	Description
rssi	uint8_t	Local signal strength
remrssi	uint8_t	Remote signal strength
txbuf	uint8_t	Remaining free buffer space in percent. (Units: %)
noise	uint8_t	Background noise level
remnoise	uint8_t	Remote background noise level
rxerrors	uint16_t	Receive errors
fixed	uint16_t	Count of error corrected packets

FILE_TRANSFER_PROTOCOL (#110)

File transfer message

Field Name	Type	Description
target_network	uint8_t	Network ID (0 for broadcast)
target_system	uint8_t	System ID (0 for broadcast)
target_component	uint8_t	Component ID (0 for broadcast)
payload	uint8_t[251]	Variable length payload. The length is defined by the remaining message length when subtracting the header and other fields. The entire content of this block is opaque unless you understand any the encoding message_type. The particular encoding used can be extension specific and might not always be documented as part of the mavlink specification.

TIMESYNC (#111)

Time synchronization message.

Field Name	Type	Description
tc1	int64_t	Time sync timestamp 1
ts1	int64_t	Time sync timestamp 2

CAMERA_TRIGGER (#112)

Camera-IMU triggering and synchronisation message.

Field Name	Type	Description
time_usec	uint64_t	Timestamp for the image frame in microseconds (Units: us)
seq	uint32_t	Image frame sequence

HIL_GPS (#113)

The global position, as returned by the Global Positioning System (GPS). This is NOT the global position estimate of the sytem, but rather a RAW sensor value. See message GLOBAL_POSITION for the global position estimate. Coordinate frame is right-handed, Z-axis up (GPS frame).

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds since UNIX epoch or microseconds since system boot) (Units: us)
fix_type	uint8_t	0-1: no fix, 2: 2D fix, 3: 3D fix. Some applications will not use the value of this field unless it is at least two, so always correctly fill in the fix.
lat	int32_t	Latitude (WGS84), in degrees * 1E7 (Units: degE7)
lon	int32_t	Longitude (WGS84), in degrees * 1E7 (Units: degE7)
alt	int32_t	Altitude (AMSL, not WGS84), in meters * 1000 (positive for up) (Units: mm)
eph	uint16_t	GPS HDOP horizontal dilution of position in cm (m*100). If unknown, set to: 65535
epv	uint16_t	GPS VDOP vertical dilution of position in cm (m*100). If unknown, set to: 65535
vel	uint16_t	GPS ground speed in cm/s. If unknown, set to: 65535 (Units: cm/s)
vn	int16_t	GPS velocity in cm/s in NORTH direction in earth-fixed NED frame (Units: cm/s)
ve	int16_t	GPS velocity in cm/s in EAST direction in earth-fixed NED frame (Units: cm/s)
vd	int16_t	GPS velocity in cm/s in DOWN direction in earth-fixed NED frame (Units: cm/s)
cog	uint16_t	Course over ground (NOT heading, but direction of movement) in degrees * 100, 0.0..359.99 degrees. If unknown, set to: 65535 (Units: cdeg)
satellites_visible	uint8_t	Number of satellites visible. If unknown, set to 255

HIL_OPTICAL_FLOW (#114)

Simulated optical flow from a flow sensor (e.g. PX4FLOW or optical mouse sensor)

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds, synced to UNIX time or since system boot) (Units: us)
sensor_id	uint8_t	Sensor ID
integration_time_us	uint32_t	Integration time in microseconds. Divide integrated_x and integrated_y by the integration time to obtain average flow. The integration time also indicates the. (Units: us)
integrated_x	float	Flow in radians around X axis (Sensor RH rotation about the X axis induces a positive flow. Sensor linear motion along the positive Y axis induces a negative flow.) (Units: rad)
integrated_y	float	Flow in radians around Y axis (Sensor RH rotation about the Y axis induces a positive flow. Sensor linear motion along the positive X axis induces a positive flow.) (Units: rad)
integrated_xgyro	float	RH rotation around X axis (rad) (Units: rad)
integrated_ygyro	float	RH rotation around Y axis (rad) (Units: rad)
integrated_zgyro	float	RH rotation around Z axis (rad) (Units: rad)
temperature	int16_t	Temperature * 100 in centi-degrees Celsius (Units: cdegC)
quality	uint8_t	Optical flow quality / confidence. 0: no valid flow, 255: maximum quality
time_delta_distance_us	uint32_t	Time in microseconds since the distance was sampled. (Units: us)
distance	float	Distance to the center of the flow field in meters. Positive value (including zero): distance known. Negative value: Unknown distance. (Units: m)

HIL_STATE_QUATERNION (#115)

Sent from simulation to autopilot, avoids in contrast to HIL_STATE singularities. This packet is useful for high throughput applications such as hardware in the loop simulations.

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds since UNIX epoch or microseconds since system boot) (Units: us)
attitude_quaternion	float[4]	Vehicle attitude expressed as normalized quaternion in w, x, y, z order (with 1 0 0 0 being the null-rotation)
rollspeed	float	Body frame roll / phi angular speed (rad/s) (Units: rad/s)
pitchspeed	float	Body frame pitch / theta angular speed (rad/s) (Units: rad/s)
yawspeed	float	Body frame yaw / psi angular speed (rad/s) (Units: rad/s)
lat	int32_t	Latitude, expressed as degrees * 1E7 (Units: degE7)
lon	int32_t	Longitude, expressed as degrees * 1E7 (Units: degE7)
alt	int32_t	Altitude in meters, expressed as * 1000 (millimeters) (Units: mm)
vx	int16_t	Ground X Speed (Latitude), expressed as cm/s (Units: cm/s)
vy	int16_t	Ground Y Speed (Longitude), expressed as cm/s (Units: cm/s)
vz	int16_t	Ground Z Speed (Altitude), expressed as cm/s (Units: cm/s)
ind_airspeed	uint16_t	Indicated airspeed, expressed as cm/s (Units: cm/s)
true_airspeed	uint16_t	True airspeed, expressed as cm/s (Units: cm/s)
xacc	int16_t	X acceleration (mg) (Units: mG)
yacc	int16_t	Y acceleration (mg) (Units: mG)
zacc	int16_t	Z acceleration (mg) (Units: mG)

SCALED_IMU2 (#116)

The RAW IMU readings for secondary 9DOF sensor setup. This message should contain the scaled values to the described units

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
xacc	int16_t	X acceleration (mg) (Units: mG)
yacc	int16_t	Y acceleration (mg) (Units: mG)
zacc	int16_t	Z acceleration (mg) (Units: mG)
xgyro	int16_t	Angular speed around X axis (millirad /sec) (Units: mrad/s)
ygyro	int16_t	Angular speed around Y axis (millirad /sec) (Units: mrad/s)
zgyro	int16_t	Angular speed around Z axis (millirad /sec) (Units: mrad/s)
xmag	int16_t	X Magnetic field (milli tesla) (Units: mT)
ymag	int16_t	Y Magnetic field (milli tesla) (Units: mT)
zmag	int16_t	Z Magnetic field (milli tesla) (Units: mT)

LOG_REQUEST_LIST (#117)

Request a list of available logs. On some systems calling this may stop on-board logging until LOG_REQUEST_END is called.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
start	uint16_t	First log id (0 for first available)
end	uint16_t	Last log id (0xffff for last available)

LOG_ENTRY (#118)

Reply to LOG_REQUEST_LIST

Field Name	Type	Description
id	uint16_t	Log id
num_logs	uint16_t	Total number of logs
last_log_num	uint16_t	High log number
time_utc	uint32_t	UTC timestamp of log in seconds since 1970, or 0 if not available (Units: s)
size	uint32_t	Size of the log (may be approximate) in bytes (Units: bytes)

LOG_REQUEST_DATA (#119)

Request a chunk of a log

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
id	uint16_t	Log id (from LOG_ENTRY reply)
ofs	uint32_t	Offset into the log
count	uint32_t	Number of bytes (Units: bytes)

LOG_DATA (#120)

Reply to LOG_REQUEST_DATA

Field Name	Type	Description
id	uint16_t	Log id (from LOG_ENTRY reply)
ofs	uint32_t	Offset into the log
count	uint8_t	Number of bytes (zero for end of log) (Units: bytes)
data	uint8_t[90]	log data

LOG_ERASE (#121)

Erase all logs

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID

LOG_REQUEST_END (#122)

Stop log transfer and resume normal logging

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID

GPS_INJECT_DATA (#123)

data for injecting into the onboard GPS (used for DGPS)

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
len	uint8_t	data length (Units: bytes)
data	uint8_t[110]	raw data (110 is enough for 12 satellites of RTCMv2)

GPS2_RAW (#124)

Second GPS data. Coordinate frame is right-handed, Z-axis up (GPS frame).

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds since UNIX epoch or microseconds since system boot) (Units: us)
fix_type	uint8_t	See the GPS_FIX_TYPE enum. (Enum:GPS_FIX_TYPE)
lat	int32_t	Latitude (WGS84), in degrees * 1E7 (Units: degE7)
lon	int32_t	Longitude (WGS84), in degrees * 1E7 (Units: degE7)
alt	int32_t	Altitude (AMSL, not WGS84), in meters * 1000 (positive for up) (Units: mm)
eph	uint16_t	GPS HDOP horizontal dilution of position in cm (m*100). If unknown, set to: UINT16_MAX (Units: cm)
epv	uint16_t	GPS VDOP vertical dilution of position in cm (m*100). If unknown, set to: UINT16_MAX (Units: cm)
vel	uint16_t	GPS ground speed (m/s * 100). If unknown, set to: UINT16_MAX (Units: cm/s)
cog	uint16_t	Course over ground (NOT heading, but direction of movement) in degrees * 100, 0.0..359.99 degrees. If unknown, set to: UINT16_MAX (Units: cdeg)
satellites_visible	uint8_t	Number of satellites visible. If unknown, set to 255
dgps_numch	uint8_t	Number of DGPS satellites
dgps_age	uint32_t	Age of DGPS info (Units: ms)

POWER_STATUS (#125)

Power supply status

Field Name	Type	Description
Vcc	uint16_t	5V rail voltage in millivolts (Units: mV)
Vservo	uint16_t	servo rail voltage in millivolts (Units: mV)
flags	uint16_t	power supply status flags (see MAV_POWER_STATUS enum) (Enum:MAV_POWER_STATUS)

SERIAL_CONTROL (#126)

Control a serial port. This can be used for raw access to an onboard serial peripheral such as a GPS or telemetry radio. It is designed to make it possible to update the devices firmware via MAVLink messages or change the devices settings. A message with zero bytes can be used to change just the baudrate.

Field Name	Type	Description
device	uint8_t	See SERIAL_CONTROL_DEV enum (Enum:SERIAL_CONTROL_DEV)
flags	uint8_t	See SERIAL_CONTROL_FLAG enum (Enum:SERIAL_CONTROL_FLAG)
timeout	uint16_t	Timeout for reply data in milliseconds (Units: ms)
baudrate	uint32_t	Baudrate of transfer. Zero means no change. (Units: bits/s)
count	uint8_t	how many bytes in this transfer (Units: bytes)
data	uint8_t[70]	serial data

GPS_RTK (#127)

RTK GPS data. Gives information on the relative baseline calculation the GPS is reporting

Field Name	Type	Description
time_last_baseline_ms	uint32_t	Time since boot of last baseline message received in ms. (Units: ms)
rtk_receiver_id	uint8_t	Identification of connected RTK receiver.
wn	uint16_t	GPS Week Number of last baseline
tow	uint32_t	GPS Time of Week of last baseline (Units: ms)
rtk_health	uint8_t	GPS-specific health report for RTK data.
rtk_rate	uint8_t	Rate of baseline messages being received by GPS, in HZ (Units: Hz)
nsats	uint8_t	Current number of sats used for RTK calculation.
baseline_coords_type	uint8_t	Coordinate system of baseline (Enum:RTK_BASELINE_COORDINATE_SYSTEM)
baseline_a_mm	int32_t	Current baseline in ECEF x or NED north component in mm. (Units: mm)
baseline_b_mm	int32_t	Current baseline in ECEF y or NED east component in mm. (Units: mm)
baseline_c_mm	int32_t	Current baseline in ECEF z or NED down component in mm. (Units: mm)
accuracy	uint32_t	Current estimate of baseline accuracy.
iar_num_hypotheses	int32_t	Current number of integer ambiguity hypotheses.

GPS2_RTK (#128)

RTK GPS data. Gives information on the relative baseline calculation the GPS is reporting

Field Name	Type	Description
time_last_baseline_ms	uint32_t	Time since boot of last baseline message received in ms. (Units: ms)
rtk_receiver_id	uint8_t	Identification of connected RTK receiver.
wn	uint16_t	GPS Week Number of last baseline
tow	uint32_t	GPS Time of Week of last baseline (Units: ms)
rtk_health	uint8_t	GPS-specific health report for RTK data.
rtk_rate	uint8_t	Rate of baseline messages being received by GPS, in HZ (Units: Hz)
nsats	uint8_t	Current number of sats used for RTK calculation.
baseline_coords_type	uint8_t	Coordinate system of baseline (Enum:RTK_BASELINE_COORDINATE_SYSTEM)
baseline_a_mm	int32_t	Current baseline in ECEF x or NED north component in mm. (Units: mm)
baseline_b_mm	int32_t	Current baseline in ECEF y or NED east component in mm. (Units: mm)
baseline_c_mm	int32_t	Current baseline in ECEF z or NED down component in mm. (Units: mm)
accuracy	uint32_t	Current estimate of baseline accuracy.
iar_num_hypotheses	int32_t	Current number of integer ambiguity hypotheses.

SCALED_IMU3 (#129)

The RAW IMU readings for 3rd 9DOF sensor setup. This message should contain the scaled values to the described units

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
xacc	int16_t	X acceleration (mg) (Units: mG)
yacc	int16_t	Y acceleration (mg) (Units: mG)
zacc	int16_t	Z acceleration (mg) (Units: mG)
xgyro	int16_t	Angular speed around X axis (millirad /sec) (Units: mrad/s)
ygyro	int16_t	Angular speed around Y axis (millirad /sec) (Units: mrad/s)
zgyro	int16_t	Angular speed around Z axis (millirad /sec) (Units: mrad/s)
xmag	int16_t	X Magnetic field (milli tesla) (Units: mT)
ymag	int16_t	Y Magnetic field (milli tesla) (Units: mT)
zmag	int16_t	Z Magnetic field (milli tesla) (Units: mT)

DATA_TRANSMISSION_HANDSHAKE (#130)

Field Name	Type	Description
type	uint8_t	type of requested/acknowledged data (as defined in ENUM DATA_TYPES in mavlink/include/mavlink_types.h)
size	uint32_t	total data size in bytes (set on ACK only) (Units: bytes)
width	uint16_t	Width of a matrix or image
height	uint16_t	Height of a matrix or image
packets	uint16_t	number of packets beeing sent (set on ACK only)
payload	uint8_t	payload size per packet (normally 253 byte, see DATA field size in message ENCAPSULATED_DATA) (set on ACK only) (Units: bytes)
jpg_quality	uint8_t	JPEG quality out of [1,100] (Units: %)

ENCAPSULATED_DATA (#131)

Field Name	Type	Description
seqnr	uint16_t	sequence number (starting with 0 on every transmission)
data	uint8_t[253]	image data bytes

DISTANCE_SENSOR (#132)

Field Name	Type	Description
time_boot_ms	uint32_t	Time since system boot (Units: ms)
min_distance	uint16_t	Minimum distance the sensor can measure in centimeters (Units: cm)
max_distance	uint16_t	Maximum distance the sensor can measure in centimeters (Units: cm)
current_distance	uint16_t	Current distance reading (Units: cm)
type	uint8_t	Type from MAV_DISTANCE_SENSOR enum. (Enum:MAV_DISTANCE_SENSOR)
id	uint8_t	Onboard ID of the sensor
orientation	uint8_t	Direction the sensor faces from MAV_SENSOR_ORIENTATION enum. downward-facing: ROTATION_PITCH_270, upward-facing: ROTATION_PITCH_90, backward-facing: ROTATION_PITCH_180, forward-facing: ROTATION_NONE, left-facing: ROTATION_YAW_90, right-facing: ROTATION_YAW_270 (Enum:MAV_SENSOR_ORIENTATION)
covariance	uint8_t	Measurement covariance in centimeters, 0 for unknown / invalid readings (Units: cm)

TERRAIN_REQUEST (#133)

Request for terrain data and terrain status

Field Name	Type	Description
lat	int32_t	Latitude of SW corner of first grid (degrees *10^7) (Units: degE7)
lon	int32_t	Longitude of SW corner of first grid (in degrees *10^7) (Units: degE7)
grid_spacing	uint16_t	Grid spacing in meters (Units: m)
mask	uint64_t	Bitmask of requested 4x4 grids (row major 8x7 array of grids, 56 bits)

TERRAIN_DATA (#134)

Terrain data sent from GCS. The lat/lon and grid_spacing must be the same as a lat/lon from a TERRAIN_REQUEST

Field Name	Type	Description
lat	int32_t	Latitude of SW corner of first grid (degrees *10^7) (Units: degE7)
lon	int32_t	Longitude of SW corner of first grid (in degrees *10^7) (Units: degE7)
grid_spacing	uint16_t	Grid spacing in meters (Units: m)
gridbit	uint8_t	bit within the terrain request mask
data	int16_t[16]	Terrain data in meters AMSL (Units: m)

TERRAIN_CHECK (#135)

Request that the vehicle report terrain height at the given location. Used by GCS to check if vehicle has all terrain data needed for a mission.

Field Name	Type	Description
lat	int32_t	Latitude (degrees *10^7) (Units: degE7)
lon	int32_t	Longitude (degrees *10^7) (Units: degE7)

TERRAIN_REPORT (#136)

Response from a TERRAIN_CHECK request

Field Name	Type	Description
lat	int32_t	Latitude (degrees *10^7) (Units: degE7)
lon	int32_t	Longitude (degrees *10^7) (Units: degE7)
spacing	uint16_t	grid spacing (zero if terrain at this location unavailable)
terrain_height	float	Terrain height in meters AMSL (Units: m)
current_height	float	Current vehicle height above lat/lon terrain height (meters) (Units: m)
pending	uint16_t	Number of 4x4 terrain blocks waiting to be received or read from disk
loaded	uint16_t	Number of 4x4 terrain blocks in memory

SCALED_PRESSURE2 (#137)

Barometer readings for 2nd barometer

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
press_abs	float	Absolute pressure (hectopascal) (Units: hPa)
press_diff	float	Differential pressure 1 (hectopascal) (Units: hPa)
temperature	int16_t	Temperature measurement (0.01 degrees celsius) (Units: cdegC)

ATT_POS_MOCAP (#138)

Motion capture attitude and position

Field Name	Type	Description
time_usec	uint64_t	Timestamp (micros since boot or Unix epoch) (Units: us)
q	float[4]	Attitude quaternion (w, x, y, z order, zero-rotation is 1, 0, 0, 0)
x	float	X position in meters (NED) (Units: m)
y	float	Y position in meters (NED) (Units: m)
z	float	Z position in meters (NED) (Units: m)
covariance **	float[21]	Pose covariance matrix upper right triangular (first six entries are the first ROW, next five entries are the second ROW, etc.)

SET_ACTUATOR_CONTROL_TARGET (#139)

Set the vehicle attitude and body angular rates.

Field Name	Type	Description
time_usec	uint64_t	Timestamp (micros since boot or Unix epoch) (Units: us)
group_mlx	uint8_t	Actuator group. The "_mlx" indicates this is a multi-instance message and a MAVLink parser should use this field to difference between instances.
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
controls	float[8]	Actuator controls. Normed to -1..+1 where 0 is neutral position. Throttle for single rotation direction motors is 0..1, negative range for reverse direction. Standard mapping for attitude controls (group 0): (index 0-7): roll, pitch, yaw, throttle, flaps, spoilers, airbrakes, landing gear. Load a pass-through mixer to repurpose them as generic outputs.

ACTUATOR_CONTROL_TARGET (#140)

Set the vehicle attitude and body angular rates.

Field Name	Type	Description
time_usec	uint64_t	Timestamp (micros since boot or Unix epoch) (Units: us)
group_mlx	uint8_t	Actuator group. The "_mlx" indicates this is a multi-instance message and a MAVLink parser should use this field to difference between instances.
controls	float[8]	Actuator controls. Normed to -1..+1 where 0 is neutral position. Throttle for single rotation direction motors is 0..1, negative range for reverse direction. Standard mapping for attitude controls (group 0): (index 0-7): roll, pitch, yaw, throttle, flaps, spoilers, airbrakes, landing gear. Load a pass-through mixer to repurpose them as generic outputs.

ALTITUDE (#141)

The current system altitude.

Field Name	Type	Description
time_usec	uint64_t	Timestamp (micros since boot or Unix epoch) (Units: us)
altitude_monotonic	float	This altitude measure is initialized on system boot and monotonic (it is never reset, but represents the local altitude change). The only guarantee on this field is that it will never be reset and is consistent within a flight. The recommended value for this field is the uncorrected barometric altitude at boot time. This altitude will also drift and vary between flights. (Units: m)
altitude_amsl	float	This altitude measure is strictly above mean sea level and might be non-monotonic (it might reset on events like GPS lock or when a new QNH value is set). It should be the altitude to which global altitude waypoints are compared to. Note that it is *not* the GPS altitude, however, most GPS modules already output AMSL by default and not the WGS84 altitude. (Units: m)
altitude_local	float	This is the local altitude in the local coordinate frame. It is not the altitude above home, but in reference to the coordinate origin (0, 0, 0). It is up-positive. (Units: m)
altitude_relative	float	This is the altitude above the home position. It resets on each change of the current home position. (Units: m)
altitude_terrain	float	This is the altitude above terrain. It might be fed by a terrain database or an altimeter. Values smaller than -1000 should be interpreted as unknown. (Units: m)
bottom_clearance	float	This is not the altitude, but the clear space below the system according to the fused clearance estimate. It generally should max out at the maximum range of e.g. the laser altimeter. It is generally a moving target. A negative value indicates no measurement available. (Units: m)

RESOURCE_REQUEST (#142)

The autopilot is requesting a resource (file, binary, other type of data)

Field Name	Type	Description
request_id	uint8_t	Request ID. This ID should be re-used when sending back URI contents
uri_type	uint8_t	The type of requested URI. 0 = a file via URL. 1 = a UAVCAN binary
uri	uint8_t[120]	The requested unique resource identifier (URI). It is not necessarily a straight domain name (depends on the URI type enum)
transfer_type	uint8_t	The way the autopilot wants to receive the URI. 0 = MAVLink FTP. 1 = binary stream.
storage	uint8_t[120]	The storage path the autopilot wants the URI to be stored in. Will only be valid if the transfer_type has a storage associated (e.g. MAVLink FTP).

SCALED_PRESSURE3 (#143)

Barometer readings for 3rd barometer

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
press_abs	float	Absolute pressure (hectopascal) (Units: hPa)
press_diff	float	Differential pressure 1 (hectopascal) (Units: hPa)
temperature	int16_t	Temperature measurement (0.01 degrees celsius) (Units: cdegC)

FOLLOW_TARGET (#144)

current motion information from a designated system

Field Name	Type	Description
timestamp	uint64_t	Timestamp in milliseconds since system boot (Units: ms)
est_capabilities	uint8_t	bit positions for tracker reporting capabilities (POS = 0, VEL = 1, ACCEL = 2, ATT + RATES = 3)
lat	int32_t	Latitude (WGS84), in degrees * 1E7 (Units: degE7)
lon	int32_t	Longitude (WGS84), in degrees * 1E7 (Units: degE7)
alt	float	AMSL, in meters (Units: m)
vel	float[3]	target velocity (0,0,0) for unknown (Units: m/s)
acc	float[3]	linear target acceleration (0,0,0) for unknown (Units: m/s/s)
attitude_q	float[4]	(1 0 0 0 for unknown)
rates	float[3]	(0 0 0 for unknown)
position_cov	float[3]	eph epv
custom_state	uint64_t	button states or switches of a tracker device

CONTROL_SYSTEM_STATE (#146)

The smoothed, monotonic system state used to feed the control loops of the system.

