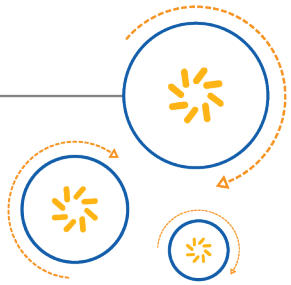
Qualcomm Technologies, Inc.



Qualcomm® Snapdragon Navigator™

User Guide

80-P4698-18 A

June 16, 2017

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

Qualcomm® Snapdragon Navigator™ is a flight stack containing a high performance flight controller with related modules and tools. Snapdragon Navigator enables aggressive and stable flight.

Snapdragon Navigator includes safety features such as propeller obstruction detection, low voltage forced landing, and intelligent mode transition in case of sensor loss.

Snapdragon Navigator is a complex system and requires understanding flight modes, user input, and status LED codes.

This document describes Snapdragon Navigator files, how to install the install the files onto a target, run Snapdragon Navigator, and operate a vehicle using Snapdragon Navigator.

This document assumes that the target has the correct metabuild and that the reader has a basic knowledge of UNIX.

1.2 Conventions

Function declarations, function names, type declarations, and code samples appear in a different font, for example, `#include`.

Code variables appear in angle brackets, for example, `<number>`.

Commands and command variables appear in a different font, for example, **`copy a:*.* b:`**.

Snapdragon Navigator-configurable parameters are formatted as ***snapdragon_navigator_parameters***.

2 Getting Started

This chapter provides instructions for getting started with Snapdragon Navigator on the Snapdragon Flight APQ8x74 board.

2.1 Licensing

Snapdragon Navigator requires a valid license file before starting up. Copy the license provided to the following directory on the vehicle:

```
/opt/qcom-licenses/
```

Without a valid license, Snapdragon Navigator does not initialize and reports a licensing error.

2.2 Uninstalling Snapdragon Navigator

Before installing Snapdragon Navigator, remove any existing installations. Back up the existing custom configuration or parameter files because they might be overwritten upon uninstallation or a new installation. If installing Snapdragon Navigator on a clean board, skip this section and see Section 2.4.1.

Purge any existing Snapdragon Navigator Debian package with the following command:

```
$ sudo dpkg -P <name>
```

This command removes all existing Snapdragon Navigator files, including configurations.

2.3 Installing Machine Vision Library

The Machine Vision library `libmv1.so` must be installed before installing Snapdragon Navigator. This library is a requirement for Snapdragon Navigator and is installed with an associated package. Without this package, the Snapdragon Navigator installation will fail.

Copy the Machine Vision Debian package to the vehicle and install it with the following command:

```
$ sudo dpkg -i /path/to/deb/<mv_deb_package_name>.deb
```

The Machine Vision library `libmv1.so` installs to `/usr/lib/`.

2.4 Installing Snapdragon Navigator

Ensure the following before installing Snapdragon Navigator:

- The target device has been prepared with the latest platform software and Flight Controller Add-On.
- The Machine Vision library has been successfully installed.

2.4.1 Package

This release includes the following packages:

- **snav_<version>_8x74.deb** – Snapdragon Navigator binaries, configuration files, and API header files
- **snav-oem_<version>_8x74.deb** – Snapdragon Navigator binaries, configuration files, and API header files intended for product development by manufacturers
- **snav-oem-release_<version>_8x74.deb** – Snapdragon Navigator binaries intended for product releases by manufacturers

Note: Only one package should be installed.

1. Choose the Debian package to be installed.
   - For the basic install, choose **snav_<version>_8x74.deb**
2. Copy the selected package to a directory on the applications processor, such as `/home/linaro`.

   Note: The `scp` and `adb` tools can be used for file transfers. Because `adb` typically requires a USB connection to the target device, use `scp` for wireless transfers.
3. To ensure that the contents are written to disk and persist across a power cycle, call `sync` from the target console.
4. Once the package has been copied to the target, run the following command from the applications processor console to install the package:

   ```
   $ sudo dpkg -i /path/to/deb/<deb_package_name>.deb
   ```
5. Respond to any prompts asking whether to overwrite conflicting symlinks. By default, the installation does not overwrite existing symlinks. If no conflicting symlinks are found, the installation completes without any prompts.