Field Name	Type	Description
time_usec	uint64_t	Timestamp (micros since boot or Unix epoch) (Units: us)
x_acc	float	X acceleration in body frame (Units: m/s/s)
y_acc	float	Y acceleration in body frame (Units: m/s/s)
z_acc	float	Z acceleration in body frame (Units: m/s/s)
x_vel	float	X velocity in body frame (Units: m/s)
y_vel	float	Y velocity in body frame (Units: m/s)
z_vel	float	Z velocity in body frame (Units: m/s)
x_pos	float	X position in local frame (Units: m)
y_pos	float	Y position in local frame (Units: m)
z_pos	float	Z position in local frame (Units: m)
airspeed	float	Airspeed, set to -1 if unknown (Units: m/s)
vel_variance	float[3]	Variance of body velocity estimate
pos_variance	float[3]	Variance in local position
q	float[4]	The attitude, represented as Quaternion
roll_rate	float	Angular rate in roll axis (Units: rad/s)
pitch_rate	float	Angular rate in pitch axis (Units: rad/s)
yaw_rate	float	Angular rate in yaw axis (Units: rad/s)

BATTERY_STATUS (#147)

Battery information

Field Name	Type	Description
id	uint8_t	Battery ID
battery_function	uint8_t	Function of the battery (Enum:MAV_BATTERY_FUNCTION)
type	uint8_t	Type (chemistry) of the battery (Enum:MAV_BATTERY_TYPE)
temperature	int16_t	Temperature of the battery in centi-degrees celsius. INT16_MAX for unknown temperature. (Units: cdegC)
voltages	uint16_t[10]	Battery voltage of cells, in millivolts (1 = 1 millivolt). Cells above the valid cell count for this battery should have the UINT16_MAX value. (Units: mV)
current_battery	int16_t	Battery current, in 10*milliamperes (1 = 10 milliampere), -1: autopilot does not measure the current (Units: cA)
current_consumed	int32_t	Consumed charge, in milliampere hours (1 = 1 mAh), -1: autopilot does not provide mAh consumption estimate (Units: mAh)
energy_consumed	int32_t	Consumed energy, in HectoJoules (intergrated U*I*dt) (1 = 100 Joule), -1: autopilot does not provide energy consumption estimate (Units: hJ)
battery_remaining	int8_t	Remaining battery energy: (0%: 0, 100%: 100), -1: autopilot does not estimate the remaining battery (Units: %)
time_remaining **	int32_t	Remaining battery time, in seconds (1 = 1s = 0% energy left), 0: autopilot does not provide remaining battery time estimate (Units: s)
charge_state **	uint8_t	State for extent of discharge, provided by autopilot for warning or external reactions (Enum:MAV_BATTERY_CHARGE_STATE)

AUTOPILOT_VERSION (#148)

Version and capability of autopilot software

Field Name	Type	Description
capabilities	uint64_t	bitmask of capabilities (see MAV_PROTOCOL_CAPABILITY enum) (Enum:MAV_PROTOCOL_CAPABILITY)
flight_sw_version	uint32_t	Firmware version number
middleware_sw_version	uint32_t	Middleware version number
os_sw_version	uint32_t	Operating system version number
board_version	uint32_t	HW / board version (last 8 bytes should be silicon ID, if any)
flight_custom_version	uint8_t[8]	Custom version field, commonly the first 8 bytes of the git hash. This is not an unique identifier, but should allow to identify the commit using the main version number even for very large code bases.
middleware_custom_version	uint8_t[8]	Custom version field, commonly the first 8 bytes of the git hash. This is not an unique identifier, but should allow to identify the commit using the main version number even for very large code bases.
os_custom_version	uint8_t[8]	Custom version field, commonly the first 8 bytes of the git hash. This is not an unique identifier, but should allow to identify the commit using the main version number even for very large code bases.
vendor_id	uint16_t	ID of the board vendor
product_id	uint16_t	ID of the product
uid	uint64_t	UID if provided by hardware (see uid2)
uid2 **	uint8_t[18]	UID if provided by hardware (supersedes the uid field. If this is non-zero, use this field, otherwise use uid)

LANDING_TARGET (#149)

The location of a landing area captured from a downward facing camera

Field Name	Type	Description
time_usec	uint64_t	Timestamp (micros since boot or Unix epoch) (Units: us)
target_num	uint8_t	The ID of the target if multiple targets are present
frame	uint8_t	MAV_FRAME enum specifying the whether the following feilds are earth-frame, body-frame, etc. (Enum:MAV_FRAME)
angle_x	float	X-axis angular offset (in radians) of the target from the center of the image (Units: rad)
angle_y	float	Y-axis angular offset (in radians) of the target from the center of the image (Units: rad)
distance	float	Distance to the target from the vehicle in meters (Units: m)
size_x	float	Size in radians of target along x-axis (Units: rad)
size_y	float	Size in radians of target along y-axis (Units: rad)
x **	float	X Position of the landing target on MAV_FRAME (Units: m)
y **	float	Y Position of the landing target on MAV_FRAME (Units: m)
z **	float	Z Position of the landing target on MAV_FRAME (Units: m)
q **	float[4]	Quaternion of landing target orientation (w, x, y, z order, zero-rotation is 1, 0, 0, 0)
type **	uint8_t	LANDING_TARGET_TYPE enum specifying the type of landing target (Enum:LANDING_TARGET_TYPE)
position_valid **	uint8_t	Boolean indicating known position (1) or default unkown position (0), for validation of positioning of the landing target

ESTIMATOR_STATUS (#230)

Estimator status message including flags, innovation test ratios and estimated accuracies. The flags message is an integer bitmask containing information on which EKF outputs are valid. See the ESTIMATOR_STATUS_FLAGS enum definition for further information. The innovaton test ratios show the magnitude of the sensor innovation divided by the innovation check threshold. Under normal operation the innovaton test ratios should be below 0.5 with occasional values up to 1.0. Values greater than 1.0 should be rare under normal operation and indicate that a measurement has been rejected by the filter. The user should be notified if an innovation test ratio greater than 1.0 is recorded. Notifications for values in the range between 0.5 and 1.0 should be optional and controllable by the user.

Field Name	Type	Description
time_usec	uint64_t	Timestamp (micros since boot or Unix epoch) (Units: us)
flags	uint16_t	Integer bitmask indicating which EKF outputs are valid. See definition for ESTIMATOR_STATUS_FLAGS. (Enum:ESTIMATOR_STATUS_FLAGS)
vel_ratio	float	Velocity innovation test ratio
pos_horiz_ratio	float	Horizontal position innovation test ratio
pos_vert_ratio	float	Vertical position innovation test ratio
mag_ratio	float	Magnetometer innovation test ratio
hagl_ratio	float	Height above terrain innovation test ratio
tas_ratio	float	True airspeed innovation test ratio
pos_horiz_accuracy	float	Horizontal position 1-STD accuracy relative to the EKF local origin (m) (Units: m)
pos_vert_accuracy	float	Vertical position 1-STD accuracy relative to the EKF local origin (m) (Units: m)

WIND_COV (#231)

Field Name	Type	Description
time_usec	uint64_t	Timestamp (micros since boot or Unix epoch) (Units: us)
wind_x	float	Wind in X (NED) direction in m/s (Units: m/s)
wind_y	float	Wind in Y (NED) direction in m/s (Units: m/s)
wind_z	float	Wind in Z (NED) direction in m/s (Units: m/s)
var_horiz	float	Variability of the wind in XY. RMS of a 1 Hz lowpassed wind estimate. (Units: m/s)
var_vert	float	Variability of the wind in Z. RMS of a 1 Hz lowpassed wind estimate. (Units: m/s)
wind_alt	float	AMSL altitude (m) this measurement was taken at (Units: m)
horiz_accuracy	float	Horizontal speed 1-STD accuracy (Units: m)
vert_accuracy	float	Vertical speed 1-STD accuracy (Units: m)

GPS_INPUT (#232)

GPS sensor input message. This is a raw sensor value sent by the GPS. This is NOT the global position estimate of the sytem.

Field Name	Type	Description
time_usec	uint64_t	Timestamp (micros since boot or Unix epoch) (Units: us)
gps_id	uint8_t	ID of the GPS for multiple GPS inputs
ignore_flags	uint16_t	Flags indicating which fields to ignore (see GPS_INPUT_IGNORE_FLAGS enum). All other fields must be provided. (Enum:GPS_INPUT_IGNORE_FLAGS)
time_week_ms	uint32_t	GPS time (milliseconds from start of GPS week) (Units: ms)
time_week	uint16_t	GPS week number
fix_type	uint8_t	0-1: no fix, 2: 2D fix, 3: 3D fix. 4: 3D with DGPS. 5: 3D with RTK
lat	int32_t	Latitude (WGS84), in degrees * 1E7 (Units: degE7)
lon	int32_t	Longitude (WGS84), in degrees * 1E7 (Units: degE7)
alt	float	Altitude (AMSL, not WGS84), in m (positive for up) (Units: m)
hdop	float	GPS HDOP horizontal dilution of position in m (Units: m)
vdop	float	GPS VDOP vertical dilution of position in m (Units: m)
vn	float	GPS velocity in m/s in NORTH direction in earth-fixed NED frame (Units: m/s)
ve	float	GPS velocity in m/s in EAST direction in earth-fixed NED frame (Units: m/s)
vd	float	GPS velocity in m/s in DOWN direction in earth-fixed NED frame (Units: m/s)
speed_accuracy	float	GPS speed accuracy in m/s (Units: m/s)
horiz_accuracy	float	GPS horizontal accuracy in m (Units: m)
vert_accuracy	float	GPS vertical accuracy in m (Units: m)
satellites_visible	uint8_t	Number of satellites visible.

GPS_RTCM_DATA (#233)

RTCM message for injecting into the onboard GPS (used for DGPS)

Field Name	Type	Description
flags	uint8_t	LSB: 1 means message is fragmented, next 2 bits are the fragment ID, the remaining 5 bits are used for the sequence ID. Messages are only to be flushed to the GPS when the entire message has been reconstructed on the autopilot. The fragment ID specifies which order the fragments should be assembled into a buffer, while the sequence ID is used to detect a mismatch between different buffers. The buffer is considered fully reconstructed when either all 4 fragments are present, or all the fragments before the first fragment with a non full payload is received. This management is used to ensure that normal GPS operation doesn't corrupt RTCM data, and to recover from a unreliable transport delivery order.
len	uint8_t	data length (Units: bytes)
data	uint8_t[180]	RTCM message (may be fragmented)

HIGH_LATENCY (#234)

Message appropriate for high latency connections like Iridium

Field Name	Type	Description
base_mode	uint8_t	System mode bitfield, as defined by MAV_MODE_FLAG enum. (Enum:MAV_MODE_FLAG)
custom_mode	uint32_t	A bitfield for use for autopilot-specific flags.
landed_state	uint8_t	The landed state. Is set to MAV_LANDED_STATE_UNDEFINED if landed state is unknown. (Enum:MAV_LANDED_STATE)
roll	int16_t	roll (centidegrees) (Units: cdeg)
pitch	int16_t	pitch (centidegrees) (Units: cdeg)
heading	uint16_t	heading (centidegrees) (Units: cdeg)
throttle	int8_t	throttle (percentage) (Units: %)
heading_sp	int16_t	heading setpoint (centidegrees) (Units: cdeg)
latitude	int32_t	Latitude, expressed as degrees * 1E7 (Units: degE7)
longitude	int32_t	Longitude, expressed as degrees * 1E7 (Units: degE7)
altitude_amsl	int16_t	Altitude above mean sea level (meters) (Units: m)
altitude_sp	int16_t	Altitude setpoint relative to the home position (meters) (Units: m)
airspeed	uint8_t	airspeed (m/s) (Units: m/s)
airspeed_sp	uint8_t	airspeed setpoint (m/s) (Units: m/s)
groundspeed	uint8_t	groundspeed (m/s) (Units: m/s)
climb_rate	int8_t	climb rate (m/s) (Units: m/s)
gps_nsat	uint8_t	Number of satellites visible. If unknown, set to 255
gps_fix_type	uint8_t	See the GPS_FIX_TYPE enum. (Enum:GPS_FIX_TYPE)
battery_remaining	uint8_t	Remaining battery (percentage) (Units: %)
temperature	int8_t	Autopilot temperature (degrees C) (Units: degC)
temperature_air	int8_t	Air temperature (degrees C) from airspeed sensor (Units: degC)
failsafe	uint8_t	failsafe (each bit represents a failsafe where 0=ok, 1=failsafe active (bit0:RC, bit1:batt, bit2:GPS, bit3:GCS, bit4:fence)
wp_num	uint8_t	current waypoint number
wp_distance	uint16_t	distance to target (meters) (Units: m)

HIGH_LATENCY2 (#235)

WIP: Message appropriate for high latency connections like Iridium (version 2)

Field Name	Type	Description
timestamp	uint32_t	Timestamp (milliseconds since boot or Unix epoch) (Units: ms)
type	uint8_t	Type of the MAV (quadrotor, helicopter, etc., up to 15 types, defined in MAV_TYPE ENUM) (Enum:MAV_TYPE)
autopilot	uint8_t	Autopilot type / class. defined in MAV_AUTOPILOT ENUM (Enum:MAV_AUTOPILOT)
custom_mode	uint16_t	A bitfield for use for autopilot-specific flags (2 byte version).
latitude	int32_t	Latitude, expressed as degrees * 1E7 (Units: degE7)
longitude	int32_t	Longitude, expressed as degrees * 1E7 (Units: degE7)
altitude	int16_t	Altitude above mean sea level (Units: m)
target_altitude	int16_t	Altitude setpoint (Units: m)
heading	uint8_t	Heading (degrees / 2) (Units: deg/2)
target_heading	uint8_t	Heading setpoint (degrees / 2) (Units: deg/2)
target_distance	uint16_t	Distance to target waypoint or position (meters / 10) (Units: dam)
throttle	uint8_t	Throttle (percentage) (Units: %)
airspeed	uint8_t	Airspeed (m/s * 5) (Units: m/s*5)
airspeed_sp	uint8_t	Airspeed setpoint (m/s * 5) (Units: m/s*5)
groundspeed	uint8_t	Groundspeed (m/s * 5) (Units: m/s*5)
windspeed	uint8_t	Windspeed (m/s * 5) (Units: m/s*5)
wind_heading	uint8_t	Wind heading (deg / 2) (Units: deg/2)
eph	uint8_t	Maximum error horizontal position since last message (m * 10) (Units: dm)
epv	uint8_t	Maximum error vertical position since last message (m * 10) (Units: dm)
temperature_air	int8_t	Air temperature (degrees C) from airspeed sensor (Units: degC)
climb_rate	int8_t	Maximum climb rate magnitude since last message (m/s * 10) (Units: dm/s)
battery	int8_t	Battery (percentage, -1 for DNU) (Units: %)
wp_num	uint16_t	Current waypoint number
failure_flags	uint16_t	Indicates failures as defined in HL_FAILURE_FLAG ENUM. (Enum:HL_FAILURE_FLAG)
custom0	int8_t	Field for custom payload.
custom1	int8_t	Field for custom payload.
custom2	int8_t	Field for custom payload.

VIBRATION (#241)

Vibration levels and accelerometer clipping

Field Name	Type	Description
time_usec	uint64_t	Timestamp (micros since boot or Unix epoch) (Units: us)
vibration_x	float	Vibration levels on X-axis
vibration_y	float	Vibration levels on Y-axis
vibration_z	float	Vibration levels on Z-axis
clipping_0	uint32_t	first accelerometer clipping count
clipping_1	uint32_t	second accelerometer clipping count
clipping_2	uint32_t	third accelerometer clipping count

HOME_POSITION (#242)

This message can be requested by sending the MAV_CMD_GET_HOME_POSITION command. The position the system will return to and land on. The position is set automatically by the system during the takeoff in case it was not explicitely set by the operator before or after. The position the system will return to and land on. The global and local positions encode the position in the respective coordinate frames, while the q parameter encodes the orientation of the surface. Under normal conditions it describes the heading and terrain slope, which can be used by the aircraft to adjust the approach. The approach 3D vector describes the point to which the system should fly in normal flight mode and then perform a landing sequence along the vector.

Field Name	Type	Description
latitude	int32_t	Latitude (WGS84), in degrees * 1E7 (Units: degE7)
longitude	int32_t	Longitude (WGS84, in degrees * 1E7 (Units: degE7)
altitude	int32_t	Altitude (AMSL), in meters * 1000 (positive for up) (Units: mm)
x	float	Local X position of this position in the local coordinate frame (Units: m)
y	float	Local Y position of this position in the local coordinate frame (Units: m)
z	float	Local Z position of this position in the local coordinate frame (Units: m)
q	float[4]	World to surface normal and heading transformation of the takeoff position. Used to indicate the heading and slope of the ground
approach_x	float	Local X position of the end of the approach vector. Multicopters should set this position based on their takeoff path. Grass-landing fixed wing aircraft should set it the same way as multicopters. Runway-landing fixed wing aircraft should set it to the opposite direction of the takeoff, assuming the takeoff happened from the threshold / touchdown zone. (Units: m)
approach_y	float	Local Y position of the end of the approach vector. Multicopters should set this position based on their takeoff path. Grass-landing fixed wing aircraft should set it the same way as multicopters. Runway-landing fixed wing aircraft should set it to the opposite direction of the takeoff, assuming the takeoff happened from the threshold / touchdown zone. (Units: m)
approach_z	float	Local Z position of the end of the approach vector. Multicopters should set this position based on their takeoff path. Grass-landing fixed wing aircraft should set it the same way as multicopters. Runway-landing fixed wing aircraft should set it to the opposite direction of the takeoff, assuming the takeoff happened from the threshold / touchdown zone. (Units: m)
time_usec **	uint64_t	Timestamp (microseconds since UNIX epoch or microseconds since system boot) (Units: us)

SET_HOME_POSITION (#243)

The position the system will return to and land on. The position is set automatically by the system during the takeoff in case it was not explicitely set by the operator before or after. The global and local positions encode the position in the respective coordinate frames, while the q parameter encodes the orientation of the surface. Under normal conditions it describes the heading and terrain slope, which can be used by the aircraft to adjust the approach. The approach 3D vector describes the point to which the system should fly in normal flight mode and then perform a landing sequence along the vector.

Field Name	Type	Description
target_system	uint8_t	System ID.
latitude	int32_t	Latitude (WGS84), in degrees * 1E7 (Units: degE7)
longitude	int32_t	Longitude (WGS84, in degrees * 1E7 (Units: degE7)
altitude	int32_t	Altitude (AMSL), in meters * 1000 (positive for up) (Units: mm)
x	float	Local X position of this position in the local coordinate frame (Units: m)
y	float	Local Y position of this position in the local coordinate frame (Units: m)
z	float	Local Z position of this position in the local coordinate frame (Units: m)
q	float[4]	World to surface normal and heading transformation of the takeoff position. Used to indicate the heading and slope of the ground
approach_x	float	Local X position of the end of the approach vector. Multicopters should set this position based on their takeoff path. Grass-landing fixed wing aircraft should set it the same way as multicopters. Runway-landing fixed wing aircraft should set it to the opposite direction of the takeoff, assuming the takeoff happened from the threshold / touchdown zone. (Units: m)
approach_y	float	Local Y position of the end of the approach vector. Multicopters should set this position based on their takeoff path. Grass-landing fixed wing aircraft should set it the same way as multicopters. Runway-landing fixed wing aircraft should set it to the opposite direction of the takeoff, assuming the takeoff happened from the threshold / touchdown zone. (Units: m)
approach_z	float	Local Z position of the end of the approach vector. Multicopters should set this position based on their takeoff path. Grass-landing fixed wing aircraft should set it the same way as multicopters. Runway-landing fixed wing aircraft should set it to the opposite direction of the takeoff, assuming the takeoff happened from the threshold / touchdown zone. (Units: m)
time_usec **	uint64_t	Timestamp (microseconds since UNIX epoch or microseconds since system boot) (Units: us)

MESSAGE_INTERVAL (#244)

This interface replaces DATA_STREAM

Field Name	Type	Description
message_id	uint16_t	The ID of the requested MAVLink message. v1.0 is limited to 254 messages.
interval_us	int32_t	The interval between two messages, in microseconds. A value of -1 indicates this stream is disabled, 0 indicates it is not available, > 0 indicates the interval at which it is sent. (Units: us)

EXTENDED_SYS_STATE (#245)

Provides state for additional features

Field Name	Type	Description
vtol_state	uint8_t	The VTOL state if applicable. Is set to MAV_VTOL_STATE_UNDEFINED if UAV is not in VTOL configuration. (Enum:MAV_VTOL_STATE)
landed_state	uint8_t	The landed state. Is set to MAV_LANDED_STATE_UNDEFINED if landed state is unknown. (Enum:MAV_LANDED_STATE)

ADSB_VEHICLE (#246)

The location and information of an ADSB vehicle

Field Name	Type	Description
ICAO_address	uint32_t	ICAO address
lat	int32_t	Latitude, expressed as degrees * 1E7 (Units: degE7)
lon	int32_t	Longitude, expressed as degrees * 1E7 (Units: degE7)
altitude_type	uint8_t	Type from ADSB_ALTITUDE_TYPE enum (Enum:ADSB_ALTITUDE_TYPE)
altitude	int32_t	Altitude(ASL) in millimeters (Units: mm)
heading	uint16_t	Course over ground in centidegrees (Units: cdeg)
hor_velocity	uint16_t	The horizontal velocity in centimeters/second (Units: cm/s)
ver_velocity	int16_t	The vertical velocity in centimeters/second, positive is up (Units: cm/s)
callsign	char[9]	The callsign, 8+null
emitter_type	uint8_t	Type from ADSB_EMITTER_TYPE enum (Enum:ADSB_EMITTER_TYPE)
tslc	uint8_t	Time since last communication in seconds (Units: s)
flags	uint16_t	Flags to indicate various statuses including valid data fields (Enum:ADSB_FLAGS)
squawk	uint16_t	Squawk code

COLLISION (#247)

Information about a potential collision

Field Name	Type	Description
src	uint8_t	Collision data source (Enum:MAV_COLLISION_SRC)
id	uint32_t	Unique identifier, domain based on src field
action	uint8_t	Action that is being taken to avoid this collision (Enum:MAV_COLLISION_ACTION)
threat_level	uint8_t	How concerned the aircraft is about this collision (Enum:MAV_COLLISION_THREAT_LEVEL)
time_to_minimum_delta	float	Estimated time until collision occurs (seconds) (Units: s)
altitude_minimum_delta	float	Closest vertical distance in meters between vehicle and object (Units: m)
horizontal_minimum_delta	float	Closest horizontal distance in meteres between vehicle and object (Units: m)

V2_EXTENSION (#248)

Message implementing parts of the V2 payload specs in V1 frames for transitional support.

Field Name	Type	Description
target_network	uint8_t	Network ID (0 for broadcast)
target_system	uint8_t	System ID (0 for broadcast)
target_component	uint8_t	Component ID (0 for broadcast)
message_type	uint16_t	A code that identifies the software component that understands this message (analogous to usb device classes or mime type strings). If this code is less than 32768, it is considered a 'registered' protocol extension and the corresponding entry should be added to https://github.com/mavlink/mavlink/extension-message-ids.xml. Software creators can register blocks of message IDs as needed (useful for GCS specific metadata, etc...). Message_types greater than 32767 are considered local experiments and should not be checked in to any widely distributed codebase.
payload	uint8_t[249]	Variable length payload. The length is defined by the remaining message length when subtracting the header and other fields. The entire content of this block is opaque unless you understand any the encoding message_type. The particular encoding used can be extension specific and might not always be documented as part of the mavlink specification.

MEMORY_VECT (#249)

Send raw controller memory. The use of this message is discouraged for normal packets, but a quite efficient way for testing new messages and getting experimental debug output.

Field Name	Type	Description
address	uint16_t	Starting address of the debug variables
ver	uint8_t	Version code of the type variable. 0=unknown, type ignored and assumed int16_t. 1=as below
type	uint8_t	Type code of the memory variables. for ver = 1: 0=16 x int16_t, 1=16 x uint16_t, 2=16 x Q15, 3=16 x 1Q14
value	int8_t[32]	Memory contents at specified address

DEBUG_VECT (#250)

Field Name	Type	Description
name	char[10]	Name
time_usec	uint64_t	Timestamp (Units: us)
x	float	x
y	float	y
z	float	z

NAMED_VALUE_FLOAT (#251)

Send a key-value pair as float. The use of this message is discouraged for normal packets, but a quite efficient way for testing new messages and getting experimental debug output.

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
name	char[10]	Name of the debug variable
value	float	Floating point value

NAMED_VALUE_INT (#252)

Send a key-value pair as integer. The use of this message is discouraged for normal packets, but a quite efficient way for testing new messages and getting experimental debug output.

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
name	char[10]	Name of the debug variable
value	int32_t	Signed integer value

STATUSTEXT (#253)

Status text message. These messages are printed in yellow in the COMM console of QGroundControl. WARNING: They consume quite some bandwidth, so use only for important status and error messages. If implemented wisely, these messages are buffered on the MCU and sent only at a limited rate (e.g. 10 Hz).

Field Name	Type	Description
severity	uint8_t	Severity of status. Relies on the definitions within RFC-5424. See enum MAV_SEVERITY. (Enum:MAV_SEVERITY)
text	char[50]	Status text message, without null termination character

DEBUG (#254)

Send a debug value. The index is used to discriminate between values. These values show up in the plot of QGroundControl as DEBUG N.

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
ind	uint8_t	index of debug variable
value	float	DEBUG value

SETUP_SIGNING (#256)

(MAVLink 2) Setup a MAVLink2 signing key. If called with secret_key of all zero and zero initial_timestamp will disable signing

Field Name	Type	Description
target_system	uint8_t	system id of the target
target_component	uint8_t	component ID of the target
secret_key	uint8_t[32]	signing key
initial_timestamp	uint64_t	initial timestamp

BUTTON_CHANGE (#257)

(MAVLink 2) Report button state change

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
last_change_ms	uint32_t	Time of last change of button state (Units: ms)
state	uint8_t	Bitmap state of buttons

PLAY_TUNE (#258)

(MAVLink 2) Control vehicle tone generation (buzzer)

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
tune	char[30]	tune in board specific format

CAMERA_INFORMATION (#259)

(MAVLink 2) Information about a camera

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
vendor_name	uint8_t[32]	Name of the camera vendor
model_name	uint8_t[32]	Name of the camera model
firmware_version	uint32_t	Version of the camera firmware (v << 24 & 0xff = Dev, v << 16 & 0xff = Patch, v << 8 & 0xff = Minor, v & 0xff = Major)
focal_length	float	Focal length in mm (Units: mm)
sensor_size_h	float	Image sensor size horizontal in mm (Units: mm)
sensor_size_v	float	Image sensor size vertical in mm (Units: mm)
resolution_h	uint16_t	Image resolution in pixels horizontal (Units: pix)
resolution_v	uint16_t	Image resolution in pixels vertical (Units: pix)
lens_id	uint8_t	Reserved for a lens ID
flags	uint32_t	CAMERA_CAP_FLAGS enum flags (bitmap) describing camera capabilities. (Enum:CAMERA_CAP_FLAGS)
cam_definition_version	uint16_t	Camera definition version (iteration)
cam_definition_uri	char[140]	Camera definition URI (if any, otherwise only basic functions will be available).

CAMERA_SETTINGS (#260)

(MAVLink 2) Settings of a camera, can be requested using MAV_CMD_REQUEST_CAMERA_SETTINGS.

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
mode_id	uint8_t	Camera mode (CAMERA_MODE) (Enum:CAMERA_MODE)

STORAGE_INFORMATION (#261)

(MAVLink 2) WIP: Information about a storage medium.

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
storage_id	uint8_t	Storage ID (1 for first, 2 for second, etc.)
storage_count	uint8_t	Number of storage devices
status	uint8_t	Status of storage (0 not available, 1 unformatted, 2 formatted)
total_capacity	float	Total capacity in MiB (Units: Mibytes)
used_capacity	float	Used capacity in MiB (Units: Mibytes)
available_capacity	float	Available capacity in MiB (Units: Mibytes)
read_speed	float	Read speed in MiB/s (Units: Mibytes/s)
write_speed	float	Write speed in MiB/s (Units: Mibytes/s)

CAMERA_CAPTURE_STATUS (#262)

(MAVLink 2) Information about the status of a capture

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
image_status	uint8_t	Current status of image capturing (0: idle, 1: capture in progress, 2: interval set but idle, 3: interval set and capture in progress)
video_status	uint8_t	Current status of video capturing (0: idle, 1: capture in progress)
image_interval	float	Image capture interval in seconds (Units: s)
recording_time_ms	uint32_t	Time in milliseconds since recording started (Units: ms)
available_capacity	float	Available storage capacity in MiB (Units: Mibytes)

CAMERA_IMAGE_CAPTURED (#263)

(MAVLink 2) Information about a captured image

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
time_utc	uint64_t	Timestamp (microseconds since UNIX epoch) in UTC. 0 for unknown. (Units: us)
camera_id	uint8_t	Camera ID (1 for first, 2 for second, etc.)
lat	int32_t	Latitude, expressed as degrees * 1E7 where image was taken (Units: degE7)
lon	int32_t	Longitude, expressed as degrees * 1E7 where capture was taken (Units: degE7)
alt	int32_t	Altitude in meters, expressed as * 1E3 (AMSL, not WGS84) where image was taken (Units: m)
relative_alt	int32_t	Altitude above ground in meters, expressed as * 1E3 where image was taken (Units: m)
q	float[4]	Quaternion of camera orientation (w, x, y, z order, zero-rotation is 0, 0, 0, 0)
image_index	int32_t	Zero based index of this image (image count since armed -1)
capture_result	int8_t	Boolean indicating success (1) or failure (0) while capturing this image.
file_url	char[205]	URL of image taken. Either local storage or http://foo.jpg if camera provides an HTTP interface.

FLIGHT_INFORMATION (#264)

(MAVLink 2) WIP: Information about flight since last arming

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
arming_time_utc	uint64_t	Timestamp at arming (microseconds since UNIX epoch) in UTC, 0 for unknown (Units: us)
takeoff_time_utc	uint64_t	Timestamp at takeoff (microseconds since UNIX epoch) in UTC, 0 for unknown (Units: us)
flight_uuid	uint64_t	Universally unique identifier (UUID) of flight, should correspond to name of logfiles

MOUNT_ORIENTATION (#265)

(MAVLink 2) Orientation of a mount

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot) (Units: ms)
roll	float	Roll in global frame in degrees (set to NaN for invalid). (Units: deg)
pitch	float	Pitch in global frame in degrees (set to NaN for invalid). (Units: deg)
yaw	float	Yaw relative to vehicle in degrees (set to NaN for invalid). (Units: deg)
yaw_absolute **	float	Yaw in absolute frame in degrees, North is 0 (set to NaN for invalid). (Units: deg)

LOGGING_DATA (#266)

(MAVLink 2) A message containing logged data (see also MAV_CMD_LOGGING_START)

Field Name	Type	Description
target_system	uint8_t	system ID of the target
target_component	uint8_t	component ID of the target
sequence	uint16_t	sequence number (can wrap)
length	uint8_t	data length (Units: bytes)
first_message_offset	uint8_t	offset into data where first message starts. This can be used for recovery, when a previous message got lost (set to 255 if no start exists). (Units: bytes)
data	uint8_t[249]	logged data

LOGGING_DATA_ACKED (#267)

(MAVLink 2) A message containing logged data which requires a LOGGING_ACK to be sent back

Field Name	Type	Description
target_system	uint8_t	system ID of the target
target_component	uint8_t	component ID of the target
sequence	uint16_t	sequence number (can wrap)
length	uint8_t	data length (Units: bytes)
first_message_offset	uint8_t	offset into data where first message starts. This can be used for recovery, when a previous message got lost (set to 255 if no start exists). (Units: bytes)
data	uint8_t[249]	logged data

LOGGING_ACK (#268)

(MAVLink 2) An ack for a LOGGING_DATA_ACKED message

Field Name	Type	Description
target_system	uint8_t	system ID of the target
target_component	uint8_t	component ID of the target
sequence	uint16_t	sequence number (must match the one in LOGGING_DATA_ACKED)

VIDEO_STREAM_INFORMATION (#269)

(MAVLink 2) WIP: Information about video stream

Field Name	Type	Description
camera_id	uint8_t	Camera ID (1 for first, 2 for second, etc.)
status	uint8_t	Current status of video streaming (0: not running, 1: in progress)
framerate	float	Frames per second (Units: Hz)
resolution_h	uint16_t	Resolution horizontal in pixels (Units: pix)
resolution_v	uint16_t	Resolution vertical in pixels (Units: pix)
bitrate	uint32_t	Bit rate in bits per second (Units: bits/s)
rotation	uint16_t	Video image rotation clockwise (Units: deg)
uri	char[230]	Video stream URI

SET_VIDEO_STREAM_SETTINGS (#270)

(MAVLink 2) WIP: Message that sets video stream settings

Field Name	Type	Description
target_system	uint8_t	system ID of the target
target_component	uint8_t	component ID of the target
camera_id	uint8_t	Camera ID (1 for first, 2 for second, etc.)
framerate	float	Frames per second (set to -1 for highest framerate possible) (Units: Hz)
resolution_h	uint16_t	Resolution horizontal in pixels (set to -1 for highest resolution possible) (Units: pix)
resolution_v	uint16_t	Resolution vertical in pixels (set to -1 for highest resolution possible) (Units: pix)
bitrate	uint32_t	Bit rate in bits per second (set to -1 for auto) (Units: bits/s)
rotation	uint16_t	Video image rotation clockwise (0-359 degrees) (Units: deg)
uri	char[230]	Video stream URI

WIFI_CONFIG_AP (#299)

(MAVLink 2) Configure AP SSID and Password.

Field Name	Type	Description
ssid	char[32]	Name of Wi-Fi network (SSID). Leave it blank to leave it unchanged.
password	char[64]	Password. Leave it blank for an open AP.

PROTOCOL_VERSION (#300)

(MAVLink 2) WIP: Version and capability of protocol version. This message is the response to REQUEST_PROTOCOL_VERSION and is used as part of the handshaking to establish which MAVLink version should be used on the network. Every node should respond to REQUEST_PROTOCOL_VERSION to enable the handshaking. Library implementers should consider adding this into the default decoding state machine to allow the protocol core to respond directly.

Field Name	Type	Description
version	uint16_t	Currently active MAVLink version number * 100: v1.0 is 100, v2.0 is 200, etc.
min_version	uint16_t	Minimum MAVLink version supported
max_version	uint16_t	Maximum MAVLink version supported (set to the same value as version by default)
spec_version_hash	uint8_t[8]	The first 8 bytes (not characters printed in hex!) of the git hash.
library_version_hash	uint8_t[8]	The first 8 bytes (not characters printed in hex!) of the git hash.

UAVCAN_NODE_STATUS (#310)

(MAVLink 2) General status information of an UAVCAN node. Please refer to the definition of the UAVCAN message "uavcan.protocol.NodeStatus" for the background information. The UAVCAN specification is available at http://uavcan.org.

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds since UNIX epoch or microseconds since system boot) (Units: us)
uptime_sec	uint32_t	The number of seconds since the start-up of the node. (Units: s)
health	uint8_t	Generalized node health status. (Enum:UAVCAN_NODE_HEALTH)
mode	uint8_t	Generalized operating mode. (Enum:UAVCAN_NODE_MODE)
sub_mode	uint8_t	Not used currently.
vendor_specific_status_code	uint16_t	Vendor-specific status information.

UAVCAN_NODE_INFO (#311)

(MAVLink 2) General information describing a particular UAVCAN node. Please refer to the definition of the UAVCAN service "uavcan.protocol.GetNodeInfo" for the background information. This message should be emitted by the system whenever a new node appears online, or an existing node reboots. Additionally, it can be emitted upon request from the other end of the MAVLink channel (see MAV_CMD_UAVCAN_GET_NODE_INFO). It is also not prohibited to emit this message unconditionally at a low frequency. The UAVCAN specification is available at http://uavcan.org.

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds since UNIX epoch or microseconds since system boot) (Units: us)
uptime_sec	uint32_t	The number of seconds since the start-up of the node. (Units: s)
name	char[80]	Node name string. For example, "sapog.px4.io".
hw_version_major	uint8_t	Hardware major version number.
hw_version_minor	uint8_t	Hardware minor version number.
hw_unique_id	uint8_t[16]	Hardware unique 128-bit ID.
sw_version_major	uint8_t	Software major version number.
sw_version_minor	uint8_t	Software minor version number.
sw_vcs_commit	uint32_t	Version control system (VCS) revision identifier (e.g. git short commit hash). Zero if unknown.

PARAM_EXT_REQUEST_READ (#320)

(MAVLink 2) Request to read the value of a parameter with the either the param_id string id or param_index.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
param_id	char[16]	Parameter id, terminated by NULL if the length is less than 16 human-readable chars and WITHOUT null termination (NULL) byte if the length is exactly 16 chars - applications have to provide 16+1 bytes storage if the ID is stored as string
param_index	int16_t	Parameter index. Set to -1 to use the Parameter ID field as identifier (else param_id will be ignored)

PARAM_EXT_REQUEST_LIST (#321)

(MAVLink 2) Request all parameters of this component. After this request, all parameters are emitted.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID

PARAM_EXT_VALUE (#322)

(MAVLink 2) Emit the value of a parameter. The inclusion of param_count and param_index in the message allows the recipient to keep track of received parameters and allows them to re-request missing parameters after a loss or timeout.

Field Name	Type	Description
param_id	char[16]	Parameter id, terminated by NULL if the length is less than 16 human-readable chars and WITHOUT null termination (NULL) byte if the length is exactly 16 chars - applications have to provide 16+1 bytes storage if the ID is stored as string
param_value	char[128]	Parameter value
param_type	uint8_t	Parameter type: see the MAV_PARAM_EXT_TYPE enum for supported data types. (Enum:MAV_PARAM_EXT_TYPE)
param_count	uint16_t	Total number of parameters
param_index	uint16_t	Index of this parameter

PARAM_EXT_SET (#323)

(MAVLink 2) Set a parameter value. In order to deal with message loss (and retransmission of PARAM_EXT_SET), when setting a parameter value and the new value is the same as the current value, you will immediately get a PARAM_ACK_ACCEPTED response. If the current state is PARAM_ACK_IN_PROGRESS, you will accordingly receive a PARAM_ACK_IN_PROGRESS in response.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
param_id	char[16]	Parameter id, terminated by NULL if the length is less than 16 human-readable chars and WITHOUT null termination (NULL) byte if the length is exactly 16 chars - applications have to provide 16+1 bytes storage if the ID is stored as string
param_value	char[128]	Parameter value
param_type	uint8_t	Parameter type: see the MAV_PARAM_EXT_TYPE enum for supported data types. (Enum:MAV_PARAM_EXT_TYPE)

PARAM_EXT_ACK (#324)

(MAVLink 2) Response from a PARAM_EXT_SET message.

Field Name	Type	Description
param_id	char[16]	Parameter id, terminated by NULL if the length is less than 16 human-readable chars and WITHOUT null termination (NULL) byte if the length is exactly 16 chars - applications have to provide 16+1 bytes storage if the ID is stored as string
param_value	char[128]	Parameter value (new value if PARAM_ACK_ACCEPTED, current value otherwise)
param_type	uint8_t	Parameter type: see the MAV_PARAM_EXT_TYPE enum for supported data types. (Enum:MAV_PARAM_EXT_TYPE)
param_result	uint8_t	Result code: see the PARAM_ACK enum for possible codes. (Enum:PARAM_ACK)

OBSTACLE_DISTANCE (#330)

(MAVLink 2) Obstacle distances in front of the sensor, starting from the left in increment degrees to the right

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds since system boot or since UNIX epoch). (Units: us)
sensor_type	uint8_t	Class id of the distance sensor type. (Enum:MAV_DISTANCE_SENSOR)
distances	uint16_t[72]	Distance of obstacles around the UAV with index 0 corresponding to local North. A value of 0 means that the obstacle is right in front of the sensor. A value of max_distance +1 means no obstacle is present. A value of UINT16_MAX for unknown/not used. In a array element, one unit corresponds to 1cm. (Units: cm)
increment	uint8_t	Angular width in degrees of each array element. (Units: deg)
min_distance	uint16_t	Minimum distance the sensor can measure in centimeters. (Units: cm)
max_distance	uint16_t	Maximum distance the sensor can measure in centimeters. (Units: cm)

ODOMETRY (#331)

(MAVLink 2) Odometry message to communicate odometry information with an external interface. Fits ROS REP 147 standard for aerial vehicles (http://www.ros.org/reps/rep-0147.html).

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds since system boot or since UNIX epoch). (Units: us)
frame_id	uint8_t	Coordinate frame of reference for the pose data, as defined by MAV_FRAME enum. (Enum:MAV_FRAME)
child_frame_id	uint8_t	Coordinate frame of reference for the velocity in free space (twist) data, as defined by MAV_FRAME enum. (Enum:MAV_FRAME)
x	float	X Position (Units: m)
y	float	Y Position (Units: m)
z	float	Z Position (Units: m)
q	float[4]	Quaternion components, w, x, y, z (1 0 0 0 is the null-rotation)
vx	float	X linear speed (Units: m/s)
vy	float	Y linear speed (Units: m/s)
vz	float	Z linear speed (Units: m/s)
rollspeed	float	Roll angular speed (Units: rad/s)
pitchspeed	float	Pitch angular speed (Units: rad/s)
yawspeed	float	Yaw angular speed (Units: rad/s)
pose_covariance	float[21]	Pose (states: x, y, z, roll, pitch, yaw) covariance matrix upper right triangle (first six entries are the first ROW, next five entries are the second ROW, etc.)
twist_covariance	float[21]	Twist (states: vx, vy, vz, rollspeed, pitchspeed, yawspeed) covariance matrix upper right triangle (first six entries are the first ROW, next five entries are the second ROW, etc.)

MAVLINK ArduPilotMega Message Set

These messages define the APM specific message set, which is custom to http://ardupilot.org.

This is a human-readable form of the XML definition file: ardupilotmega.xml.

> The ArduPilot MAVLink fork of ardupilotmega.xml may contain messages that have not yet been merged into this documentation.

> MAVLink 2 messages have an ID > 255 and are marked up using **(MAVLink 2)** in their description.

> MAVLink 2 extension fields that have been added to MAVLink 1 messages are displayed in blue.

MAVLink Include Files: common.xml

MAVLink Include Files: uAvionix.xml

MAVLink Include Files: icarous.xml

This file has protocol dialect: 2.

MAVLink Type Enumerations

ACCELCAL_VEHICLE_POS

CMD ID	Field Name	Description
1	ACCELCAL_VEHICLE_POS_LEVEL	
2	ACCELCAL_VEHICLE_POS_LEFT	
3	ACCELCAL_VEHICLE_POS_RIGHT	
4	ACCELCAL_VEHICLE_POS_NOSEDOWN	
5	ACCELCAL_VEHICLE_POS_NOSEUP	
6	ACCELCAL_VEHICLE_POS_BACK	
16777215	ACCELCAL_VEHICLE_POS_SUCCESS	
16777216	ACCELCAL_VEHICLE_POS_FAILED	

MAV_CMD

CMD ID	Field Name	Description
211	MAV_CMD_DO_GRIPPER	Mission command to operate EPM gripper
	Mission Param #1	gripper number (a number from 1 to max number of grippers on the vehicle)
	Mission Param #2	gripper action (0=release, 1=grab. See GRIPPER_ACTIONS enum)
	Mission Param #3	Empty
	Mission Param #4	Empty

	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
212	MAV_CMD_DO_AUTOTUNE_ENABLE	Enable/disable autotune
	Mission Param #1	enable (1: enable, 0:disable)
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
83	MAV_CMD_NAV_ALTITUDE_WAIT	Mission command to wait for an altitude or downwards vertical speed. This is meant for high altitude balloon launches, allowing the aircraft to be idle until either an altitude is reached or a negative vertical speed is reached (indicating early balloon burst). The wiggle time is how often to wiggle the control surfaces to prevent them seizing up.
	Mission Param #1	altitude (m)
	Mission Param #2	descent speed (m/s)
	Mission Param #3	Wiggle Time (s)
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
42000	MAV_CMD_POWER_OFF_INITIATED	A system wide power-off event has been initiated.
	Mission Param #1	Empty
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
42001	MAV_CMD_SOLO_BTN_FLY_CLICK	FLY button has been clicked.

	Mission Param #1	Empty
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
42002	MAV_CMD_SOLO_BTN_FLY_HOLD	FLY button has been held for 1.5 seconds.
	Mission Param #1	Takeoff altitude
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
42003	MAV_CMD_SOLO_BTN_PAUSE_CLICK	PAUSE button has been clicked.
	Mission Param #1	1 if Solo is in a shot mode, 0 otherwise
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
42004	MAV_CMD_FIXED_MAG_CAL	Magnetometer calibration based on fixed position in earth field given by inclination, declination and intensity
	Mission Param #1	MagDeclinationDegrees
	Mission Param #2	MagInclinationDegrees
	Mission Param #3	MagIntensityMilliGauss
	Mission Param #4	YawDegrees
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty

42005	MAV_CMD_FIXED_MAG_CAL_FIELD	Magnetometer calibration based on fixed expected field values in milliGauss
	Mission Param #1	FieldX
	Mission Param #2	FieldY
	Mission Param #3	FieldZ
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
42424	MAV_CMD_DO_START_MAG_CAL	Initiate a magnetometer calibration
	Mission Param #1	uint8_t bitmask of magnetometers (0 means all)
	Mission Param #2	Automatically retry on failure (0=no retry, 1=retry).
	Mission Param #3	Save without user input (0=require input, 1=autosave).
	Mission Param #4	Delay (seconds)
	Mission Param #5	Autoreboot (0=user reboot, 1=autoreboot)
	Mission Param #6	Empty
	Mission Param #7	Empty
42425	MAV_CMD_DO_ACCEPT_MAG_CAL	Initiate a magnetometer calibration
	Mission Param #1	uint8_t bitmask of magnetometers (0 means all)
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
42426	MAV_CMD_DO_CANCEL_MAG_CAL	Cancel a running magnetometer calibration
	Mission Param #1	uint8_t bitmask of magnetometers (0 means all)
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty

	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
42429	MAV_CMD_ACCELCAL_VEHICLE_POS	Used when doing accelerometer calibration. When sent to the GCS tells it what position to put the vehicle in. When sent to the vehicle says what position the vehicle is in.
	Mission Param #1	Position, one of the ACCELCAL_VEHICLE_POS enum values
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
42428	MAV_CMD_DO_SEND_BANNER	Reply with the version banner
	Mission Param #1	Empty
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
42427	MAV_CMD_SET_FACTORY_TEST_MODE	Command autopilot to get into factory test/diagnostic mode
	Mission Param #1	0 means get out of test mode, 1 means get into test mode
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
42501	MAV_CMD_GIMBAL_RESET	Causes the gimbal to reset and boot as if it was just powered on

	Mission Param #1	Empty
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
42502	MAV_CMD_GIMBAL_AXIS_CALIBRATION_STATUS	Reports progress and success or failure of gimbal axis calibration procedure
	Mission Param #1	Gimbal axis we're reporting calibration progress for
	Mission Param #2	Current calibration progress for this axis, 0x64=100%
	Mission Param #3	Status of the calibration
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
42503	MAV_CMD_GIMBAL_REQUEST_AXIS_CALIBRATION	Starts commutation calibration on the gimbal
	Mission Param #1	Empty
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
42505	MAV_CMD_GIMBAL_FULL_RESET	Erases gimbal application and parameters
	Mission Param #1	Magic number
	Mission Param #2	Magic number
	Mission Param #3	Magic number
	Mission Param #4	Magic number
	Mission Param #5	Magic number
	Mission Param #6	Magic number
	Mission Param #7	Magic number

42600	MAV_CMD_DO_WINCH	Command to operate winch
	Mission Param #1	winch number (0 for the default winch, otherwise a number from 1 to max number of winches on the vehicle)
	Mission Param #2	action (0=relax, 1=relative length control, 2=rate control. See WINCH_ACTIONS enum)
	Mission Param #3	release length (cable distance to unwind in meters, negative numbers to wind in cable)
	Mission Param #4	release rate (meters/second)
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty

LIMITS_STATE

CMD ID	Field Name	Description
0	LIMITS_INIT	pre-initialization
1	LIMITS_DISABLED	disabled
2	LIMITS_ENABLED	checking limits
3	LIMITS_TRIGGERED	a limit has been breached
4	LIMITS_RECOVERING	taking action eg. RTL
5	LIMITS_RECOVERED	we're no longer in breach of a limit

LIMIT_MODULE

CMD ID	Field Name	Description
1	LIMIT_GPSLOCK	pre-initialization
2	LIMIT_GEOFENCE	disabled
4	LIMIT_ALTITUDE	checking limits

RALLY_FLAGS

Flags in RALLY_POINT message

CMD ID	Field Name	Description
1	FAVORABLE_WIND	Flag set when requiring favorable winds for landing.
2	LAND_IMMEDIATELY	Flag set when plane is to immediately descend to break altitude and land without GCS intervention. Flag not set when plane is to loiter at Rally point until commanded to land.

PARACHUTE_ACTION

CMD ID	Field Name	Description
0	PARACHUTE_DISABLE	Disable parachute release
1	PARACHUTE_ENABLE	Enable parachute release
2	PARACHUTE_RELEASE	Release parachute

GRIPPER_ACTIONS

Gripper actions.

CMD ID	Field Name	Description
0	GRIPPER_ACTION_RELEASE	gripper release of cargo
1	GRIPPER_ACTION_GRAB	gripper grabs onto cargo

WINCH_ACTIONS

Winch actions

CMD ID	Field Name	Description
0	WINCH_RELAXED	relax winch
1	WINCH_RELATIVE_LENGTH_CONTROL	winch unwinds or winds specified length of cable optionally using specified rate
2	WINCH_RATE_CONTROL	winch unwinds or winds cable at specified rate in meters/seconds

CAMERA_STATUS_TYPES

CMD ID	Field Name	Description
0	CAMERA_STATUS_TYPE_HEARTBEAT	Camera heartbeat, announce camera component ID at 1hz
1	CAMERA_STATUS_TYPE_TRIGGER	Camera image triggered
2	CAMERA_STATUS_TYPE_DISCONNECT	Camera connection lost
3	CAMERA_STATUS_TYPE_ERROR	Camera unknown error
4	CAMERA_STATUS_TYPE_LOWBATT	Camera battery low. Parameter p1 shows reported voltage
5	CAMERA_STATUS_TYPE_LOWSTORE	Camera storage low. Parameter p1 shows reported shots remaining
6	CAMERA_STATUS_TYPE_LOWSTOREV	Camera storage low. Parameter p1 shows reported video minutes remaining

CAMERA_FEEDBACK_FLAGS

CMD ID	Field Name	Description
0	CAMERA_FEEDBACK_PHOTO	Shooting photos, not video
1	CAMERA_FEEDBACK_VIDEO	Shooting video, not stills
2	CAMERA_FEEDBACK_BADEXPOSURE	Unable to achieve requested exposure (e.g. shutter speed too low)
3	CAMERA_FEEDBACK_CLOSEDLOOP	Closed loop feedback from camera, we know for sure it has successfully taken a picture
4	CAMERA_FEEDBACK_OPENLOOP	Open loop camera, an image trigger has been requested but we can't know for sure it has successfully taken a picture

MAV_MODE_GIMBAL

CMD ID	Field Name	Description
0	MAV_MODE_GIMBAL_UNINITIALIZED	Gimbal is powered on but has not started initializing yet
1	MAV_MODE_GIMBAL_CALIBRATING_PITCH	Gimbal is currently running calibration on the pitch axis
2	MAV_MODE_GIMBAL_CALIBRATING_ROLL	Gimbal is currently running calibration on the roll axis
3	MAV_MODE_GIMBAL_CALIBRATING_YAW	Gimbal is currently running calibration on the yaw axis
4	MAV_MODE_GIMBAL_INITIALIZED	Gimbal has finished calibrating and initializing, but is relaxed pending reception of first rate command from copter
5	MAV_MODE_GIMBAL_ACTIVE	Gimbal is actively stabilizing
6	MAV_MODE_GIMBAL_RATE_CMD_TIMEOUT	Gimbal is relaxed because it missed more than 10 expected rate command messages in a row. Gimbal will move back to active mode when it receives a new rate command

GIMBAL_AXIS

CMD ID	Field Name	Description
0	GIMBAL_AXIS_YAW	Gimbal yaw axis
1	GIMBAL_AXIS_PITCH	Gimbal pitch axis
2	GIMBAL_AXIS_ROLL	Gimbal roll axis

GIMBAL_AXIS_CALIBRATION_STATUS

CMD ID	Field Name	Description
0	GIMBAL_AXIS_CALIBRATION_STATUS_IN_PROGRESS	Axis calibration is in progress
1	GIMBAL_AXIS_CALIBRATION_STATUS_SUCCEEDED	Axis calibration succeeded
2	GIMBAL_AXIS_CALIBRATION_STATUS_FAILED	Axis calibration failed

GIMBAL_AXIS_CALIBRATION_REQUIRED

CMD ID	Field Name	Description
0	GIMBAL_AXIS_CALIBRATION_REQUIRED_UNKNOWN	Whether or not this axis requires calibration is unknown at this time
1	GIMBAL_AXIS_CALIBRATION_REQUIRED_TRUE	This axis requires calibration
2	GIMBAL_AXIS_CALIBRATION_REQUIRED_FALSE	This axis does not require calibration

GOPRO_HEARTBEAT_STATUS

CMD ID	Field Name	Description
0	GOPRO_HEARTBEAT_STATUS_DISCONNECTED	No GoPro connected
1	GOPRO_HEARTBEAT_STATUS_INCOMPATIBLE	The detected GoPro is not HeroBus compatible
2	GOPRO_HEARTBEAT_STATUS_CONNECTED	A HeroBus compatible GoPro is connected
3	GOPRO_HEARTBEAT_STATUS_ERROR	An unrecoverable error was encountered with the connected GoPro, it may require a power cycle

GOPRO_HEARTBEAT_FLAGS

CMD ID	Field Name	Description
1	GOPRO_FLAG_RECORDING	GoPro is currently recording

GOPRO_REQUEST_STATUS

CMD ID	Field Name	Description
0	GOPRO_REQUEST_SUCCESS	The write message with ID indicated succeeded
1	GOPRO_REQUEST_FAILED	The write message with ID indicated failed

GOPRO_COMMAND

CMD ID	Field Name	Description
0	GOPRO_COMMAND_POWER	(Get/Set)
1	GOPRO_COMMAND_CAPTURE_MODE	(Get/Set)
2	GOPRO_COMMAND_SHUTTER	(___/Set)
3	GOPRO_COMMAND_BATTERY	(Get/___)
4	GOPRO_COMMAND_MODEL	(Get/___)
5	GOPRO_COMMAND_VIDEO_SETTINGS	(Get/Set)
6	GOPRO_COMMAND_LOW_LIGHT	(Get/Set)
7	GOPRO_COMMAND_PHOTO_RESOLUTION	(Get/Set)
8	GOPRO_COMMAND_PHOTO_BURST_RATE	(Get/Set)
9	GOPRO_COMMAND_PROTUNE	(Get/Set)
10	GOPRO_COMMAND_PROTUNE_WHITE_BALANCE	(Get/Set) Hero 3+ Only
11	GOPRO_COMMAND_PROTUNE_COLOUR	(Get/Set) Hero 3+ Only
12	GOPRO_COMMAND_PROTUNE_GAIN	(Get/Set) Hero 3+ Only
13	GOPRO_COMMAND_PROTUNE_SHARPNESS	(Get/Set) Hero 3+ Only
14	GOPRO_COMMAND_PROTUNE_EXPOSURE	(Get/Set) Hero 3+ Only
15	GOPRO_COMMAND_TIME	(Get/Set)
16	GOPRO_COMMAND_CHARGING	(Get/Set)

GOPRO_CAPTURE_MODE

CMD ID	Field Name	Description
0	GOPRO_CAPTURE_MODE_VIDEO	Video mode
1	GOPRO_CAPTURE_MODE_PHOTO	Photo mode
2	GOPRO_CAPTURE_MODE_BURST	Burst mode, hero 3+ only
3	GOPRO_CAPTURE_MODE_TIME_LAPSE	Time lapse mode, hero 3+ only
4	GOPRO_CAPTURE_MODE_MULTI_SHOT	Multi shot mode, hero 4 only
5	GOPRO_CAPTURE_MODE_PLAYBACK	Playback mode, hero 4 only, silver only except when LCD or HDMI is connected to black
6	GOPRO_CAPTURE_MODE_SETUP	Playback mode, hero 4 only
255	GOPRO_CAPTURE_MODE_UNKNOWN	Mode not yet known

GOPRO_RESOLUTION

CMD ID	Field Name	Description
0	GOPRO_RESOLUTION_480p	848 x 480 (480p)
1	GOPRO_RESOLUTION_720p	1280 x 720 (720p)
2	GOPRO_RESOLUTION_960p	1280 x 960 (960p)
3	GOPRO_RESOLUTION_1080p	1920 x 1080 (1080p)
4	GOPRO_RESOLUTION_1440p	1920 x 1440 (1440p)
5	GOPRO_RESOLUTION_2_7k_17_9	2704 x 1440 (2.7k-17:9)
6	GOPRO_RESOLUTION_2_7k_16_9	2704 x 1524 (2.7k-16:9)
7	GOPRO_RESOLUTION_2_7k_4_3	2704 x 2028 (2.7k-4:3)
8	GOPRO_RESOLUTION_4k_16_9	3840 x 2160 (4k-16:9)
9	GOPRO_RESOLUTION_4k_17_9	4096 x 2160 (4k-17:9)
10	GOPRO_RESOLUTION_720p_SUPERVIEW	1280 x 720 (720p-SuperView)
11	GOPRO_RESOLUTION_1080p_SUPERVIEW	1920 x 1080 (1080p-SuperView)
12	GOPRO_RESOLUTION_2_7k_SUPERVIEW	2704 x 1520 (2.7k-SuperView)
13	GOPRO_RESOLUTION_4k_SUPERVIEW	3840 x 2160 (4k-SuperView)

GOPRO_FRAME_RATE

CMD ID	Field Name	Description
0	GOPRO_FRAME_RATE_12	12 FPS
1	GOPRO_FRAME_RATE_15	15 FPS
2	GOPRO_FRAME_RATE_24	24 FPS
3	GOPRO_FRAME_RATE_25	25 FPS
4	GOPRO_FRAME_RATE_30	30 FPS
5	GOPRO_FRAME_RATE_48	48 FPS
6	GOPRO_FRAME_RATE_50	50 FPS
7	GOPRO_FRAME_RATE_60	60 FPS
8	GOPRO_FRAME_RATE_80	80 FPS
9	GOPRO_FRAME_RATE_90	90 FPS
10	GOPRO_FRAME_RATE_100	100 FPS
11	GOPRO_FRAME_RATE_120	120 FPS
12	GOPRO_FRAME_RATE_240	240 FPS
13	GOPRO_FRAME_RATE_12_5	12.5 FPS

GOPRO_FIELD_OF_VIEW

CMD ID	Field Name	Description
0	GOPRO_FIELD_OF_VIEW_WIDE	0x00: Wide
1	GOPRO_FIELD_OF_VIEW_MEDIUM	0x01: Medium
2	GOPRO_FIELD_OF_VIEW_NARROW	0x02: Narrow

GOPRO_VIDEO_SETTINGS_FLAGS

CMD ID	Field Name	Description
1	GOPRO_VIDEO_SETTINGS_TV_MODE	0=NTSC, 1=PAL

GOPRO_PHOTO_RESOLUTION

CMD ID	Field Name	Description
0	GOPRO_PHOTO_RESOLUTION_5MP_MEDIUM	5MP Medium
1	GOPRO_PHOTO_RESOLUTION_7MP_MEDIUM	7MP Medium
2	GOPRO_PHOTO_RESOLUTION_7MP_WIDE	7MP Wide
3	GOPRO_PHOTO_RESOLUTION_10MP_WIDE	10MP Wide
4	GOPRO_PHOTO_RESOLUTION_12MP_WIDE	12MP Wide

GOPRO_PROTUNE_WHITE_BALANCE

CMD ID	Field Name	Description
0	GOPRO_PROTUNE_WHITE_BALANCE_AUTO	Auto
1	GOPRO_PROTUNE_WHITE_BALANCE_3000K	3000K
2	GOPRO_PROTUNE_WHITE_BALANCE_5500K	5500K
3	GOPRO_PROTUNE_WHITE_BALANCE_6500K	6500K
4	GOPRO_PROTUNE_WHITE_BALANCE_RAW	Camera Raw

GOPRO_PROTUNE_COLOUR

CMD ID	Field Name	Description
0	GOPRO_PROTUNE_COLOUR_STANDARD	Auto
1	GOPRO_PROTUNE_COLOUR_NEUTRAL	Neutral

GOPRO_PROTUNE_GAIN

CMD ID	Field Name	Description
0	GOPRO_PROTUNE_GAIN_400	ISO 400
1	GOPRO_PROTUNE_GAIN_800	ISO 800 (Only Hero 4)
2	GOPRO_PROTUNE_GAIN_1600	ISO 1600
3	GOPRO_PROTUNE_GAIN_3200	ISO 3200 (Only Hero 4)
4	GOPRO_PROTUNE_GAIN_6400	ISO 6400

GOPRO_PROTUNE_SHARPNESS

CMD ID	Field Name	Description
0	GOPRO_PROTUNE_SHARPNESS_LOW	Low Sharpness
1	GOPRO_PROTUNE_SHARPNESS_MEDIUM	Medium Sharpness
2	GOPRO_PROTUNE_SHARPNESS_HIGH	High Sharpness

GOPRO_PROTUNE_EXPOSURE

CMD ID	Field Name	Description
0	GOPRO_PROTUNE_EXPOSURE_NEG_5_0	-5.0 EV (Hero 3+ Only)
1	GOPRO_PROTUNE_EXPOSURE_NEG_4_5	-4.5 EV (Hero 3+ Only)
2	GOPRO_PROTUNE_EXPOSURE_NEG_4_0	-4.0 EV (Hero 3+ Only)
3	GOPRO_PROTUNE_EXPOSURE_NEG_3_5	-3.5 EV (Hero 3+ Only)
4	GOPRO_PROTUNE_EXPOSURE_NEG_3_0	-3.0 EV (Hero 3+ Only)
5	GOPRO_PROTUNE_EXPOSURE_NEG_2_5	-2.5 EV (Hero 3+ Only)
6	GOPRO_PROTUNE_EXPOSURE_NEG_2_0	-2.0 EV
7	GOPRO_PROTUNE_EXPOSURE_NEG_1_5	-1.5 EV
8	GOPRO_PROTUNE_EXPOSURE_NEG_1_0	-1.0 EV
9	GOPRO_PROTUNE_EXPOSURE_NEG_0_5	-0.5 EV
10	GOPRO_PROTUNE_EXPOSURE_ZERO	0.0 EV
11	GOPRO_PROTUNE_EXPOSURE_POS_0_5	+0.5 EV
12	GOPRO_PROTUNE_EXPOSURE_POS_1_0	+1.0 EV
13	GOPRO_PROTUNE_EXPOSURE_POS_1_5	+1.5 EV
14	GOPRO_PROTUNE_EXPOSURE_POS_2_0	+2.0 EV
15	GOPRO_PROTUNE_EXPOSURE_POS_2_5	+2.5 EV (Hero 3+ Only)
16	GOPRO_PROTUNE_EXPOSURE_POS_3_0	+3.0 EV (Hero 3+ Only)
17	GOPRO_PROTUNE_EXPOSURE_POS_3_5	+3.5 EV (Hero 3+ Only)
18	GOPRO_PROTUNE_EXPOSURE_POS_4_0	+4.0 EV (Hero 3+ Only)
19	GOPRO_PROTUNE_EXPOSURE_POS_4_5	+4.5 EV (Hero 3+ Only)
20	GOPRO_PROTUNE_EXPOSURE_POS_5_0	+5.0 EV (Hero 3+ Only)

GOPRO_CHARGING

CMD ID	Field Name	Description
0	GOPRO_CHARGING_DISABLED	Charging disabled
1	GOPRO_CHARGING_ENABLED	Charging enabled

GOPRO_MODEL

CMD ID	Field Name	Description
0	GOPRO_MODEL_UNKNOWN	Unknown gopro model
1	GOPRO_MODEL_HERO_3_PLUS_SILVER	Hero 3+ Silver (HeroBus not supported by GoPro)
2	GOPRO_MODEL_HERO_3_PLUS_BLACK	Hero 3+ Black
3	GOPRO_MODEL_HERO_4_SILVER	Hero 4 Silver
4	GOPRO_MODEL_HERO_4_BLACK	Hero 4 Black

GOPRO_BURST_RATE

CMD ID	Field Name	Description
0	GOPRO_BURST_RATE_3_IN_1_SECOND	3 Shots / 1 Second
1	GOPRO_BURST_RATE_5_IN_1_SECOND	5 Shots / 1 Second
2	GOPRO_BURST_RATE_10_IN_1_SECOND	10 Shots / 1 Second
3	GOPRO_BURST_RATE_10_IN_2_SECOND	10 Shots / 2 Second
4	GOPRO_BURST_RATE_10_IN_3_SECOND	10 Shots / 3 Second (Hero 4 Only)
5	GOPRO_BURST_RATE_30_IN_1_SECOND	30 Shots / 1 Second
6	GOPRO_BURST_RATE_30_IN_2_SECOND	30 Shots / 2 Second
7	GOPRO_BURST_RATE_30_IN_3_SECOND	30 Shots / 3 Second
8	GOPRO_BURST_RATE_30_IN_6_SECOND	30 Shots / 6 Second

LED_CONTROL_PATTERN

CMD ID	Field Name	Description
0	LED_CONTROL_PATTERN_OFF	LED patterns off (return control to regular vehicle control)
1	LED_CONTROL_PATTERN_FIRMWAREUPDATE	LEDs show pattern during firmware update
255	LED_CONTROL_PATTERN_CUSTOM	Custom Pattern using custom bytes fields

EKF_STATUS_FLAGS

Flags in EKF_STATUS message

CMD ID	Field Name	Description
1	EKF_ATTITUDE	set if EKF's attitude estimate is good
2	EKF_VELOCITY_HORIZ	set if EKF's horizontal velocity estimate is good
4	EKF_VELOCITY_VERT	set if EKF's vertical velocity estimate is good
8	EKF_POS_HORIZ_REL	set if EKF's horizontal position (relative) estimate is good
16	EKF_POS_HORIZ_ABS	set if EKF's horizontal position (absolute) estimate is good
32	EKF_POS_VERT_ABS	set if EKF's vertical position (absolute) estimate is good
64	EKF_POS_VERT_AGL	set if EKF's vertical position (above ground) estimate is good
128	EKF_CONST_POS_MODE	EKF is in constant position mode and does not know it's absolute or relative position
256	EKF_PRED_POS_HORIZ_REL	set if EKF's predicted horizontal position (relative) estimate is good
512	EKF_PRED_POS_HORIZ_ABS	set if EKF's predicted horizontal position (absolute) estimate is good

PID_TUNING_AXIS

CMD ID	Field Name	Description
1	PID_TUNING_ROLL	
2	PID_TUNING_PITCH	
3	PID_TUNING_YAW	
4	PID_TUNING_ACCZ	
5	PID_TUNING_STEER	
6	PID_TUNING_LANDING	

MAG_CAL_STATUS

CMD ID	Field Name	Description
0	MAG_CAL_NOT_STARTED	
1	MAG_CAL_WAITING_TO_START	
2	MAG_CAL_RUNNING_STEP_ONE	
3	MAG_CAL_RUNNING_STEP_TWO	
4	MAG_CAL_SUCCESS	
5	MAG_CAL_FAILED	

MAV_REMOTE_LOG_DATA_BLOCK_COMMANDS

Special ACK block numbers control activation of dataflash log streaming

CMD ID	Field Name	Description
2147483645	MAV_REMOTE_LOG_DATA_BLOCK_STOP	UAV to stop sending DataFlash blocks
2147483646	MAV_REMOTE_LOG_DATA_BLOCK_START	UAV to start sending DataFlash blocks

MAV_REMOTE_LOG_DATA_BLOCK_STATUSES

Possible remote log data block statuses

CMD ID	Field Name	Description
0	MAV_REMOTE_LOG_DATA_BLOCK_NACK	This block has NOT been received
1	MAV_REMOTE_LOG_DATA_BLOCK_ACK	This block has been received

DEVICE_OP_BUSTYPE

Bus types for device operations

CMD ID	Field Name	Description
0	DEVICE_OP_BUSTYPE_I2C	I2C Device operation
1	DEVICE_OP_BUSTYPE_SPI	SPI Device operation

DEEPSTALL_STAGE

Deepstall flight stage

CMD ID	Field Name	Description
0	DEEPSTALL_STAGE_FLY_TO_LANDING	Flying to the landing point
1	DEEPSTALL_STAGE_ESTIMATE_WIND	Building an estimate of the wind
2	DEEPSTALL_STAGE_WAIT_FOR_BREAKOUT	Waiting to breakout of the loiter to fly the approach
3	DEEPSTALL_STAGE_FLY_TO_ARC	Flying to the first arc point to turn around to the landing point
4	DEEPSTALL_STAGE_ARC	Turning around back to the deepstall landing point
5	DEEPSTALL_STAGE_APPROACH	Approaching the landing point
6	DEEPSTALL_STAGE_LAND	Stalling and steering towards the land point

PLANE_MODE

A mapping of plane flight modes for custom_mode field of heartbeat

CMD ID	Field Name	Description
0	PLANE_MODE_MANUAL	
1	PLANE_MODE_CIRCLE	
2	PLANE_MODE_STABILIZE	
3	PLANE_MODE_TRAINING	
4	PLANE_MODE_ACRO	
5	PLANE_MODE_FLY_BY_WIRE_A	
6	PLANE_MODE_FLY_BY_WIRE_B	
7	PLANE_MODE_CRUISE	
8	PLANE_MODE_AUTOTUNE	
10	PLANE_MODE_AUTO	
11	PLANE_MODE_RTL	
12	PLANE_MODE_LOITER	
14	PLANE_MODE_AVOID_ADSB	
15	PLANE_MODE_GUIDED	
16	PLANE_MODE_INITIALIZING	
17	PLANE_MODE_QSTABILIZE	
18	PLANE_MODE_QHOVER	
19	PLANE_MODE_QLOITER	
20	PLANE_MODE_QLAND	
21	PLANE_MODE_QRTL	

COPTER_MODE

A mapping of copter flight modes for custom_mode field of heartbeat

CMD ID	Field Name	Description
0	COPTER_MODE_STABILIZE	
1	COPTER_MODE_ACRO	
2	COPTER_MODE_ALT_HOLD	
3	COPTER_MODE_AUTO	
4	COPTER_MODE_GUIDED	
5	COPTER_MODE_LOITER	
6	COPTER_MODE_RTL	
7	COPTER_MODE_CIRCLE	
9	COPTER_MODE_LAND	
11	COPTER_MODE_DRIFT	
13	COPTER_MODE_SPORT	
14	COPTER_MODE_FLIP	
15	COPTER_MODE_AUTOTUNE	
16	COPTER_MODE_POSHOLD	
17	COPTER_MODE_BRAKE	
18	COPTER_MODE_THROW	
19	COPTER_MODE_AVOID_ADSB	
20	COPTER_MODE_GUIDED_NOGPS	
21	COPTER_MODE_SMART_RTL	

SUB_MODE

A mapping of sub flight modes for custom_mode field of heartbeat

CMD ID	Field Name	Description
0	SUB_MODE_STABILIZE	
1	SUB_MODE_ACRO	
2	SUB_MODE_ALT_HOLD	
3	SUB_MODE_AUTO	
4	SUB_MODE_GUIDED	
7	SUB_MODE_CIRCLE	
9	SUB_MODE_SURFACE	
16	SUB_MODE_POSHOLD	
19	SUB_MODE_MANUAL	

ROVER_MODE

A mapping of rover flight modes for custom_mode field of heartbeat

CMD ID	Field Name	Description
0	ROVER_MODE_MANUAL	
1	ROVER_MODE_ACRO	
3	ROVER_MODE_STEERING	
4	ROVER_MODE_HOLD	
10	ROVER_MODE_AUTO	
11	ROVER_MODE_RTL	
12	ROVER_MODE_SMART_RTL	
15	ROVER_MODE_GUIDED	
16	ROVER_MODE_INITIALIZING	

TRACKER_MODE

A mapping of antenna tracker flight modes for custom_mode field of heartbeat

CMD ID	Field Name	Description
0	TRACKER_MODE_MANUAL	
1	TRACKER_MODE_STOP	
2	TRACKER_MODE_SCAN	
3	TRACKER_MODE_SERVO_TEST	
10	TRACKER_MODE_AUTO	
16	TRACKER_MODE_INITIALIZING	

MAVLink Messages

SENSOR_OFFSETS (#150)

Offsets and calibrations values for hardware sensors. This makes it easier to debug the calibration process.

Field Name	Type	Description
mag_ofs_x	int16_t	magnetometer X offset
mag_ofs_y	int16_t	magnetometer Y offset
mag_ofs_z	int16_t	magnetometer Z offset
mag_declination	float	magnetic declination (radians) (Units: rad)
raw_press	int32_t	raw pressure from barometer
raw_temp	int32_t	raw temperature from barometer
gyro_cal_x	float	gyro X calibration
gyro_cal_y	float	gyro Y calibration
gyro_cal_z	float	gyro Z calibration
accel_cal_x	float	accel X calibration
accel_cal_y	float	accel Y calibration
accel_cal_z	float	accel Z calibration

SET_MAG_OFFSETS (#151)

Deprecated. Use MAV_CMD_PREFLIGHT_SET_SENSOR_OFFSETS instead. Set the magnetometer offsets

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
mag_ofs_x	int16_t	magnetometer X offset
mag_ofs_y	int16_t	magnetometer Y offset
mag_ofs_z	int16_t	magnetometer Z offset

MEMINFO (#152)

state of APM memory

Field Name	Type	Description
brkval	uint16_t	heap top
freemem	uint16_t	free memory (Units: bytes)
freemem32 **	uint32_t	free memory (32 bit) (Units: bytes)

AP_ADC (#153)

raw ADC output

Field Name	Type	Description
adc1	uint16_t	ADC output 1
adc2	uint16_t	ADC output 2
adc3	uint16_t	ADC output 3
adc4	uint16_t	ADC output 4
adc5	uint16_t	ADC output 5
adc6	uint16_t	ADC output 6

DIGICAM_CONFIGURE (#154)

Configure on-board Camera Control System.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
mode	uint8_t	Mode enumeration from 1 to N //P, TV, AV, M, Etc (0 means ignore)
shutter_speed	uint16_t	Divisor number //e.g. 1000 means 1/1000 (0 means ignore)
aperture	uint8_t	F stop number x 10 //e.g. 28 means 2.8 (0 means ignore)
iso	uint8_t	ISO enumeration from 1 to N //e.g. 80, 100, 200, Etc (0 means ignore)
exposure_type	uint8_t	Exposure type enumeration from 1 to N (0 means ignore)
command_id	uint8_t	Command Identity (incremental loop: 0 to 255)//A command sent multiple times will be executed or pooled just once
engine_cut_off	uint8_t	Main engine cut-off time before camera trigger in seconds/10 (0 means no cut-off) (Units: ds)
extra_param	uint8_t	Extra parameters enumeration (0 means ignore)
extra_value	float	Correspondent value to given extra_param

DIGICAM_CONTROL (#155)

Control on-board Camera Control System to take shots.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
session	uint8_t	0: stop, 1: start or keep it up //Session control e.g. show/hide lens
zoom_pos	uint8_t	1 to N //Zoom's absolute position (0 means ignore)
zoom_step	int8_t	-100 to 100 //Zooming step value to offset zoom from the current position
focus_lock	uint8_t	0: unlock focus or keep unlocked, 1: lock focus or keep locked, 3: re-lock focus
shot	uint8_t	0: ignore, 1: shot or start filming
command_id	uint8_t	Command Identity (incremental loop: 0 to 255)//A command sent multiple times will be executed or pooled just once
extra_param	uint8_t	Extra parameters enumeration (0 means ignore)
extra_value	float	Correspondent value to given extra_param

MOUNT_CONFIGURE (#156)

Message to configure a camera mount, directional antenna, etc.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
mount_mode	uint8_t	mount operating mode (see MAV_MOUNT_MODE enum) (Enum:MAV_MOUNT_MODE)
stab_roll	uint8_t	(1 = yes, 0 = no)
stab_pitch	uint8_t	(1 = yes, 0 = no)
stab_yaw	uint8_t	(1 = yes, 0 = no)

MOUNT_CONTROL (#157)

Message to control a camera mount, directional antenna, etc.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
input_a	int32_t	pitch(deg*100) or lat, depending on mount mode
input_b	int32_t	roll(deg*100) or lon depending on mount mode
input_c	int32_t	yaw(deg*100) or alt (in cm) depending on mount mode
save_position	uint8_t	if "1" it will save current trimmed position on EEPROM (just valid for NEUTRAL and LANDING)

MOUNT_STATUS (#158)

Message with some status from APM to GCS about camera or antenna mount

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
pointing_a	int32_t	pitch(deg*100) (Units: cdeg)
pointing_b	int32_t	roll(deg*100) (Units: cdeg)
pointing_c	int32_t	yaw(deg*100) (Units: cdeg)

FENCE_POINT (#160)

A fence point. Used to set a point when from GCS -> MAV. Also used to return a point from MAV -> GCS

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
idx	uint8_t	point index (first point is 1, 0 is for return point)
count	uint8_t	total number of points (for sanity checking)
lat	float	Latitude of point (Units: deg)
lng	float	Longitude of point (Units: deg)

FENCE_FETCH_POINT (#161)

Request a current fence point from MAV

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
idx	uint8_t	point index (first point is 1, 0 is for return point)

FENCE_STATUS (#162)

Status of geo-fencing. Sent in extended status stream when fencing enabled

Field Name	Type	Description
breach_status	uint8_t	0 if currently inside fence, 1 if outside
breach_count	uint16_t	number of fence breaches
breach_type	uint8_t	last breach type (see FENCE_BREACH_* enum) (Enum:FENCE_BREACH)
breach_time	uint32_t	time of last breach in milliseconds since boot (Units: ms)

AHRS (#163)

Status of DCM attitude estimator

Field Name	Type	Description
omegaIx	float	X gyro drift estimate rad/s (Units: rad/s)
omegaIy	float	Y gyro drift estimate rad/s (Units: rad/s)
omegaIz	float	Z gyro drift estimate rad/s (Units: rad/s)
accel_weight	float	average accel_weight
renorm_val	float	average renormalisation value
error_rp	float	average error_roll_pitch value
error_yaw	float	average error_yaw value

SIMSTATE (#164)

Status of simulation environment, if used

Field Name	Type	Description
roll	float	Roll angle (rad) (Units: rad)
pitch	float	Pitch angle (rad) (Units: rad)
yaw	float	Yaw angle (rad) (Units: rad)
xacc	float	X acceleration m/s/s (Units: m/s/s)
yacc	float	Y acceleration m/s/s (Units: m/s/s)
zacc	float	Z acceleration m/s/s (Units: m/s/s)
xgyro	float	Angular speed around X axis rad/s (Units: rad/s)
ygyro	float	Angular speed around Y axis rad/s (Units: rad/s)
zgyro	float	Angular speed around Z axis rad/s (Units: rad/s)
lat	int32_t	Latitude in degrees * 1E7 (Units: degE7)
lng	int32_t	Longitude in degrees * 1E7 (Units: degE7)

HWSTATUS (#165)

Status of key hardware

Field Name	Type	Description
Vcc	uint16_t	board voltage (mV) (Units: mV)
I2Cerr	uint8_t	I2C error count

RADIO (#166)

Status generated by radio

Field Name	Type	Description
rssi	uint8_t	local signal strength
remrssi	uint8_t	remote signal strength
txbuf	uint8_t	how full the tx buffer is as a percentage (Units: %)
noise	uint8_t	background noise level
remnoise	uint8_t	remote background noise level
rxerrors	uint16_t	receive errors
fixed	uint16_t	count of error corrected packets

LIMITS_STATUS (#167)

Status of AP_Limits. Sent in extended status stream when AP_Limits is enabled

Field Name	Type	Description
limits_state	uint8_t	state of AP_Limits, (see enum LimitState, LIMITS_STATE) (Enum:LIMITS_STATE)
last_trigger	uint32_t	time of last breach in milliseconds since boot (Units: ms)
last_action	uint32_t	time of last recovery action in milliseconds since boot (Units: ms)
last_recovery	uint32_t	time of last successful recovery in milliseconds since boot (Units: ms)
last_clear	uint32_t	time of last all-clear in milliseconds since boot (Units: ms)
breach_count	uint16_t	number of fence breaches
mods_enabled	uint8_t	AP_Limit_Module bitfield of enabled modules, (see enum moduleid or LIMIT_MODULE) (Enum:LIMIT_MODULE)
mods_required	uint8_t	AP_Limit_Module bitfield of required modules, (see enum moduleid or LIMIT_MODULE) (Enum:LIMIT_MODULE)
mods_triggered	uint8_t	AP_Limit_Module bitfield of triggered modules, (see enum moduleid or LIMIT_MODULE) (Enum:LIMIT_MODULE)

WIND (#168)

Wind estimation

Field Name	Type	Description
direction	float	wind direction that wind is coming from (degrees) (Units: deg)
speed	float	wind speed in ground plane (m/s) (Units: m/s)
speed_z	float	vertical wind speed (m/s) (Units: m/s)

DATA16 (#169)

Data packet, size 16

Field Name	Type	Description
type	uint8_t	data type
len	uint8_t	data length (Units: bytes)
data	uint8_t[16]	raw data

DATA32 (#170)

Data packet, size 32

Field Name	Type	Description
type	uint8_t	data type
len	uint8_t	data length (Units: bytes)
data	uint8_t[32]	raw data

DATA64 (#171)

Data packet, size 64

Field Name	Type	Description
type	uint8_t	data type
len	uint8_t	data length (Units: bytes)
data	uint8_t[64]	raw data

DATA96 (#172)

Data packet, size 96

Field Name	Type	Description
type	uint8_t	data type
len	uint8_t	data length (Units: bytes)
data	uint8_t[96]	raw data

RANGEFINDER (#173)

Rangefinder reporting

Field Name	Type	Description
distance	float	distance in meters (Units: m)
voltage	float	raw voltage if available, zero otherwise (Units: V)

AIRSPEED_AUTOCAL (#174)

Airspeed auto-calibration

Field Name	Type	Description
vx	float	GPS velocity north m/s (Units: m/s)
vy	float	GPS velocity east m/s (Units: m/s)
vz	float	GPS velocity down m/s (Units: m/s)
diff_pressure	float	Differential pressure pascals (Units: Pa)
EAS2TAS	float	Estimated to true airspeed ratio
ratio	float	Airspeed ratio
state_x	float	EKF state x
state_y	float	EKF state y
state_z	float	EKF state z
Pax	float	EKF Pax
Pby	float	EKF Pby
Pcz	float	EKF Pcz

RALLY_POINT (#175)

A rally point. Used to set a point when from GCS -> MAV. Also used to return a point from MAV -> GCS

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
idx	uint8_t	point index (first point is 0)
count	uint8_t	total number of points (for sanity checking)
lat	int32_t	Latitude of point in degrees * 1E7 (Units: degE7)
lng	int32_t	Longitude of point in degrees * 1E7 (Units: degE7)
alt	int16_t	Transit / loiter altitude in meters relative to home (Units: m)
break_alt	int16_t	Break altitude in meters relative to home (Units: m)
land_dir	uint16_t	Heading to aim for when landing. In centi-degrees. (Units: cdeg)
flags	uint8_t	See RALLY_FLAGS enum for definition of the bitmask. (Enum:RALLY_FLAGS)

RALLY_FETCH_POINT (#176)

Request a current rally point from MAV. MAV should respond with a RALLY_POINT message. MAV should not respond if the request is invalid.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
idx	uint8_t	point index (first point is 0)

COMPASSMOT_STATUS (#177)

Status of compassmot calibration

Field Name	Type	Description
throttle	uint16_t	throttle (percent*10) (Units: d%)
current	float	current (Ampere) (Units: A)
interference	uint16_t	interference (percent) (Units: %)
CompensationX	float	Motor Compensation X
CompensationY	float	Motor Compensation Y
CompensationZ	float	Motor Compensation Z

AHRS2 (#178)

Status of secondary AHRS filter if available

Field Name	Type	Description
roll	float	Roll angle (rad) (Units: rad)
pitch	float	Pitch angle (rad) (Units: rad)
yaw	float	Yaw angle (rad) (Units: rad)
altitude	float	Altitude (MSL) (Units: m)
lat	int32_t	Latitude in degrees * 1E7 (Units: degE7)
lng	int32_t	Longitude in degrees * 1E7 (Units: degE7)

CAMERA_STATUS (#179)

Camera Event

Field Name	Type	Description
time_usec	uint64_t	Image timestamp (microseconds since UNIX epoch, according to camera clock) (Units: us)
target_system	uint8_t	System ID
cam_idx	uint8_t	Camera ID
img_idx	uint16_t	Image index
event_id	uint8_t	See CAMERA_STATUS_TYPES enum for definition of the bitmask (Enum:CAMERA_STATUS_TYPES)
p1	float	Parameter 1 (meaning depends on event, see CAMERA_STATUS_TYPES enum)
p2	float	Parameter 2 (meaning depends on event, see CAMERA_STATUS_TYPES enum)
p3	float	Parameter 3 (meaning depends on event, see CAMERA_STATUS_TYPES enum)
p4	float	Parameter 4 (meaning depends on event, see CAMERA_STATUS_TYPES enum)

CAMERA_FEEDBACK (#180)

Camera Capture Feedback

Field Name	Type	Description
time_usec	uint64_t	Image timestamp (microseconds since UNIX epoch), as passed in by CAMERA_STATUS message (or autopilot if no CCB) (Units: us)
target_system	uint8_t	System ID
cam_idx	uint8_t	Camera ID
img_idx	uint16_t	Image index
lat	int32_t	Latitude in (deg * 1E7) (Units: degE7)
lng	int32_t	Longitude in (deg * 1E7) (Units: degE7)
alt_msl	float	Altitude Absolute (meters AMSL) (Units: m)
alt_rel	float	Altitude Relative (meters above HOME location) (Units: m)
roll	float	Camera Roll angle (earth frame, degrees, +-180) (Units: deg)
pitch	float	Camera Pitch angle (earth frame, degrees, +-180) (Units: deg)
yaw	float	Camera Yaw (earth frame, degrees, 0-360, true) (Units: deg)
foc_len	float	Focal Length (mm) (Units: mm)
flags	uint8_t	See CAMERA_FEEDBACK_FLAGS enum for definition of the bitmask (Enum:CAMERA_FEEDBACK_FLAGS)

BATTERY2 (#181)

2nd Battery status

Field Name	Type	Description
voltage	uint16_t	voltage in millivolts (Units: mV)
current_battery	int16_t	Battery current, in 10*milliamperes (1 = 10 milliampere), -1: autopilot does not measure the current (Units: cA)

AHRS3 (#182)

Status of third AHRS filter if available. This is for ANU research group (Ali and Sean)

Field Name	Type	Description
roll	float	Roll angle (rad) (Units: rad)
pitch	float	Pitch angle (rad) (Units: rad)
yaw	float	Yaw angle (rad) (Units: rad)
altitude	float	Altitude (MSL) (Units: m)
lat	int32_t	Latitude in degrees * 1E7 (Units: degE7)
lng	int32_t	Longitude in degrees * 1E7 (Units: degE7)
v1	float	test variable1
v2	float	test variable2
v3	float	test variable3
v4	float	test variable4

AUTOPILOT_VERSION_REQUEST (#183)

Request the autopilot version from the system/component.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID

REMOTE_LOG_DATA_BLOCK (#184)

Send a block of log data to remote location

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
seqno	uint32_t	log data block sequence number (Enum:MAV_REMOTE_LOG_DATA_BLOCK_COMMANDS)
data	uint8_t[200]	log data block

REMOTE_LOG_BLOCK_STATUS (#185)

Send Status of each log block that autopilot board might have sent

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
seqno	uint32_t	log data block sequence number
status	uint8_t	log data block status (Enum:MAV_REMOTE_LOG_DATA_BLOCK_STATUSES)

LED_CONTROL (#186)

Control vehicle LEDs

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
instance	uint8_t	Instance (LED instance to control or 255 for all LEDs)
pattern	uint8_t	Pattern (see LED_PATTERN_ENUM)
custom_len	uint8_t	Custom Byte Length
custom_bytes	uint8_t[24]	Custom Bytes

MAG_CAL_PROGRESS (#191)

Reports progress of compass calibration.

Field Name	Type	Description
compass_id	uint8_t	Compass being calibrated
cal_mask	uint8_t	Bitmask of compasses being calibrated
cal_status	uint8_t	Status (see MAG_CAL_STATUS enum) (Enum:MAG_CAL_STATUS)
attempt	uint8_t	Attempt number
completion_pct	uint8_t	Completion percentage (Units: %)
completion_mask	uint8_t[10]	Bitmask of sphere sections (see http://en.wikipedia.org/wiki/Geodesic_grid)
direction_x	float	Body frame direction vector for display
direction_y	float	Body frame direction vector for display
direction_z	float	Body frame direction vector for display

MAG_CAL_REPORT (#192)

Reports results of completed compass calibration. Sent until MAG_CAL_ACK received.

Field Name	Type	Description
compass_id	uint8_t	Compass being calibrated
cal_mask	uint8_t	Bitmask of compasses being calibrated
cal_status	uint8_t	Status (see MAG_CAL_STATUS enum) (Enum:MAG_CAL_STATUS)
autosaved	uint8_t	0=requires a MAV_CMD_DO_ACCEPT_MAG_CAL, 1=saved to parameters
fitness	float	RMS milligauss residuals (Units: mgauss)
ofs_x	float	X offset
ofs_y	float	Y offset
ofs_z	float	Z offset
diag_x	float	X diagonal (matrix 11)
diag_y	float	Y diagonal (matrix 22)
diag_z	float	Z diagonal (matrix 33)
offdiag_x	float	X off-diagonal (matrix 12 and 21)
offdiag_y	float	Y off-diagonal (matrix 13 and 31)
offdiag_z	float	Z off-diagonal (matrix 32 and 23)

EKF_STATUS_REPORT (#193)

EKF Status message including flags and variances

Field Name	Type	Description
flags	uint16_t	Flags (Enum:EKF_STATUS_FLAGS)
velocity_variance	float	Velocity variance
pos_horiz_variance	float	Horizontal Position variance
pos_vert_variance	float	Vertical Position variance
compass_variance	float	Compass variance
terrain_alt_variance	float	Terrain Altitude variance

PID_TUNING (#194)

PID tuning information

Field Name	Type	Description
axis	uint8_t	axis (Enum:PID_TUNING_AXIS)
desired	float	desired rate (degrees/s) (Units: deg/s)
achieved	float	achieved rate (degrees/s) (Units: deg/s)
FF	float	FF component
P	float	P component
I	float	I component
D	float	D component

DEEPSTALL (#195)

Deepstall path planning

Field Name	Type	Description
landing_lat	int32_t	Landing latitude (deg * 1E7) (Units: degE7)
landing_lon	int32_t	Landing longitude (deg * 1E7) (Units: degE7)
path_lat	int32_t	Final heading start point, latitude (deg * 1E7) (Units: degE7)
path_lon	int32_t	Final heading start point, longitude (deg * 1E7) (Units: degE7)
arc_entry_lat	int32_t	Arc entry point, latitude (deg * 1E7) (Units: degE7)
arc_entry_lon	int32_t	Arc entry point, longitude (deg * 1E7) (Units: degE7)
altitude	float	Altitude (meters) (Units: m)
expected_travel_distance	float	Distance the aircraft expects to travel during the deepstall (Units: m)
cross_track_error	float	Deepstall cross track error in meters (only valid when in DEEPSTALL_STAGE_LAND) (Units: m)
stage	uint8_t	Deepstall stage, see enum MAV_DEEPSTALL_STAGE (Enum:DEEPSTALL_STAGE)

GIMBAL_REPORT (#200)

3 axis gimbal mesuraments

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
delta_time	float	Time since last update (seconds) (Units: s)
delta_angle_x	float	Delta angle X (radians) (Units: rad)
delta_angle_y	float	Delta angle Y (radians) (Units: rad)
delta_angle_z	float	Delta angle X (radians) (Units: rad)
delta_velocity_x	float	Delta velocity X (m/s) (Units: m/s)
delta_velocity_y	float	Delta velocity Y (m/s) (Units: m/s)
delta_velocity_z	float	Delta velocity Z (m/s) (Units: m/s)
joint_roll	float	Joint ROLL (radians) (Units: rad)
joint_el	float	Joint EL (radians) (Units: rad)
joint_az	float	Joint AZ (radians) (Units: rad)

GIMBAL_CONTROL (#201)

Control message for rate gimbal

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
demanded_rate_x	float	Demanded angular rate X (rad/s) (Units: rad/s)
demanded_rate_y	float	Demanded angular rate Y (rad/s) (Units: rad/s)
demanded_rate_z	float	Demanded angular rate Z (rad/s) (Units: rad/s)

GIMBAL_TORQUE_CMD_REPORT (#214)

100 Hz gimbal torque command telemetry

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
rl_torque_cmd	int16_t	Roll Torque Command
el_torque_cmd	int16_t	Elevation Torque Command
az_torque_cmd	int16_t	Azimuth Torque Command

GOPRO_HEARTBEAT (#215)

Heartbeat from a HeroBus attached GoPro

Field Name	Type	Description
status	uint8_t	Status (Enum:GOPRO_HEARTBEAT_STATUS)
capture_mode	uint8_t	Current capture mode (Enum:GOPRO_CAPTURE_MODE)
flags	uint8_t	additional status bits (Enum:GOPRO_HEARTBEAT_FLAGS)

GOPRO_GET_REQUEST (#216)

Request a GOPRO_COMMAND response from the GoPro

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
cmd_id	uint8_t	Command ID (Enum:GOPRO_COMMAND)

GOPRO_GET_RESPONSE (#217)

Response from a GOPRO_COMMAND get request

Field Name	Type	Description
cmd_id	uint8_t	Command ID (Enum:GOPRO_COMMAND)
status	uint8_t	Status (Enum:GOPRO_REQUEST_STATUS)
value	uint8_t[4]	Value

GOPRO_SET_REQUEST (#218)

Request to set a GOPRO_COMMAND with a desired

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
cmd_id	uint8_t	Command ID (Enum:GOPRO_COMMAND)
value	uint8_t[4]	Value

GOPRO_SET_RESPONSE (#219)

Response from a GOPRO_COMMAND set request

Field Name	Type	Description
cmd_id	uint8_t	Command ID (Enum:GOPRO_COMMAND)
status	uint8_t	Status (Enum:GOPRO_REQUEST_STATUS)

RPM (#226)

RPM sensor output

Field Name	Type	Description
rpm1	float	RPM Sensor1
rpm2	float	RPM Sensor2

DEVICE_OP_READ (#11000)

(MAVLink 2) Read registers for a device

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
request_id	uint32_t	request ID - copied to reply
bustype	uint8_t	The bus type (Enum:DEVICE_OP_BUSTYPE)
bus	uint8_t	Bus number
address	uint8_t	Bus address
busname	char[40]	Name of device on bus (for SPI)
regstart	uint8_t	First register to read
count	uint8_t	count of registers to read

DEVICE_OP_READ_REPLY (#11001)

(MAVLink 2) Read registers reply

Field Name	Type	Description
request_id	uint32_t	request ID - copied from request
result	uint8_t	0 for success, anything else is failure code
regstart	uint8_t	starting register
count	uint8_t	count of bytes read
data	uint8_t[128]	reply data

DEVICE_OP_WRITE (#11002)

(MAVLink 2) Write registers for a device

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
request_id	uint32_t	request ID - copied to reply
bustype	uint8_t	The bus type (Enum:DEVICE_OP_BUSTYPE)
bus	uint8_t	Bus number
address	uint8_t	Bus address
busname	char[40]	Name of device on bus (for SPI)
regstart	uint8_t	First register to write
count	uint8_t	count of registers to write
data	uint8_t[128]	write data

DEVICE_OP_WRITE_REPLY (#11003)

(MAVLink 2) Write registers reply

Field Name	Type	Description
request_id	uint32_t	request ID - copied from request
result	uint8_t	0 for success, anything else is failure code

ADAP_TUNING (#11010)

(MAVLink 2) Adaptive Controller tuning information

Field Name	Type	Description
axis	uint8_t	axis (Enum:PID_TUNING_AXIS)
desired	float	desired rate (degrees/s) (Units: deg/s)
achieved	float	achieved rate (degrees/s) (Units: deg/s)
error	float	error between model and vehicle
theta	float	theta estimated state predictor
omega	float	omega estimated state predictor
sigma	float	sigma estimated state predictor
theta_dot	float	theta derivative
omega_dot	float	omega derivative
sigma_dot	float	sigma derivative
f	float	projection operator value
f_dot	float	projection operator derivative
u	float	u adaptive controlled output command

VISION_POSITION_DELTA (#11011)

(MAVLink 2) camera vision based attitude and position deltas

Field Name	Type	Description
time_usec	uint64_t	Timestamp (microseconds, synced to UNIX time or since system boot) (Units: us)
time_delta_usec	uint64_t	Time in microseconds since the last reported camera frame (Units: us)
angle_delta	float[3]	Defines a rotation vector in body frame that rotates the vehicle from the previous to the current orientation
position_delta	float[3]	Change in position in meters from previous to current frame rotated into body frame (0=forward, 1=right, 2=down) (Units: m)
confidence	float	normalised confidence value from 0 to 100 (Units: %)

AOA_SSA (#11020)

(MAVLink 2) Angle of Attack and Side Slip Angle

Field Name	Type	Description
time_usec	uint64_t	Timestamp (micros since boot or Unix epoch) (Units: us)
AOA	float	Angle of Attack (degrees) (Units: deg)
SSA	float	Side Slip Angle (degrees) (Units: deg)

MAVLINK Message Set: ASLUAV.xml

This is a human-readable form of the XML definition file: ASLUAV.xml.

MAVLink 2 messages have an ID > 255 and are marked up using **(MAVLink 2)** in their description.

MAVLink 2 extension fields that have been added to MAVLink 1 messages are displayed in blue.

MAVLink Include Files: common.xml

MAVLink Type Enumerations

MAV_CMD

CMD ID	Field Name	Description
40001	MAV_CMD_RESET_MPPT	Mission command to reset Maximum Power Point Tracker (MPPT)
	Mission Param #1	MPPT number
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
40002	MAV_CMD_PAYLOAD_CONTROL	Mission command to perform a power cycle on payload
	Mission Param #1	Complete power cycle
	Mission Param #2	VISensor power cycle
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty

MAVLink Messages

SENS_POWER (#201)

Voltage and current sensor data

Field Name	Type	Description
adc121_vspb_volt	float	Power board voltage sensor reading in volts
adc121_cspb_amp	float	Power board current sensor reading in amps
adc121_cs1_amp	float	Board current sensor 1 reading in amps
adc121_cs2_amp	float	Board current sensor 2 reading in amps

SENS_MPPT (#202)

Maximum Power Point Tracker (MPPT) sensor data for solar module power performance tracking

Field Name	Type	Description
mppt_timestamp	uint64_t	MPPT last timestamp
mppt1_volt	float	MPPT1 voltage
mppt1_amp	float	MPPT1 current
mppt1_pwm	uint16_t	MPPT1 pwm
mppt1_status	uint8_t	MPPT1 status
mppt2_volt	float	MPPT2 voltage
mppt2_amp	float	MPPT2 current
mppt2_pwm	uint16_t	MPPT2 pwm
mppt2_status	uint8_t	MPPT2 status
mppt3_volt	float	MPPT3 voltage
mppt3_amp	float	MPPT3 current
mppt3_pwm	uint16_t	MPPT3 pwm
mppt3_status	uint8_t	MPPT3 status

ASLCTRL_DATA (#203)

ASL-fixed-wing controller data

Field Name	Type	Description
timestamp	uint64_t	Timestamp
aslctrl_mode	uint8_t	ASLCTRL control-mode (manual, stabilized, auto, etc...)
h	float	See sourcecode for a description of these values...
hRef	float	
hRef_t	float	
PitchAngle	float	Pitch angle [deg]
PitchAngleRef	float	Pitch angle reference[deg]
q	float	
qRef	float	
uElev	float	
uThrot	float	
uThrot2	float	
nZ	float	
AirspeedRef	float	Airspeed reference [m/s]
SpoilersEngaged	uint8_t	
YawAngle	float	Yaw angle [deg]
YawAngleRef	float	Yaw angle reference[deg]
RollAngle	float	Roll angle [deg]
RollAngleRef	float	Roll angle reference[deg]
p	float	
pRef	float	
r	float	
rRef	float	
uAil	float	
uRud	float	

ASLCTRL_DEBUG (#204)

ASL-fixed-wing controller debug data

Field Name	Type	Description
i32_1	uint32_t	Debug data
i8_1	uint8_t	Debug data
i8_2	uint8_t	Debug data
f_1	float	Debug data
f_2	float	Debug data
f_3	float	Debug data
f_4	float	Debug data
f_5	float	Debug data
f_6	float	Debug data
f_7	float	Debug data
f_8	float	Debug data

ASLUAV_STATUS (#205)

Extended state information for ASLUAVs

Field Name	Type	Description
LED_status	uint8_t	Status of the position-indicator LEDs
SATCOM_status	uint8_t	Status of the IRIDIUM satellite communication system
Servo_status	uint8_t[8]	Status vector for up to 8 servos
Motor_rpm	float	Motor RPM

EKF_EXT (#206)

Extended EKF state estimates for ASLUAVs

Field Name	Type	Description
timestamp	uint64_t	Time since system start [us]
Windspeed	float	Magnitude of wind velocity (in lateral inertial plane) [m/s]
WindDir	float	Wind heading angle from North [rad]
WindZ	float	Z (Down) component of inertial wind velocity [m/s]
Airspeed	float	Magnitude of air velocity [m/s]
beta	float	Sideslip angle [rad]
alpha	float	Angle of attack [rad]

ASL_OBCTRL (#207)

Off-board controls/commands for ASLUAVs

Field Name	Type	Description
timestamp	uint64_t	Time since system start [us]
uElev	float	Elevator command [~]
uThrot	float	Throttle command [~]
uThrot2	float	Throttle 2 command [~]
uAilL	float	Left aileron command [~]
uAilR	float	Right aileron command [~]
uRud	float	Rudder command [~]
obctrl_status	uint8_t	Off-board computer status

SENS_ATMOS (#208)

Atmospheric sensors (temperature, humidity, ...)

Field Name	Type	Description
TempAmbient	float	Ambient temperature [degrees Celsius]
Humidity	float	Relative humidity [%]

SENS_BATMON (#209)

Battery pack monitoring data for Li-Ion batteries

Field Name	Type	Description
batmon_timestamp	uint64_t	Time since system start [us]
temperature	float	Battery pack temperature in [deg C]
voltage	uint16_t	Battery pack voltage in [mV]
current	int16_t	Battery pack current in [mA]
SoC	uint8_t	Battery pack state-of-charge
batterystatus	uint16_t	Battery monitor status report bits in Hex
serialnumber	uint16_t	Battery monitor serial number in Hex
safetystatus	uint32_t	Battery monitor safetystatus report bits in Hex
operationstatus	uint32_t	Battery monitor operation status report bits in Hex
cellvoltage1	uint16_t	Battery pack cell 1 voltage in [mV]
cellvoltage2	uint16_t	Battery pack cell 2 voltage in [mV]
cellvoltage3	uint16_t	Battery pack cell 3 voltage in [mV]
cellvoltage4	uint16_t	Battery pack cell 4 voltage in [mV]
cellvoltage5	uint16_t	Battery pack cell 5 voltage in [mV]
cellvoltage6	uint16_t	Battery pack cell 6 voltage in [mV]

FW_SOARING_DATA (#210)

Fixed-wing soaring (i.e. thermal seeking) data

Field Name	Type	Description
timestamp	uint64_t	Timestamp [ms]
timestampModeChanged	uint64_t	Timestamp since last mode change[ms]
xW	float	Thermal core updraft strength [m/s]
xR	float	Thermal radius [m]
xLat	float	Thermal center latitude [deg]
xLon	float	Thermal center longitude [deg]
VarW	float	Variance W
VarR	float	Variance R
VarLat	float	Variance Lat
VarLon	float	Variance Lon
LoiterRadius	float	Suggested loiter radius [m]
LoiterDirection	float	Suggested loiter direction
DistToSoarPoint	float	Distance to soar point [m]
vSinkExp	float	Expected sink rate at current airspeed, roll and throttle [m/s]
z1_LocalUpdraftSpeed	float	Measurement / updraft speed at current/local airplane position [m/s]
z2_DeltaRoll	float	Measurement / roll angle tracking error [deg]
z1_exp	float	Expected measurement 1
z2_exp	float	Expected measurement 2
ThermalGSNorth	float	Thermal drift (from estimator prediction step only) [m/s]
ThermalGSEast	float	Thermal drift (from estimator prediction step only) [m/s]
TSE_dot	float	Total specific energy change (filtered) [m/s]
DebugVar1	float	Debug variable 1
DebugVar2	float	Debug variable 2
ControlMode	uint8_t	Control Mode [-]
valid	uint8_t	Data valid [-]

SENSORPOD_STATUS (#211)

Monitoring of sensorpod status

Field Name	Type	Description
timestamp	uint64_t	Timestamp in linuxtime [ms] (since 1.1.1970)
visensor_rate_1	uint8_t	Rate of ROS topic 1
visensor_rate_2	uint8_t	Rate of ROS topic 2
visensor_rate_3	uint8_t	Rate of ROS topic 3
visensor_rate_4	uint8_t	Rate of ROS topic 4
recording_nodes_count	uint8_t	Number of recording nodes
cpu_temp	uint8_t	Temperature of sensorpod CPU in [deg C]
free_space	uint16_t	Free space available in recordings directory in [Gb] * 1e2

SENS_POWER_BOARD (#212)

Monitoring of power board status

Field Name	Type	Description
timestamp	uint64_t	Timestamp
pwr_brd_status	uint8_t	Power board status register
pwr_brd_led_status	uint8_t	Power board leds status
pwr_brd_system_volt	float	Power board system voltage
pwr_brd_servo_volt	float	Power board servo voltage
pwr_brd_digital_volt	float	Power board digital voltage
pwr_brd_mot_l_amp	float	Power board left motor current sensor
pwr_brd_mot_r_amp	float	Power board right motor current sensor
pwr_brd_analog_amp	float	Power board analog current sensor
pwr_brd_digital_amp	float	Power board digital current sensor
pwr_brd_ext_amp	float	Power board extension current sensor
pwr_brd_aux_amp	float	Power board aux current sensor

MAVLINK Message Set: autoquad.xml

This is a human-readable form of the XML definition file: autoquad.xml.

> MAVLink 2 messages have an ID > 255 and are marked up using **(MAVLink 2)** in their description.

> MAVLink 2 extension fields that have been added to MAVLink 1 messages are displayed in blue.

MAVLink Include Files: common.xml

MAVLink Protocol Version

The current MAVLink version is 2.3. The minor version numbers (after the dot) range from 1-255.

MAVLink Type Enumerations

AUTOQUAD_MAVLINK_DEFS_VERSION

Track current version of these definitions (can be used by checking value of AUTOQUAD_MAVLINK_DEFS_VERSION_ENUM_END). Append a new entry for each published change.

CMD ID	Field Name	Description
	AQ_MAVLINK_DEFS_VERSION_1	

AUTOQUAD_NAV_STATUS

Available operating modes/statuses for AutoQuad flight controller. Bitmask up to 32 bits. Low side bits for base modes, high side for additional active features/modifiers/constraints.

CMD ID	Field Name	Description
0	AQ_NAV_STATUS_INIT	System is initializing
0x00000001	AQ_NAV_STATUS_STANDBY	System is *armed* and standing by, with no throttle input and no autonomous mode
0x00000002	AQ_NAV_STATUS_MANUAL	Flying (throttle input detected), assumed under manual control unless other mode bits are set
0x00000004	AQ_NAV_STATUS_ALTHOLD	Altitude hold engaged
0x00000008	AQ_NAV_STATUS_POSHOLD	Position hold engaged
0x00000010	AQ_NAV_STATUS_GUIDED	Externally-guided (eg. GCS) navigation mode
0x00000020	AQ_NAV_STATUS_MISSION	Autonomous mission execution mode
0x00000100	AQ_NAV_STATUS_READY	Ready but *not armed*
0x00000200	AQ_NAV_STATUS_CALIBRATING	Calibration mode active
0x00001000	AQ_NAV_STATUS_NO_RC	No valid control input (eg. no radio link)
0x00002000	AQ_NAV_STATUS_FUEL_LOW	Battery is low (stage 1 warning)
0x00004000	AQ_NAV_STATUS_FUEL_CRITICAL	Battery is depleted (stage 2 warning)
0x01000000	AQ_NAV_STATUS_DVH	Dynamic Velocity Hold is active (PH with proportional manual direction override)
0x02000000	AQ_NAV_STATUS_DAO	ynamic Altitude Override is active (AH with proportional manual adjustment)
0x04000000	AQ_NAV_STATUS_CEILING_REACHED	Craft is at ceiling altitude
0x08000000	AQ_NAV_STATUS_CEILING	Ceiling altitude is set
0x10000000	AQ_NAV_STATUS_HF_DYNAMIC	Heading-Free dynamic mode active
0x20000000	AQ_NAV_STATUS_HF_LOCKED	Heading-Free locked mode active
0x40000000	AQ_NAV_STATUS_RTH	Automatic Return to Home is active
0x80000000	AQ_NAV_STATUS_FAILSAFE	System is in failsafe recovery mode

MAV_CMD

CMD ID	Field Name	Description
1	MAV_CMD_AQ_NAV_LEG_ORBIT	Orbit a waypoint.
	Mission Param #1	Orbit radius in meters
	Mission Param #2	Loiter time in decimal seconds
	Mission Param #3	Maximum horizontal speed in m/s
	Mission Param #4	Desired yaw angle at waypoint
	Mission Param #5	Latitude
	Mission Param #6	Longitude
	Mission Param #7	Altitude
2	MAV_CMD_AQ_TELEMETRY	Start/stop AutoQuad telemetry values stream.
	Mission Param #1	Start or stop (1 or 0)
	Mission Param #2	Stream frequency in us
	Mission Param #3	Dataset ID (refer to aq_mavlink.h::mavlinkCustomDataSets enum in AQ flight controller code)
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty
4	MAV_CMD_AQ_REQUEST_VERSION	Request AutoQuad firmware version number.
	Mission Param #1	Empty
	Mission Param #2	Empty
	Mission Param #3	Empty
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty

MAV_DATA_STREAM

CMD ID	Field Name	Description
	MAV_DATA_STREAM_PROPULSION	Motor/ESC telemetry data.

MAVLink Messages

AQ_TELEMETRY_F (#150)

Sends up to 20 raw float values.

Field Name	Type	Description
Index	uint16_t	Index of message
value1	float	value1
value2	float	value2
value3	float	value3
value4	float	value4
value5	float	value5
value6	float	value6
value7	float	value7
value8	float	value8
value9	float	value9
value10	float	value10
value11	float	value11
value12	float	value12
value13	float	value13
value14	float	value14
value15	float	value15
value16	float	value16
value17	float	value17
value18	float	value18
value19	float	value19
value20	float	value20

AQ_ESC_TELEMETRY (#152)

Sends ESC32 telemetry data for up to 4 motors. Multiple messages may be sent in sequence when system has > 4 motors. Data is described as follows: // unsigned int state : 3; // unsigned int vin : 12; // x 100 // unsigned int amps : 14; // x 100 // unsigned int rpm : 15; // unsigned int duty : 8; // x (255/100) // - Data Version 2 - // unsigned int errors : 9; // Bad detects error count // - Data Version 3 - // unsigned int temp : 9; // (Deg C + 32) * 4 // unsigned int errCode : 3;

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp of the component clock since boot time in ms.
seq	uint8_t	Sequence number of message (first set of 4 motors is #1, next 4 is #2, etc).
num_motors	uint8_t	Total number of active ESCs/motors on the system.
num_in_seq	uint8_t	Number of active ESCs in this sequence (1 through this many array members will be populated with data)
escid	uint8_t[4]	ESC/Motor ID
status_age	uint16_t[4]	Age of each ESC telemetry reading in ms compared to boot time. A value of 0xFFFF means timeout/no data.
data_version	uint8_t[4]	Version of data structure (determines contents).
data0	uint32_t[4]	Data bits 1-32 for each ESC.
data1	uint32_t[4]	Data bits 33-64 for each ESC.

MAVLINK Message Set: matrixpilot.xml

This is a human-readable form of the XML definition file: matrixpilot.xml.

MAVLink 2 messages have an ID > 255 and are marked up using **(MAVLink 2)** in their description.

MAVLink 2 extension fields that have been added to MAVLink 1 messages are displayed in blue.

MAVLink Include Files: common.xml

MAVLink Type Enumerations

MAV_PREFLIGHT_STORAGE_ACTION

Action required when performing CMD_PREFLIGHT_STORAGE

CMD ID	Field Name	Description
0	MAV_PFS_CMD_READ_ALL	Read all parameters from storage
1	MAV_PFS_CMD_WRITE_ALL	Write all parameters to storage
2	MAV_PFS_CMD_CLEAR_ALL	Clear all parameters in storage
3	MAV_PFS_CMD_READ_SPECIFIC	Read specific parameters from storage
4	MAV_PFS_CMD_WRITE_SPECIFIC	Write specific parameters to storage
5	MAV_PFS_CMD_CLEAR_SPECIFIC	Clear specific parameters in storage
6	MAV_PFS_CMD_DO_NOTHING	do nothing

MAV_CMD

CMD ID	Field Name	Description
0	MAV_CMD_PREFLIGHT_STORAGE_ADVANCED	Request storage of different parameter values and logs. This command will be only accepted if in pre-flight mode.
	Mission Param #1	Storage action: Action defined by MAV_PREFLIGHT_STORAGE_ACTION_ADVANCED
	Mission Param #2	Storage area as defined by parameter database
	Mission Param #3	Storage flags as defined by parameter database
	Mission Param #4	Empty
	Mission Param #5	Empty
	Mission Param #6	Empty
	Mission Param #7	Empty

MAVLink Messages

FLEXIFUNCTION_SET (#150)

Depreciated but used as a compiler flag. Do not remove

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID

FLEXIFUNCTION_READ_REQ (#151)

Reqest reading of flexifunction data

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
read_req_type	int16_t	Type of flexifunction data requested
data_index	int16_t	index into data where needed

FLEXIFUNCTION_BUFFER_FUNCTION (#152)

Flexifunction type and parameters for component at function index from buffer

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
func_index	uint16_t	Function index
func_count	uint16_t	Total count of functions
data_address	uint16_t	Address in the flexifunction data, Set to 0xFFFF to use address in target memory
data_size	uint16_t	Size of the
data	int8_t[48]	Settings data

FLEXIFUNCTION_BUFFER_FUNCTION_ACK (#153)

Flexifunction type and parameters for component at function index from buffer

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
func_index	uint16_t	Function index
result	uint16_t	result of acknowledge, 0=fail, 1=good

FLEXIFUNCTION_DIRECTORY (#155)

Acknowldge sucess or failure of a flexifunction command

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
directory_type	uint8_t	0=inputs, 1=outputs
start_index	uint8_t	index of first directory entry to write
count	uint8_t	count of directory entries to write
directory_data	int8_t[48]	Settings data

FLEXIFUNCTION_DIRECTORY_ACK (#156)

Acknowldge sucess or failure of a flexifunction command

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
directory_type	uint8_t	0=inputs, 1=outputs
start_index	uint8_t	index of first directory entry to write
count	uint8_t	count of directory entries to write
result	uint16_t	result of acknowledge, 0=fail, 1=good

FLEXIFUNCTION_COMMAND (#157)

Acknowldge sucess or failure of a flexifunction command

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
command_type	uint8_t	Flexifunction command type

FLEXIFUNCTION_COMMAND_ACK (#158)

Acknowldge sucess or failure of a flexifunction command

Field Name	Type	Description
command_type	uint16_t	Command acknowledged
result	uint16_t	result of acknowledge

SERIAL_UDB_EXTRA_F2_A (#170)

Backwards compatible MAVLink version of SERIAL_UDB_EXTRA - F2: Format Part A

Field Name	Type	Description
sue_time	uint32_t	Serial UDB Extra Time
sue_status	uint8_t	Serial UDB Extra Status
sue_latitude	int32_t	Serial UDB Extra Latitude
sue_longitude	int32_t	Serial UDB Extra Longitude
sue_altitude	int32_t	Serial UDB Extra Altitude
sue_waypoint_index	uint16_t	Serial UDB Extra Waypoint Index
sue_rmat0	int16_t	Serial UDB Extra Rmat 0
sue_rmat1	int16_t	Serial UDB Extra Rmat 1
sue_rmat2	int16_t	Serial UDB Extra Rmat 2
sue_rmat3	int16_t	Serial UDB Extra Rmat 3
sue_rmat4	int16_t	Serial UDB Extra Rmat 4
sue_rmat5	int16_t	Serial UDB Extra Rmat 5
sue_rmat6	int16_t	Serial UDB Extra Rmat 6
sue_rmat7	int16_t	Serial UDB Extra Rmat 7
sue_rmat8	int16_t	Serial UDB Extra Rmat 8
sue_cog	uint16_t	Serial UDB Extra GPS Course Over Ground
sue_sog	int16_t	Serial UDB Extra Speed Over Ground
sue_cpu_load	uint16_t	Serial UDB Extra CPU Load
sue_air_speed_3DIMU	uint16_t	Serial UDB Extra 3D IMU Air Speed
sue_estimated_wind_0	int16_t	Serial UDB Extra Estimated Wind 0
sue_estimated_wind_1	int16_t	Serial UDB Extra Estimated Wind 1
sue_estimated_wind_2	int16_t	Serial UDB Extra Estimated Wind 2
sue_magFieldEarth0	int16_t	Serial UDB Extra Magnetic Field Earth 0
sue_magFieldEarth1	int16_t	Serial UDB Extra Magnetic Field Earth 1
sue_magFieldEarth2	int16_t	Serial UDB Extra Magnetic Field Earth 2
sue_svs	int16_t	Serial UDB Extra Number of Sattelites in View
sue_hdop	int16_t	Serial UDB Extra GPS Horizontal Dilution of Precision

SERIAL_UDB_EXTRA_F2_B (#171)

Backwards compatible version of SERIAL_UDB_EXTRA - F2: Part B

Field Name	Type	Description
sue_time	uint32_t	Serial UDB Extra Time
sue_pwm_input_1	int16_t	Serial UDB Extra PWM Input Channel 1
sue_pwm_input_2	int16_t	Serial UDB Extra PWM Input Channel 2
sue_pwm_input_3	int16_t	Serial UDB Extra PWM Input Channel 3
sue_pwm_input_4	int16_t	Serial UDB Extra PWM Input Channel 4

sue_pwm_input_5	int16_t	Serial UDB Extra PWM Input Channel 5
sue_pwm_input_6	int16_t	Serial UDB Extra PWM Input Channel 6
sue_pwm_input_7	int16_t	Serial UDB Extra PWM Input Channel 7
sue_pwm_input_8	int16_t	Serial UDB Extra PWM Input Channel 8
sue_pwm_input_9	int16_t	Serial UDB Extra PWM Input Channel 9
sue_pwm_input_10	int16_t	Serial UDB Extra PWM Input Channel 10
sue_pwm_input_11	int16_t	Serial UDB Extra PWM Input Channel 11
sue_pwm_input_12	int16_t	Serial UDB Extra PWM Input Channel 12
sue_pwm_output_1	int16_t	Serial UDB Extra PWM Output Channel 1
sue_pwm_output_2	int16_t	Serial UDB Extra PWM Output Channel 2
sue_pwm_output_3	int16_t	Serial UDB Extra PWM Output Channel 3
sue_pwm_output_4	int16_t	Serial UDB Extra PWM Output Channel 4
sue_pwm_output_5	int16_t	Serial UDB Extra PWM Output Channel 5
sue_pwm_output_6	int16_t	Serial UDB Extra PWM Output Channel 6
sue_pwm_output_7	int16_t	Serial UDB Extra PWM Output Channel 7
sue_pwm_output_8	int16_t	Serial UDB Extra PWM Output Channel 8
sue_pwm_output_9	int16_t	Serial UDB Extra PWM Output Channel 9
sue_pwm_output_10	int16_t	Serial UDB Extra PWM Output Channel 10
sue_pwm_output_11	int16_t	Serial UDB Extra PWM Output Channel 11
sue_pwm_output_12	int16_t	Serial UDB Extra PWM Output Channel 12
sue_imu_location_x	int16_t	Serial UDB Extra IMU Location X
sue_imu_location_y	int16_t	Serial UDB Extra IMU Location Y
sue_imu_location_z	int16_t	Serial UDB Extra IMU Location Z
sue_location_error_earth_x	int16_t	Serial UDB Location Error Earth X
sue_location_error_earth_y	int16_t	Serial UDB Location Error Earth Y
sue_location_error_earth_z	int16_t	Serial UDB Location Error Earth Z
sue_flags	uint32_t	Serial UDB Extra Status Flags
sue_osc_fails	int16_t	Serial UDB Extra Oscillator Failure Count
sue_imu_velocity_x	int16_t	Serial UDB Extra IMU Velocity X
sue_imu_velocity_y	int16_t	Serial UDB Extra IMU Velocity Y
sue_imu_velocity_z	int16_t	Serial UDB Extra IMU Velocity Z
sue_waypoint_goal_x	int16_t	Serial UDB Extra Current Waypoint Goal X
sue_waypoint_goal_y	int16_t	Serial UDB Extra Current Waypoint Goal Y
sue_waypoint_goal_z	int16_t	Serial UDB Extra Current Waypoint Goal Z
sue_aero_x	int16_t	Aeroforce in UDB X Axis
sue_aero_y	int16_t	Aeroforce in UDB Y Axis
sue_aero_z	int16_t	Aeroforce in UDB Z axis
sue_barom_temp	int16_t	SUE barometer temperature

sue_barom_press	int32_t	SUE barometer pressure
sue_barom_alt	int32_t	SUE barometer altitude
sue_bat_volt	int16_t	SUE battery voltage
sue_bat_amp	int16_t	SUE battery current
sue_bat_amp_hours	int16_t	SUE battery milli amp hours used
sue_desired_height	int16_t	Sue autopilot desired height
sue_memory_stack_free	int16_t	Serial UDB Extra Stack Memory Free

SERIAL_UDB_EXTRA_F4 (#172)

Backwards compatible version of SERIAL_UDB_EXTRA F4: format

Field Name	Type	Description
sue_ROLL_STABILIZATION_AILERONS	uint8_t	Serial UDB Extra Roll Stabilization with Ailerons Enabled
sue_ROLL_STABILIZATION_RUDDER	uint8_t	Serial UDB Extra Roll Stabilization with Rudder Enabled
sue_PITCH_STABILIZATION	uint8_t	Serial UDB Extra Pitch Stabilization Enabled
sue_YAW_STABILIZATION_RUDDER	uint8_t	Serial UDB Extra Yaw Stabilization using Rudder Enabled
sue_YAW_STABILIZATION_AILERON	uint8_t	Serial UDB Extra Yaw Stabilization using Ailerons Enabled
sue_AILERON_NAVIGATION	uint8_t	Serial UDB Extra Navigation with Ailerons Enabled
sue_RUDDER_NAVIGATION	uint8_t	Serial UDB Extra Navigation with Rudder Enabled
sue_ALTITUDEHOLD_STABILIZED	uint8_t	Serial UDB Extra Type of Alitude Hold when in Stabilized Mode
sue_ALTITUDEHOLD_WAYPOINT	uint8_t	Serial UDB Extra Type of Alitude Hold when in Waypoint Mode
sue_RACING_MODE	uint8_t	Serial UDB Extra Firmware racing mode enabled

SERIAL_UDB_EXTRA_F5 (#173)

Backwards compatible version of SERIAL_UDB_EXTRA F5: format

Field Name	Type	Description
sue_YAWKP_AILERON	float	Serial UDB YAWKP_AILERON Gain for Proporional control of navigation
sue_YAWKD_AILERON	float	Serial UDB YAWKD_AILERON Gain for Rate control of navigation
sue_ROLLKP	float	Serial UDB Extra ROLLKP Gain for Proportional control of roll stabilization
sue_ROLLKD	float	Serial UDB Extra ROLLKD Gain for Rate control of roll stabilization

SERIAL_UDB_EXTRA_F6 (#174)

Backwards compatible version of SERIAL_UDB_EXTRA F6: format

Field Name	Type	Description
sue_PITCHGAIN	float	Serial UDB Extra PITCHGAIN Proportional Control
sue_PITCHKD	float	Serial UDB Extra Pitch Rate Control
sue_RUDDER_ELEV_MIX	float	Serial UDB Extra Rudder to Elevator Mix
sue_ROLL_ELEV_MIX	float	Serial UDB Extra Roll to Elevator Mix
sue_ELEVATOR_BOOST	float	Gain For Boosting Manual Elevator control When Plane Stabilized

SERIAL_UDB_EXTRA_F7 (#175)

Backwards compatible version of SERIAL_UDB_EXTRA F7: format

Field Name	Type	Description
sue_YAWKP_RUDDER	float	Serial UDB YAWKP_RUDDER Gain for Proporional control of navigation
sue_YAWKD_RUDDER	float	Serial UDB YAWKD_RUDDER Gain for Rate control of navigation
sue_ROLLKP_RUDDER	float	Serial UDB Extra ROLLKP_RUDDER Gain for Proportional control of roll stabilization
sue_ROLLKD_RUDDER	float	Serial UDB Extra ROLLKD_RUDDER Gain for Rate control of roll stabilization
sue_RUDDER_BOOST	float	SERIAL UDB EXTRA Rudder Boost Gain to Manual Control when stabilized
sue_RTL_PITCH_DOWN	float	Serial UDB Extra Return To Landing - Angle to Pitch Plane Down

SERIAL_UDB_EXTRA_F8 (#176)

Backwards compatible version of SERIAL_UDB_EXTRA F8: format

Field Name	Type	Description
sue_HEIGHT_TARGET_MAX	float	Serial UDB Extra HEIGHT_TARGET_MAX
sue_HEIGHT_TARGET_MIN	float	Serial UDB Extra HEIGHT_TARGET_MIN
sue_ALT_HOLD_THROTTLE_MIN	float	Serial UDB Extra ALT_HOLD_THROTTLE_MIN
sue_ALT_HOLD_THROTTLE_MAX	float	Serial UDB Extra ALT_HOLD_THROTTLE_MAX
sue_ALT_HOLD_PITCH_MIN	float	Serial UDB Extra ALT_HOLD_PITCH_MIN
sue_ALT_HOLD_PITCH_MAX	float	Serial UDB Extra ALT_HOLD_PITCH_MAX
sue_ALT_HOLD_PITCH_HIGH	float	Serial UDB Extra ALT_HOLD_PITCH_HIGH

SERIAL_UDB_EXTRA_F13 (#177)

Backwards compatible version of SERIAL_UDB_EXTRA F13: format

Field Name	Type	Description
sue_week_no	int16_t	Serial UDB Extra GPS Week Number
sue_lat_origin	int32_t	Serial UDB Extra MP Origin Latitude
sue_lon_origin	int32_t	Serial UDB Extra MP Origin Longitude
sue_alt_origin	int32_t	Serial UDB Extra MP Origin Altitude Above Sea Level

SERIAL_UDB_EXTRA_F14 (#178)

Backwards compatible version of SERIAL_UDB_EXTRA F14: format

Field Name	Type	Description
sue_WIND_ESTIMATION	uint8_t	Serial UDB Extra Wind Estimation Enabled
sue_GPS_TYPE	uint8_t	Serial UDB Extra Type of GPS Unit
sue_DR	uint8_t	Serial UDB Extra Dead Reckoning Enabled
sue_BOARD_TYPE	uint8_t	Serial UDB Extra Type of UDB Hardware
sue_AIRFRAME	uint8_t	Serial UDB Extra Type of Airframe
sue_RCON	int16_t	Serial UDB Extra Reboot Register of DSPIC
sue_TRAP_FLAGS	int16_t	Serial UDB Extra Last dspic Trap Flags
sue_TRAP_SOURCE	uint32_t	Serial UDB Extra Type Program Address of Last Trap
sue_osc_fail_count	int16_t	Serial UDB Extra Number of Ocillator Failures
sue_CLOCK_CONFIG	uint8_t	Serial UDB Extra UDB Internal Clock Configuration
sue_FLIGHT_PLAN_TYPE	uint8_t	Serial UDB Extra Type of Flight Plan

SERIAL_UDB_EXTRA_F15 (#179)

Backwards compatible version of SERIAL_UDB_EXTRA F15 format

Field Name	Type	Description
sue_ID_VEHICLE_MODEL_NAME	uint8_t[40]	Serial UDB Extra Model Name Of Vehicle
sue_ID_VEHICLE_REGISTRATION	uint8_t[20]	Serial UDB Extra Registraton Number of Vehicle

SERIAL_UDB_EXTRA_F16 (#180)

Backwards compatible version of SERIAL_UDB_EXTRA F16 format

Field Name	Type	Description
sue_ID_LEAD_PILOT	uint8_t[40]	Serial UDB Extra Name of Expected Lead Pilot
sue_ID_DIY_DRONES_URL	uint8_t[70]	Serial UDB Extra URL of Lead Pilot or Team

ALTITUDES (#181)

The altitude measured by sensors and IMU

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot)
alt_gps	int32_t	GPS altitude in meters, expressed as * 1000 (millimeters), above MSL
alt_imu	int32_t	IMU altitude above ground in meters, expressed as * 1000 (millimeters)
alt_barometric	int32_t	barometeric altitude above ground in meters, expressed as * 1000 (millimeters)
alt_optical_flow	int32_t	Optical flow altitude above ground in meters, expressed as * 1000 (millimeters)
alt_range_finder	int32_t	Rangefinder Altitude above ground in meters, expressed as * 1000 (millimeters)
alt_extra	int32_t	Extra altitude above ground in meters, expressed as * 1000 (millimeters)

AIRSPEEDS (#182)

The airspeed measured by sensors and IMU

Field Name	Type	Description
time_boot_ms	uint32_t	Timestamp (milliseconds since system boot)
airspeed_imu	int16_t	Airspeed estimate from IMU, cm/s
airspeed_pitot	int16_t	Pitot measured forward airpseed, cm/s
airspeed_hot_wire	int16_t	Hot wire anenometer measured airspeed, cm/s
airspeed_ultrasonic	int16_t	Ultrasonic measured airspeed, cm/s
aoa	int16_t	Angle of attack sensor, degrees * 10
aoy	int16_t	Yaw angle sensor, degrees * 10

SERIAL_UDB_EXTRA_F17 (#183)

Backwards compatible version of SERIAL_UDB_EXTRA F17 format

Field Name	Type	Description
sue_feed_forward	float	SUE Feed Forward Gain
sue_turn_rate_nav	float	SUE Max Turn Rate when Navigating
sue_turn_rate_fbw	float	SUE Max Turn Rate in Fly By Wire Mode

SERIAL_UDB_EXTRA_F18 (#184)

Backwards compatible version of SERIAL_UDB_EXTRA F18 format

Field Name	Type	Description
angle_of_attack_normal	float	SUE Angle of Attack Normal
angle_of_attack_inverted	float	SUE Angle of Attack Inverted
elevator_trim_normal	float	SUE Elevator Trim Normal
elevator_trim_inverted	float	SUE Elevator Trim Inverted
reference_speed	float	SUE reference_speed

SERIAL_UDB_EXTRA_F19 (#185)

Backwards compatible version of SERIAL_UDB_EXTRA F19 format

Field Name	Type	Description
sue_aileron_output_channel	uint8_t	SUE aileron output channel
sue_aileron_reversed	uint8_t	SUE aileron reversed
sue_elevator_output_channel	uint8_t	SUE elevator output channel
sue_elevator_reversed	uint8_t	SUE elevator reversed
sue_throttle_output_channel	uint8_t	SUE throttle output channel
sue_throttle_reversed	uint8_t	SUE throttle reversed
sue_rudder_output_channel	uint8_t	SUE rudder output channel
sue_rudder_reversed	uint8_t	SUE rudder reversed

SERIAL_UDB_EXTRA_F20 (#186)

Backwards compatible version of SERIAL_UDB_EXTRA F20 format

Field Name	Type	Description
sue_number_of_inputs	uint8_t	SUE Number of Input Channels
sue_trim_value_input_1	int16_t	SUE UDB PWM Trim Value on Input 1
sue_trim_value_input_2	int16_t	SUE UDB PWM Trim Value on Input 2
sue_trim_value_input_3	int16_t	SUE UDB PWM Trim Value on Input 3
sue_trim_value_input_4	int16_t	SUE UDB PWM Trim Value on Input 4
sue_trim_value_input_5	int16_t	SUE UDB PWM Trim Value on Input 5
sue_trim_value_input_6	int16_t	SUE UDB PWM Trim Value on Input 6
sue_trim_value_input_7	int16_t	SUE UDB PWM Trim Value on Input 7
sue_trim_value_input_8	int16_t	SUE UDB PWM Trim Value on Input 8
sue_trim_value_input_9	int16_t	SUE UDB PWM Trim Value on Input 9
sue_trim_value_input_10	int16_t	SUE UDB PWM Trim Value on Input 10
sue_trim_value_input_11	int16_t	SUE UDB PWM Trim Value on Input 11
sue_trim_value_input_12	int16_t	SUE UDB PWM Trim Value on Input 12

SERIAL_UDB_EXTRA_F21 (#187)

Backwards compatible version of SERIAL_UDB_EXTRA F21 format

Field Name	Type	Description
sue_accel_x_offset	int16_t	SUE X accelerometer offset
sue_accel_y_offset	int16_t	SUE Y accelerometer offset
sue_accel_z_offset	int16_t	SUE Z accelerometer offset
sue_gyro_x_offset	int16_t	SUE X gyro offset
sue_gyro_y_offset	int16_t	SUE Y gyro offset
sue_gyro_z_offset	int16_t	SUE Z gyro offset

SERIAL_UDB_EXTRA_F22 (#188)

Backwards compatible version of SERIAL_UDB_EXTRA F22 format

Field Name	Type	Description
sue_accel_x_at_calibration	int16_t	SUE X accelerometer at calibration time
sue_accel_y_at_calibration	int16_t	SUE Y accelerometer at calibration time
sue_accel_z_at_calibration	int16_t	SUE Z accelerometer at calibration time
sue_gyro_x_at_calibration	int16_t	SUE X gyro at calibration time
sue_gyro_y_at_calibration	int16_t	SUE Y gyro at calibration time
sue_gyro_z_at_calibration	int16_t	SUE Z gyro at calibration time

MAVLINK Message Set: minimal.xml

This is a human-readable form of the XML definition file: minimal.xml.

MAVLink 2 messages have an ID > 255 and are marked up using **(MAVLink 2)** in their description.

MAVLink 2 extension fields that have been added to MAVLink 1 messages are displayed in blue.

MAVLink Protocol Version

The current MAVLink version is 2.2. The minor version numbers (after the dot) range from 1-255.

MAVLink Type Enumerations

MAV_AUTOPILOT

Micro air vehicle / autopilot classes. This identifies the individual model.

CMD ID	Field Name	Description
0	MAV_AUTOPILOT_GENERIC	Generic autopilot, full support for everything
1	MAV_AUTOPILOT_PIXHAWK	PIXHAWK autopilot, http://pixhawk.ethz.ch
2	MAV_AUTOPILOT_SLUGS	SLUGS autopilot, http://slugsuav.soe.ucs
3	MAV_AUTOPILOT_ARDUPILOTMEGA	ArduPilotMega / ArduCopter, http://diydrones.com
4	MAV_AUTOPILOT_OPENPILOT	OpenPilot, http://openpilot.org
5	MAV_AUTOPILOT_GENERIC_WAYPOINTS_ONLY	Generic autopilot only supporting simple way
6	MAV_AUTOPILOT_GENERIC_WAYPOINTS_AND_SIMPLE_NAVIGATION_ONLY	Generic autopilot supp waypoints and other si navigation commands
7	MAV_AUTOPILOT_GENERIC_MISSION_FULL	Generic autopilot supp the full mission comma set
8	MAV_AUTOPILOT_INVALID	No valid autopilot, e.g. GCS or other MAVLink component
9	MAV_AUTOPILOT_PPZ	PPZ UAV - http://nongnu.org/papa
10	MAV_AUTOPILOT_UDB	UAV Dev Board
11	MAV_AUTOPILOT_FP	FlexiPilot

MAV_TYPE

CMD ID	Field Name	Description
0	MAV_TYPE_GENERIC	Generic micro air vehicle.
1	MAV_TYPE_FIXED_WING	Fixed wing aircraft.
2	MAV_TYPE_QUADROTOR	Quadrotor
3	MAV_TYPE_COAXIAL	Coaxial helicopter
4	MAV_TYPE_HELICOPTER	Normal helicopter with tail rotor.
5	MAV_TYPE_ANTENNA_TRACKER	Ground installation
6	MAV_TYPE_GCS	Operator control unit / ground control station
7	MAV_TYPE_AIRSHIP	Airship, controlled
8	MAV_TYPE_FREE_BALLOON	Free balloon, uncontrolled
9	MAV_TYPE_ROCKET	Rocket
10	MAV_TYPE_GROUND_ROVER	Ground rover
11	MAV_TYPE_SURFACE_BOAT	Surface vessel, boat, ship
12	MAV_TYPE_SUBMARINE	Submarine
13	MAV_TYPE_HEXAROTOR	Hexarotor
14	MAV_TYPE_OCTOROTOR	Octorotor
15	MAV_TYPE_TRICOPTER	Octorotor
16	MAV_TYPE_FLAPPING_WING	Flapping wing

MAV_MODE_FLAG

These flags encode the MAV mode.

CMD ID	Field Name	Description
128	MAV_MODE_FLAG_SAFETY_ARMED	0b10000000 MAV safety set to armed. Motors are enabled / running / can start. Ready to fly.
64	MAV_MODE_FLAG_MANUAL_INPUT_ENABLED	0b01000000 remote control input is enabled.
32	MAV_MODE_FLAG_HIL_ENABLED	0b00100000 hardware in the loop simulation. All motors / actuators are blocked, but internal software is full operational.
16	MAV_MODE_FLAG_STABILIZE_ENABLED	0b00010000 system stabilizes electronically its attitude (and optionally position). It needs however further control inputs to move around.
8	MAV_MODE_FLAG_GUIDED_ENABLED	0b00001000 guided mode enabled, system flies waypoints / mission items.
4	MAV_MODE_FLAG_AUTO_ENABLED	0b00000100 autonomous mode enabled, system finds its own goal positions. Guided flag can be set or not, depends on the actual implementation.
2	MAV_MODE_FLAG_TEST_ENABLED	0b00000010 system has a test mode enabled. This flag is intended for temporary system tests and should not be used for stable implementations.
1	MAV_MODE_FLAG_CUSTOM_MODE_ENABLED	0b00000001 Reserved for future use.

MAV_MODE_FLAG_DECODE_POSITION

These values encode the bit positions of the decode position. These values can be used to read the value of a flag bit by combining the base_mode variable with AND with the flag position value. The result will be either 0 or 1, depending on if the flag is set or not.

CMD ID	Field Name	Description
128	MAV_MODE_FLAG_DECODE_POSITION_SAFETY	First bit: 10000000
64	MAV_MODE_FLAG_DECODE_POSITION_MANUAL	Second bit: 01000000
32	MAV_MODE_FLAG_DECODE_POSITION_HIL	Third bit: 00100000
16	MAV_MODE_FLAG_DECODE_POSITION_STABILIZE	Fourth bit: 00010000
8	MAV_MODE_FLAG_DECODE_POSITION_GUIDED	Fifth bit: 00001000
4	MAV_MODE_FLAG_DECODE_POSITION_AUTO	Sixt bit: 00000100
2	MAV_MODE_FLAG_DECODE_POSITION_TEST	Seventh bit: 00000010
1	MAV_MODE_FLAG_DECODE_POSITION_CUSTOM_MODE	Eighth bit: 00000001

MAV_STATE

CMD ID	Field Name	Description
0	MAV_STATE_UNINIT	Uninitialized system, state is unknown.
	MAV_STATE_BOOT	System is booting up.
	MAV_STATE_CALIBRATING	System is calibrating and not flight-ready.
	MAV_STATE_STANDBY	System is grounded and on standby. It can be launched any time.
	MAV_STATE_ACTIVE	System is active and might be already airborne. Motors are engaged.
	MAV_STATE_CRITICAL	System is in a non-normal flight mode. It can however still navigate.
	MAV_STATE_EMERGENCY	System is in a non-normal flight mode. It lost control over parts or over the whole airframe. It is in mayday and going down.
	MAV_STATE_POWEROFF	System just initialized its power-down sequence, will shut down now.

MAVLink Messages

HEARTBEAT (#0)

The heartbeat message shows that a system is present and responding. The type of the MAV and Autopilot hardware allow the receiving system to treat further messages from this system appropriate (e.g. by laying out the user interface based on the autopilot).

Field Name	Type	Description
type	uint8_t	Type of the MAV (quadrotor, helicopter, etc., up to 15 types, defined in MAV_TYPE ENUM) (Enum:MAV_TYPE)
autopilot	uint8_t	Autopilot type / class. defined in MAV_AUTOPILOT ENUM (Enum:MAV_AUTOPILOT)
base_mode	uint8_t	System mode bitfield, see MAV_MODE_FLAGS ENUM in mavlink/include/mavlink_types.h
custom_mode	uint32_t	A bitfield for use for autopilot-specific flags.
system_status	uint8_t	System status flag, see MAV_STATE ENUM (Enum:MAV_STATE)
mavlink_version	uint8_t_mavlink_version	MAVLink version

MAVLINK Message Set: paparazzi.xml

This is a human-readable form of the XML definition file: paparazzi.xml.

> MAVLink 2 messages have an ID > 255 and are marked up using **(MAVLink 2)** in their description.

> MAVLink 2 extension fields that have been added to MAVLink 1 messages are displayed in blue.

MAVLink Include Files: common.xml

MAVLink Protocol Version

The current MAVLink version is 2.3. The minor version numbers (after the dot) range from 1-255.

MAVLink Type Enumerations

MAVLink Messages

SCRIPT_ITEM (#180)

Message encoding a mission script item. This message is emitted upon a request for the next script item.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
seq	uint16_t	Sequence
name	char[50]	The name of the mission script, NULL terminated.

SCRIPT_REQUEST (#181)

Request script item with the sequence number seq. The response of the system to this message should be a SCRIPT_ITEM message.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
seq	uint16_t	Sequence

SCRIPT_REQUEST_LIST (#182)

Request the overall list of mission items from the system/component.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID

SCRIPT_COUNT (#183)

This message is emitted as response to SCRIPT_REQUEST_LIST by the MAV to get the number of mission scripts.

Field Name	Type	Description
target_system	uint8_t	System ID
target_component	uint8_t	Component ID
count	uint16_t	Number of script items in the sequence

SCRIPT_CURRENT (#184)

This message informs about the currently active SCRIPT.

Field Name	Type	Description
seq	uint16_t	Active Sequence

MAVLINK Message Set: ualberta.xml

This is a human-readable form of the XML definition file: ualberta.xml.

> MAVLink 2 messages have an ID > 255 and are marked up using **(MAVLink 2)** in their description.

> MAVLink 2 extension fields that have been added to MAVLink 1 messages are displayed in blue.

MAVLink Include Files: common.xml

MAVLink Type Enumerations

UALBERTA_AUTOPILOT_MODE

Available autopilot modes for ualberta uav

CMD ID	Field Name	Description
	MODE_MANUAL_DIRECT	Raw input pulse widts sent to output
	MODE_MANUAL_SCALED	Inputs are normalized using calibration, the converted back to raw pulse widths for output
	MODE_AUTO_PID_ATT	dfsdfs
	MODE_AUTO_PID_VEL	dfsfds
	MODE_AUTO_PID_POS	dfsdfsdfs

UALBERTA_NAV_MODE

Navigation filter mode

CMD ID	Field Name	Description
	NAV_AHRS_INIT	
	NAV_AHRS	AHRS mode
	NAV_INS_GPS_INIT	INS/GPS initialization mode
	NAV_INS_GPS	INS/GPS mode

UALBERTA_PILOT_MODE

Mode currently commanded by pilot

CMD ID	Field Name	Description
	PILOT_MANUAL	sdf
	PILOT_AUTO	dfs
	PILOT_ROTO	Rotomotion mode

MAVLink Messages

NAV_FILTER_BIAS (#220)

Accelerometer and Gyro biases from the navigation filter

Field Name	Type	Description
usec	uint64_t	Timestamp (microseconds)
accel_0	float	b_f[0]
accel_1	float	b_f[1]
accel_2	float	b_f[2]
gyro_0	float	b_f[0]
gyro_1	float	b_f[1]
gyro_2	float	b_f[2]

RADIO_CALIBRATION (#221)

Complete set of calibration parameters for the radio

Field Name	Type	Description
aileron	uint16_t[3]	Aileron setpoints: left, center, right
elevator	uint16_t[3]	Elevator setpoints: nose down, center, nose up
rudder	uint16_t[3]	Rudder setpoints: nose left, center, nose right
gyro	uint16_t[2]	Tail gyro mode/gain setpoints: heading hold, rate mode
pitch	uint16_t[5]	Pitch curve setpoints (every 25%)
throttle	uint16_t[5]	Throttle curve setpoints (every 25%)

UALBERTA_SYS_STATUS (#222)

System status specific to ualberta uav

Field Name	Type	Description
mode	uint8_t	System mode, see UALBERTA_AUTOPILOT_MODE ENUM
nav_mode	uint8_t	Navigation mode, see UALBERTA_NAV_MODE ENUM
pilot	uint8_t	Pilot mode, see UALBERTA_PILOT_MODE

MAVLINK Message Set: uAvionix.xml

This is a human-readable form of the XML definition file: uAvionix.xml.

> MAVLink 2 messages have an ID > 255 and are marked up using **(MAVLink 2)** in their description.

> MAVLink 2 extension fields that have been added to MAVLink 1 messages are displayed in blue.

MAVLink Type Enumerations

UAVIONIX_ADSB_OUT_DYNAMIC_STATE

State flags for ADS-B transponder dynamic report

CMD ID	Field Name	Description
1	UAVIONIX_ADSB_OUT_DYNAMIC_STATE_INTENT_CHANGE	
2	UAVIONIX_ADSB_OUT_DYNAMIC_STATE_AUTOPILOT_ENABLED	
4	UAVIONIX_ADSB_OUT_DYNAMIC_STATE_NICBARO_CROSSCHECKED	
8	UAVIONIX_ADSB_OUT_DYNAMIC_STATE_ON_GROUND	
16	UAVIONIX_ADSB_OUT_DYNAMIC_STATE_IDENT	

UAVIONIX_ADSB_OUT_RF_SELECT

Transceiver RF control flags for ADS-B transponder dynamic reports

CMD ID	Field Name	Description
0	UAVIONIX_ADSB_OUT_RF_SELECT_STANDBY	
1	UAVIONIX_ADSB_OUT_RF_SELECT_RX_ENABLED	
2	UAVIONIX_ADSB_OUT_RF_SELECT_TX_ENABLED	

UAVIONIX_ADSB_OUT_DYNAMIC_GPS_FIX

Status for ADS-B transponder dynamic input

CMD ID	Field Name	Description
0	UAVIONIX_ADSB_OUT_DYNAMIC_GPS_FIX_NONE_0	
1	UAVIONIX_ADSB_OUT_DYNAMIC_GPS_FIX_NONE_1	
2	UAVIONIX_ADSB_OUT_DYNAMIC_GPS_FIX_2D	
3	UAVIONIX_ADSB_OUT_DYNAMIC_GPS_FIX_3D	
4	UAVIONIX_ADSB_OUT_DYNAMIC_GPS_FIX_DGPS	
5	UAVIONIX_ADSB_OUT_DYNAMIC_GPS_FIX_RTK	

UAVIONIX_ADSB_RF_HEALTH

Status flags for ADS-B transponder dynamic output

CMD ID	Field Name	Description
0	UAVIONIX_ADSB_RF_HEALTH_INITIALIZING	
1	UAVIONIX_ADSB_RF_HEALTH_OK	
2	UAVIONIX_ADSB_RF_HEALTH_FAIL_TX	
16	UAVIONIX_ADSB_RF_HEALTH_FAIL_RX	

UAVIONIX_ADSB_OUT_CFG_AIRCRAFT_SIZE

Definitions for aircraft size

CMD ID	Field Name	Description
0	UAVIONIX_ADSB_OUT_CFG_AIRCRAFT_SIZE_NO_DATA	
1	UAVIONIX_ADSB_OUT_CFG_AIRCRAFT_SIZE_L15M_W23M	
2	UAVIONIX_ADSB_OUT_CFG_AIRCRAFT_SIZE_L25M_W28P5M	
3	UAVIONIX_ADSB_OUT_CFG_AIRCRAFT_SIZE_L25_34M	
4	UAVIONIX_ADSB_OUT_CFG_AIRCRAFT_SIZE_L35_33M	
5	UAVIONIX_ADSB_OUT_CFG_AIRCRAFT_SIZE_L35_38M	
6	UAVIONIX_ADSB_OUT_CFG_AIRCRAFT_SIZE_L45_39P5M	
7	UAVIONIX_ADSB_OUT_CFG_AIRCRAFT_SIZE_L45_45M	
8	UAVIONIX_ADSB_OUT_CFG_AIRCRAFT_SIZE_L55_45M	
9	UAVIONIX_ADSB_OUT_CFG_AIRCRAFT_SIZE_L55_52M	
10	UAVIONIX_ADSB_OUT_CFG_AIRCRAFT_SIZE_L65_59P5M	
11	UAVIONIX_ADSB_OUT_CFG_AIRCRAFT_SIZE_L65_67M	
12	UAVIONIX_ADSB_OUT_CFG_AIRCRAFT_SIZE_L75_W72P5M	
13	UAVIONIX_ADSB_OUT_CFG_AIRCRAFT_SIZE_L75_W80M	
14	UAVIONIX_ADSB_OUT_CFG_AIRCRAFT_SIZE_L85_W80M	
15	UAVIONIX_ADSB_OUT_CFG_AIRCRAFT_SIZE_L85_W90M	

UAVIONIX_ADSB_OUT_CFG_GPS_OFFSET_LAT

GPS lataral offset encoding

CMD ID	Field Name	Description
0	UAVIONIX_ADSB_OUT_CFG_GPS_OFFSET_LAT_NO_DATA	
1	UAVIONIX_ADSB_OUT_CFG_GPS_OFFSET_LAT_LEFT_2M	
2	UAVIONIX_ADSB_OUT_CFG_GPS_OFFSET_LAT_LEFT_4M	
3	UAVIONIX_ADSB_OUT_CFG_GPS_OFFSET_LAT_LEFT_6M	
4	UAVIONIX_ADSB_OUT_CFG_GPS_OFFSET_LAT_RIGHT_0M	
5	UAVIONIX_ADSB_OUT_CFG_GPS_OFFSET_LAT_RIGHT_2M	
6	UAVIONIX_ADSB_OUT_CFG_GPS_OFFSET_LAT_RIGHT_4M	
7	UAVIONIX_ADSB_OUT_CFG_GPS_OFFSET_LAT_RIGHT_6M	

UAVIONIX_ADSB_OUT_CFG_GPS_OFFSET_LON

GPS longitudinal offset encoding

CMD ID	Field Name	Description
0	UAVIONIX_ADSB_OUT_CFG_GPS_OFFSET_LON_NO_DATA	
1	UAVIONIX_ADSB_OUT_CFG_GPS_OFFSET_LON_APPLIED_BY_SENSOR	

UAVIONIX_ADSB_EMERGENCY_STATUS

Emergency status encoding

CMD ID	Field Name	Description
0	UAVIONIX_ADSB_OUT_NO_EMERGENCY	
1	UAVIONIX_ADSB_OUT_GENERAL_EMERGENCY	
2	UAVIONIX_ADSB_OUT_LIFEGUARD_EMERGENCY	
3	UAVIONIX_ADSB_OUT_MINIMUM_FUEL_EMERGENCY	
4	UAVIONIX_ADSB_OUT_NO_COMM_EMERGENCY	
5	UAVIONIX_ADSB_OUT_UNLAWFUL_INTERFERANCE_EMERGENCY	
6	UAVIONIX_ADSB_OUT_DOWNED_AIRCRAFT_EMERGENCY	
7	UAVIONIX_ADSB_OUT_RESERVED	

MAVLink Messages

UAVIONIX_ADSB_OUT_CFG (#10001)

(MAVLink 2) Static data to configure the ADS-B transponder (send within 10 sec of a POR and every 10 sec thereafter)

Field Name	Type	Description
ICAO	uint32_t	Vehicle address (24 bit)
callsign	char[9]	Vehicle identifier (8 characters, null terminated, valid characters are A-Z, 0-9, " " only)
emitterType	uint8_t	Transmitting vehicle type. See ADSB_EMITTER_TYPE enum (Enum:ADSB_EMITTER_TYPE)
aircraftSize	uint8_t	Aircraft length and width encoding (table 2-35 of DO-282B) (Enum:UAVIONIX_ADSB_OUT_CFG_AIRCRAFT_SIZE)
gpsOffsetLat	uint8_t	GPS antenna lateral offset (table 2-36 of DO-282B) (Enum:UAVIONIX_ADSB_OUT_CFG_GPS_OFFSET_LAT)
gpsOffsetLon	uint8_t	GPS antenna longitudinal offset from nose [if non-zero, take position (in meters) divide by 2 and add one] (table 2-37 DO-282B) (Enum:UAVIONIX_ADSB_OUT_CFG_GPS_OFFSET_LON)
stallSpeed	uint16_t	Aircraft stall speed in cm/s (Units: cm/s)
rfSelect	uint8_t	ADS-B transponder reciever and transmit enable flags (Enum:UAVIONIX_ADSB_OUT_RF_SELECT)

UAVIONIX_ADSB_OUT_DYNAMIC (#10002)

(MAVLink 2) Dynamic data used to generate ADS-B out transponder data (send at 5Hz)

Field Name	Type	Description
utcTime	uint32_t	UTC time in seconds since GPS epoch (Jan 6, 1980). If unknown set to UINT32_MAX (Units: s)
gpsLat	int32_t	Latitude WGS84 (deg * 1E7). If unknown set to INT32_MAX (Units: degE7)
gpsLon	int32_t	Longitude WGS84 (deg * 1E7). If unknown set to INT32_MAX (Units: degE7)
gpsAlt	int32_t	Altitude in mm (m * 1E-3) UP +ve. WGS84 altitude. If unknown set to INT32_MAX (Units: mm)
gpsFix	uint8_t	0-1: no fix, 2: 2D fix, 3: 3D fix, 4: DGPS, 5: RTK (Enum:UAVIONIX_ADSB_OUT_DYNAMIC_GPS_FIX)
numSats	uint8_t	Number of satellites visible. If unknown set to UINT8_MAX
baroAltMSL	int32_t	Barometric pressure altitude relative to a standard atmosphere of 1013.2 mBar and NOT bar corrected altitude (m * 1E-3). (up +ve). If unknown set to INT32_MAX (Units: mbar)
accuracyHor	uint32_t	Horizontal accuracy in mm (m * 1E-3). If unknown set to UINT32_MAX (Units: mm)
accuracyVert	uint16_t	Vertical accuracy in cm. If unknown set to UINT16_MAX (Units: cm)
accuracyVel	uint16_t	Velocity accuracy in mm/s (m * 1E-3). If unknown set to UINT16_MAX (Units: mm/s)
velVert	int16_t	GPS vertical speed in cm/s. If unknown set to INT16_MAX (Units: cm/s)
velNS	int16_t	North-South velocity over ground in cm/s North +ve. If unknown set to INT16_MAX (Units: cm/s)
VelEW	int16_t	East-West velocity over ground in cm/s East +ve. If unknown set to INT16_MAX (Units: cm/s)
emergencyStatus	uint8_t	Emergency status (Enum:UAVIONIX_ADSB_EMERGENCY_STATUS)
state	uint16_t	ADS-B transponder dynamic input state flags (Enum:UAVIONIX_ADSB_OUT_DYNAMIC_STATE)
squawk	uint16_t	Mode A code (typically 1200 [0x04B0] for VFR)

UAVIONIX_ADSB_TRANSCEIVER_HEALTH_REPORT (#10003)

(MAVLink 2) Transceiver heartbeat with health report (updated every 10s)

Field Name	Type	Description
rfHealth	uint8_t	ADS-B transponder messages (Enum:UAVIONIX_ADSB_RF_HEALTH)

MAVLINK Message Set: icarous.xml

This is a human-readable form of the XML definition file: icarous.xml.

MAVLink 2 messages have an ID > 255 and are marked up using **(MAVLink 2)** in their description.

MAVLink 2 extension fields that have been added to MAVLink 1 messages are displayed in blue.

MAVLink Type Enumerations

ICAROUS_TRACK_BAND_TYPES

CMD ID	Field Name	Description
0	ICAROUS_TRACK_BAND_TYPE_NONE	
1	ICAROUS_TRACK_BAND_TYPE_NEAR	
2	ICAROUS_TRACK_BAND_TYPE_RECOVERY	

ICAROUS_FMS_STATE

CMD ID	Field Name	Description
0	ICAROUS_FMS_STATE_IDLE	
1	ICAROUS_FMS_STATE_TAKEOFF	
2	ICAROUS_FMS_STATE_CLIMB	
3	ICAROUS_FMS_STATE_CRUISE	
4	ICAROUS_FMS_STATE_APPROACH	
5	ICAROUS_FMS_STATE_LAND	

MAVLink Messages

ICAROUS_HEARTBEAT (#42000)

(MAVLink 2) ICAROUS heartbeat

Field Name	Type	Description
status	uint8_t	See the FMS_STATE enum. (Enum:ICAROUS_FMS_STATE)

ICAROUS_KINEMATIC_BANDS (#42001)

(MAVLink 2) Kinematic multi bands (track) output from Daidalus

Field Name	Type	Description
numBands	int8_t	Number of track bands
type1	uint8_t	See the TRACK_BAND_TYPES enum. (Enum:ICAROUS_TRACK_BAND_TYPES)
min1	float	min angle (degrees) (Units: deg)
max1	float	max angle (degrees) (Units: deg)
type2	uint8_t	See the TRACK_BAND_TYPES enum. (Enum:ICAROUS_TRACK_BAND_TYPES)
min2	float	min angle (degrees) (Units: deg)
max2	float	max angle (degrees) (Units: deg)
type3	uint8_t	See the TRACK_BAND_TYPES enum. (Enum:ICAROUS_TRACK_BAND_TYPES)
min3	float	min angle (degrees) (Units: deg)
max3	float	max angle (degrees) (Units: deg)
type4	uint8_t	See the TRACK_BAND_TYPES enum. (Enum:ICAROUS_TRACK_BAND_TYPES)
min4	float	min angle (degrees) (Units: deg)
max4	float	max angle (degrees) (Units: deg)
type5	uint8_t	See the TRACK_BAND_TYPES enum. (Enum:ICAROUS_TRACK_BAND_TYPES)
min5	float	min angle (degrees) (Units: deg)
max5	float	max angle (degrees) (Units: deg)

MAVLINK Message Set: standard.xml

This is a human-readable form of the XML definition file: standard.xml.

> MAVLink 2 messages have an ID > 255 and are marked up using **(MAVLink 2)** in their description.

> MAVLink 2 extension fields that have been added to MAVLink 1 messages are displayed in blue.

MAVLink Include Files: common.xml

This file has protocol dialect: 0.

MAVLink Type Enumerations

MAVLink Messages

MAVLINK Message Set: test.xml

This is a human-readable form of the XML definition file: test.xml.

> MAVLink 2 messages have an ID > 255 and are marked up using **(MAVLink 2)** in their description.

> MAVLink 2 extension fields that have been added to MAVLink 1 messages are displayed in blue.

MAVLink Protocol Version

The current MAVLink version is 2.3. The minor version numbers (after the dot) range from 1-255.

MAVLink Messages

TEST_TYPES (#0)

Test all field types

Field Name	Type	Description
c	char	char
s	char[10]	string
u8	uint8_t	uint8_t
u16	uint16_t	uint16_t
u32	uint32_t	uint32_t
u64	uint64_t	uint64_t
s8	int8_t	int8_t
s16	int16_t	int16_t
s32	int32_t	int32_t
s64	int64_t	int64_t
f	float	float
d	double	double
u8_array	uint8_t[3]	uint8_t_array
u16_array	uint16_t[3]	uint16_t_array
u32_array	uint32_t[3]	uint32_t_array
u64_array	uint64_t[3]	uint64_t_array
s8_array	int8_t[3]	int8_t_array
s16_array	int16_t[3]	int16_t_array
s32_array	int32_t[3]	int32_t_array
s64_array	int64_t[3]	int64_t_array
f_array	float[3]	float_array
d_array	double[3]	double_array

MAVLINK Message Set: python_array_test.xml

This is a human-readable form of the XML definition file: python_array_test.xml.

MAVLink 2 messages have an ID > 255 and are marked up using **(MAVLink 2)** in their description.

MAVLink 2 extension fields that have been added to MAVLink 1 messages are displayed in blue.

MAVLink Include Files: common.xml

MAVLink Messages

ARRAY_TEST_0 (#150)

Array test #0.

Field Name	Type	Description
v1	uint8_t	Stub field
ar_i8	int8_t[4]	Value array
ar_u8	uint8_t[4]	Value array
ar_u16	uint16_t[4]	Value array
ar_u32	uint32_t[4]	Value array

ARRAY_TEST_1 (#151)

Array test #1.

Field Name	Type	Description
ar_u32	uint32_t[4]	Value array

ARRAY_TEST_3 (#153)

Array test #3.

Field Name	Type	Description
v	uint8_t	Stub field
ar_u32	uint32_t[4]	Value array

ARRAY_TEST_4 (#154)

Array test #4.

Field Name	Type	Description
ar_u32	uint32_t[4]	Value array
v	uint8_t	Stub field

ARRAY_TEST_5 (#155)

Array test #5.

Field Name	Type	Description
c1	char[5]	Value array
c2	char[5]	Value array

ARRAY_TEST_6 (#156)

Array test #6.

Field Name	Type	Description
v1	uint8_t	Stub field
v2	uint16_t	Stub field
v3	uint32_t	Stub field
ar_u32	uint32_t[2]	Value array
ar_i32	int32_t[2]	Value array
ar_u16	uint16_t[2]	Value array
ar_i16	int16_t[2]	Value array
ar_u8	uint8_t[2]	Value array
ar_i8	int8_t[2]	Value array
ar_c	char[32]	Value array
ar_d	double[2]	Value array
ar_f	float[2]	Value array

ARRAY_TEST_7 (#157)

Array test #7.

Field Name	Type	Description
ar_d	double[2]	Value array
ar_f	float[2]	Value array
ar_u32	uint32_t[2]	Value array
ar_i32	int32_t[2]	Value array
ar_u16	uint16_t[2]	Value array
ar_i16	int16_t[2]	Value array
ar_u8	uint8_t[2]	Value array
ar_i8	int8_t[2]	Value array
ar_c	char[32]	Value array

ARRAY_TEST_8 (#158)

Array test #8.

Field Name	Type	Description
v3	uint32_t	Stub field
ar_d	double[2]	Value array
ar_u16	uint16_t[2]	Value array

MAVLINK Message Set: slugs.xml

This is a human-readable form of the XML definition file: slugs.xml.

MAVLink 2 messages have an ID > 255 and are marked up using **(MAVLink 2)** in their description.

MAVLink 2 extension fields that have been added to MAVLink 1 messages are displayed in blue.

MAVLink Include Files: common.xml

MAVLink Type Enumerations

MAV_CMD

CMD ID	Field Name	Description
10001	MAV_CMD_DO_NOTHING	Does nothing.
	Mission Param #1	1 to arm, 0 to disarm
10011	MAV_CMD_RETURN_TO_BASE	Return vehicle to base.
	Mission Param #1	0: return to base, 1: track mobile base
10012	MAV_CMD_STOP_RETURN_TO_BASE	Stops the vehicle from returning to base and resumes flight.
10013	MAV_CMD_TURN_LIGHT	Turns the vehicle's visible or infrared lights on or off.
	Mission Param #1	0: visible lights, 1: infrared lights
	Mission Param #2	0: turn on, 1: turn off
10014	MAV_CMD_GET_MID_LEVEL_COMMANDS	Requests vehicle to send current mid-level commands to ground station.
10015	MAV_CMD_MIDLEVEL_STORAGE	Requests storage of mid-level commands.
	Mission Param #1	Mid-level command storage: 0: read from flash/EEPROM, 1: write to flash/EEPROM

SLUGS_MODE

Slugs-specific navigation modes.

CMD ID	Field Name	Description
0	SLUGS_MODE_NONE	No change to SLUGS mode.
1	SLUGS_MODE_LIFTOFF	Vehicle is in liftoff mode.
2	SLUGS_MODE_PASSTHROUGH	Vehicle is in passthrough mode, being controlled by a pilot.
3	SLUGS_MODE_WAYPOINT	Vehicle is in waypoint mode, navigating to waypoints.
4	SLUGS_MODE_MID_LEVEL	Vehicle is executing mid-level commands.
5	SLUGS_MODE_RETURNING	Vehicle is returning to the home location.
6	SLUGS_MODE_LANDING	Vehicle is landing.
7	SLUGS_MODE_LOST	Lost connection with vehicle.
8	SLUGS_MODE_SELECTIVE_PASSTHROUGH	Vehicle is in selective passthrough mode, where selected surfaces are being manually controlled.
9	SLUGS_MODE_ISR	Vehicle is in ISR mode, performing reconaissance at a point specified by ISR_LOCATION message.
10	SLUGS_MODE_LINE_PATROL	Vehicle is patrolling along lines between waypoints.
11	SLUGS_MODE_GROUNDED	Vehicle is grounded or an error has occurred.

CONTROL_SURFACE_FLAG

These flags encode the control surfaces for selective passthrough mode. If a bit is set then the pilot console has control of the surface, and if not then the autopilot has control of the surface.

CMD ID	Field Name	Description
128	CONTROL_SURFACE_FLAG_THROTTLE	0b10000000 Throttle control passes through to pilot console.
64	CONTROL_SURFACE_FLAG_LEFT_AILERON	0b01000000 Left aileron control passes through to pilot console.
32	CONTROL_SURFACE_FLAG_RIGHT_AILERON	0b00100000 Right aileron control passes through to pilot console.
16	CONTROL_SURFACE_FLAG_RUDDER	0b00010000 Rudder control passes through to pilot console.
8	CONTROL_SURFACE_FLAG_LEFT_ELEVATOR	0b00001000 Left elevator control passes through to pilot console.
4	CONTROL_SURFACE_FLAG_RIGHT_ELEVATOR	0b00000100 Right elevator control passes through to pilot console.
2	CONTROL_SURFACE_FLAG_LEFT_FLAP	0b00000010 Left flap control passes through to pilot console.
1	CONTROL_SURFACE_FLAG_RIGHT_FLAP	0b00000001 Right flap control passes through to pilot console.

MAVLink Messages

CPU_LOAD (#170)

Sensor and DSC control loads.

Field Name	Type	Description
sensLoad	uint8_t	Sensor DSC Load
ctrlLoad	uint8_t	Control DSC Load
batVolt	uint16_t	Battery Voltage in millivolts

SENSOR_BIAS (#172)

Accelerometer and gyro biases.

Field Name	Type	Description
axBias	float	Accelerometer X bias (m/s)
ayBias	float	Accelerometer Y bias (m/s)
azBias	float	Accelerometer Z bias (m/s)
gxBias	float	Gyro X bias (rad/s)
gyBias	float	Gyro Y bias (rad/s)
gzBias	float	Gyro Z bias (rad/s)

DIAGNOSTIC (#173)

Configurable diagnostic messages.

Field Name	Type	Description
diagFl1	float	Diagnostic float 1
diagFl2	float	Diagnostic float 2
diagFl3	float	Diagnostic float 3
diagSh1	int16_t	Diagnostic short 1
diagSh2	int16_t	Diagnostic short 2
diagSh3	int16_t	Diagnostic short 3

SLUGS_NAVIGATION (#176)

Data used in the navigation algorithm.

Field Name	Type	Description
u_m	float	Measured Airspeed prior to the nav filter in m/s
phi_c	float	Commanded Roll
theta_c	float	Commanded Pitch
psiDot_c	float	Commanded Turn rate
ay_body	float	Y component of the body acceleration
totalDist	float	Total Distance to Run on this leg of Navigation
dist2Go	float	Remaining distance to Run on this leg of Navigation
fromWP	uint8_t	Origin WP
toWP	uint8_t	Destination WP
h_c	uint16_t	Commanded altitude in 0.1 m

DATA_LOG (#177)

Configurable data log probes to be used inside Simulink

Field Name	Type	Description
fl_1	float	Log value 1
fl_2	float	Log value 2
fl_3	float	Log value 3
fl_4	float	Log value 4
fl_5	float	Log value 5
fl_6	float	Log value 6

GPS_DATE_TIME (#179)

Pilot console PWM messges.

Field Name	Type	Description
year	uint8_t	Year reported by Gps
month	uint8_t	Month reported by Gps
day	uint8_t	Day reported by Gps
hour	uint8_t	Hour reported by Gps
min	uint8_t	Min reported by Gps
sec	uint8_t	Sec reported by Gps
clockStat	uint8_t	Clock Status. See table 47 page 211 OEMStar Manual
visSat	uint8_t	Visible satellites reported by Gps
useSat	uint8_t	Used satellites in Solution
GppGl	uint8_t	GPS+GLONASS satellites in Solution
sigUsedMask	uint8_t	GPS and GLONASS usage mask (bit 0 GPS_used? bit_4 GLONASS_used?)
percentUsed	uint8_t	Percent used GPS

MID_LVL_CMDS (#180)

Mid Level commands sent from the GS to the autopilot. These are only sent when being operated in mid-level commands mode from the ground.

Field Name	Type	Description
target	uint8_t	The system setting the commands
hCommand	float	Commanded Altitude in meters
uCommand	float	Commanded Airspeed in m/s
rCommand	float	Commanded Turnrate in rad/s

CTRL_SRFC_PT (#181)

This message sets the control surfaces for selective passthrough mode.

Field Name	Type	Description
target	uint8_t	The system setting the commands
bitfieldPt	uint16_t	Bitfield containing the passthrough configuration, see CONTROL_SURFACE_FLAG ENUM.

SLUGS_CAMERA_ORDER (#184)

Orders generated to the SLUGS camera mount.

Field Name	Type	Description
target	uint8_t	The system reporting the action
pan	int8_t	Order the mount to pan: -1 left, 0 No pan motion, +1 right
tilt	int8_t	Order the mount to tilt: -1 down, 0 No tilt motion, +1 up
zoom	int8_t	Order the zoom values 0 to 10
moveHome	int8_t	Orders the camera mount to move home. The other fields are ignored when this field is set. 1: move home, 0 ignored

CONTROL_SURFACE (#185)

Control for surface; pending and order to origin.

Field Name	Type	Description
target	uint8_t	The system setting the commands
idSurface	uint8_t	ID control surface send 0: throttle 1: aileron 2: elevator 3: rudder
mControl	float	Pending
bControl	float	Order to origin

SLUGS_MOBILE_LOCATION (#186)

Transmits the last known position of the mobile GS to the UAV. Very relevant when Track Mobile is enabled

Field Name	Type	Description
target	uint8_t	The system reporting the action
latitude	float	Mobile Latitude
longitude	float	Mobile Longitude

SLUGS_CONFIGURATION_CAMERA (#188)

Control for camara.

Field Name	Type	Description
target	uint8_t	The system setting the commands
idOrder	uint8_t	ID 0: brightness 1: aperture 2: iris 3: ICR 4: backlight
order	uint8_t	1: up/on 2: down/off 3: auto/reset/no action

ISR_LOCATION (#189)

Transmits the position of watch

Field Name	Type	Description
target	uint8_t	The system reporting the action
latitude	float	ISR Latitude
longitude	float	ISR Longitude
height	float	ISR Height
option1	uint8_t	Option 1
option2	uint8_t	Option 2
option3	uint8_t	Option 3

VOLT_SENSOR (#191)

Transmits the readings from the voltage and current sensors

Field Name	Type	Description
r2Type	uint8_t	It is the value of reading 2: 0 - Current, 1 - Foreward Sonar, 2 - Back Sonar, 3 - RPM
voltage	uint16_t	Voltage in uS of PWM. 0 uS = 0V, 20 uS = 21.5V
reading2	uint16_t	Depends on the value of r2Type (0) Current consumption in uS of PWM, 20 uS = 90Amp (1) Distance in cm (2) Distance in cm (3) Absolute value

PTZ_STATUS (#192)

Transmits the actual Pan, Tilt and Zoom values of the camera unit

Field Name	Type	Description
zoom	uint8_t	The actual Zoom Value
pan	int16_t	The Pan value in 10ths of degree
tilt	int16_t	The Tilt value in 10ths of degree

UAV_STATUS (#193)

Transmits the actual status values UAV in flight

Field Name	Type	Description
target	uint8_t	The ID system reporting the action
latitude	float	Latitude UAV
longitude	float	Longitude UAV
altitude	float	Altitude UAV
speed	float	Speed UAV
course	float	Course UAV

STATUS_GPS (#194)

This contains the status of the GPS readings

Field Name	Type	Description
csFails	uint16_t	Number of times checksum has failed
gpsQuality	uint8_t	The quality indicator, 0=fix not available or invalid, 1=GPS fix, 2=C/A differential GPS, 6=Dead reckoning mode, 7=Manual input mode (fixed position), 8=Simulator mode, 9= WAAS a
msgsType	uint8_t	Indicates if GN, GL or GP messages are being received
posStatus	uint8_t	A = data valid, V = data invalid
magVar	float	Magnetic variation, degrees
magDir	int8_t	Magnetic variation direction E/W. Easterly variation (E) subtracts from True course and Westerly variation (W) adds to True course
modeInd	uint8_t	Positioning system mode indicator. A - Autonomous;D-Differential; E-Estimated (dead reckoning) mode;M-Manual input; N-Data not valid

NOVATEL_DIAG (#195)

Transmits the diagnostics data from the Novatel OEMStar GPS

Field Name	Type	Description
timeStatus	uint8_t	The Time Status. See Table 8 page 27 Novatel OEMStar Manual
receiverStatus	uint32_t	Status Bitfield. See table 69 page 350 Novatel OEMstar Manual
solStatus	uint8_t	solution Status. See table 44 page 197
posType	uint8_t	position type. See table 43 page 196
velType	uint8_t	velocity type. See table 43 page 196
posSolAge	float	Age of the position solution in seconds
csFails	uint16_t	Times the CRC has failed since boot

SENSOR_DIAG (#196)

Diagnostic data Sensor MCU

Field Name	Type	Description
float1	float	Float field 1
float2	float	Float field 2
int1	int16_t	Int 16 field 1
char1	int8_t	Int 8 field 1

BOOT (#197)

The boot message indicates that a system is starting. The onboard software version allows to keep track of onboard soft/firmware revisions. This message allows the sensor and control MCUs to communicate version numbers on startup.

Field Name	Type	Description
version	uint32_t	The onboard software version

Contributing to MAVLink

We follow the Github flow development model. Contributions fall into two main categories: Design and micro-service changes include new features that come with a state machine and message specifications for a new type of interface (examples: parameter or mission protocol). These are major contributions requiring a lot of vetting and should come with a RFC pull request in https://github.com/mavlink/rfcs. Protocol specification and documentation changes are usually changes with less impact and can be directly raised as pull requests against this repository.

Below we explain the processes for contributing to each category and how to raise a pull request.

How to Contribute Design and Micro-service Changes

- Open a pull request against the RFC repository containing a new RFC number https://github.com/mavlink/rfcs and use the template in the 0000 RFC.
- Reach out to the community on Slack and on http://discuss.dronecode.org to raise awareness
- Address concerns by pushing more commits to the pull request

How to Contribute Protocol Specification Changes

- Open a pull request against the specification repository: https://github.com/mavlink/mavlink
- Reach out to the community on Slack and on http://discuss.dronecode.org to raise awareness
- Address concerns by pushing more commits to the pull request

How to Open a Pull Request

1. First fork and clone the project project.
2. Create a feature branch off master

   ```
   git checkout -b mydescriptivebranchname
   ```

   > *Always* branch off master for new features.

3. Commit your changes with a descriptive commit message.
   - Include context information, what was fixed, and an issue number (Github will link these then conveniently)
   - **Example:**

     ```
     Change the attitude output spec documentation

     - Fixes a typo
     - Clarifies that units are radians

     Fixes issue #123
     ```

4. Test your changes (we may ask you for test results in your PR).
5. Push changes to your repo:

   ```
   git push origin mydescriptivebranchname
   ```

6. Send a pull request to merge changes in the branch.

Support

Weekly Dev Call

MAVLink developers, adopting companies and the surrounding community of users meet weekly to help define the direction of the project, discuss RFCs, Issues and have a Q&A session.

Who should attend:

- Core project maintainers
- Component maintainers
- Dronecode members
- Community members
- Anyone interested in the development of MAVLink

> The dev call is open to all interested developers (not just the dev team). This is a great opportunity to meet the team and contribute to the ongoing development of the project.

Schedule

- **TIME**: Wednesday 6:15PM CET, 12:15AM EST, 9:15AM PST (subscribe to calendar)
- **Join the call**: https://zoom.us/j/579545140
- **Meeting ID**: 625 711 763
- **Dial(for higher quality, dial a number based on your current location)**:
  - **Switzerland**: +41 (0) 31 528 0988
  - **US**: +1 646 876 9923 or +1 669 900 6833 or +1 408 740 3766
  - **Germany**: +49 (0) 30 3080 6188
  - **Mexico**: +52 554 161 4288
  - **Australia**: +61 (0) 2 8015 2088
  - **United Kingdom**: +44 (0) 20 3695 0088
  - **South Korea**: +82 (0) 2 6022 2322
  - **Spain**: +34 91 198 0188
  - **International numbers available**
- **Nominate Issues and PRs for the call with the Dev Call label on Github**