Note: The installation is based on the small sample drone example. For this software to be used with other vehicles, runtime parameter files must be changed. Runtime parameters are described in Chapter 8.

2.4.2 Installation Overview

The Snapdragon Navigator software is comprised of processes that run on both the applications processor and the DSP. The Snapdragon Navigator package installs a number of libraries, executables, and configuration files to the target.

2.4.2.1 Applications Processor

Shared libraries are installed to `/usr/lib`, and executables are installed to `/usr/bin`. Additional configuration files for the applications processor are installed to `/etc/snav`.

Scripts related to autostarting Snapdragon Navigator software are installed to `/etc/init`.

2.4.2.2 DSP

Binary and configuration files related to software that runs on the DSP is installed to `/usr/share/data/adsp`.

2.4.3 Configuring Runtime Parameters

Note: When installing for the first time, you must define `/usr/share/data/adsp/snav_params.xml`, either as a symlink pointing to a file containing the vehicle-specific runtime parameters or as a file containing the vehicle-specific runtime parameters. Once defined, it persists across installations. For the small sample drone example, create a symlink called `snav_params.xml` pointing to `200qc_runtime_params.xml`. **Snapdragon Navigator will not start unless all required parameters are specified in this file or are contained in files included from this file.**

Caution: It is crucial to use the correct parameter files for the vehicle that you are using. The default parameters installed by the Debian package are for the small sample drone example. **These parameters must be changed for any other vehicle.** Do not change these parameters without understanding their purpose, because incorrect parameters might cause the vehicle to crash.

2.5 Running Snapdragon Navigator

After completing the package installation and configuring the correct runtime parameters, Snapdragon Navigator can be started with the following command:

```
$ sudo start snav
```

This command also attempts to start any other processes that are configured to autostart. By default, Snapdragon Navigator is configured to autostart with Optic Flow (DFT) and Qualcomm DroneController companion processes.

Caution: If a runtime error occurs when starting `snav`, make sure that the correct platform image is installed on the applications processor. Snapdragon Navigator depends on some libraries that are included in the platform release or Flight Controller Add-On. It is crucial that the correct version of these libraries are installed.

Note: Some actions are required before running Snapdragon Navigator the first time. See Section 6.5.

Once Snapdragon Navigator is running, it can be stopped using the following command:

```
$ sudo stop snav
```

Note: The stop command does not work during flight.

2.5.1 Autostart Capability

The installation procedure configures Snapdragon Navigator to start automatically as part of the boot process using upstart jobs. Upstart jobs are defined in `/etc/init/` and have a `.conf` extension.

These scripts are run during the boot process and can be used to launch the intended applications. Console output from the upstart jobs gets stored in `/var/log/upstart`, where each `name.log` corresponds to the output from `name.conf`.

A detailed explanation of upstart jobs is beyond the scope of this document. Information is available online, such as http://upstart.ubuntu.com/getting-started.html.

2.5.1.1 Disabling Autostart

To stop a process from launching automatically at boot, issue a command to create a file that overrides the upstart script. For example, to prevent Snapdragon Navigator from launching automatically during the next boot, run the following command:

```
$ sudo echo "manual" > /etc/init/snav.override
```

To enable the Snapdragon Navigator upstart again, remove the `.override` file:

```
$ sudo rm /etc/init/snav.override
```

2.5.2 Manually Starting Snapdragon Navigator

Snapdragon Navigator can also be started manually without using the autostart scripts, but this is not recommended for basic use. To start the Snapdragon Navigator core functionality with default logging enabled, execute `snav` on the applications processor as root.

```
$ sudo snav
```

Run the following command to view `snav` usage:

```
$ snav -h
```

2.6 Interacting with Snapdragon Navigator

Snapdragon Navigator supports user interaction through the following:

- Spektrum Radio Controller (RC) (commonly referred to as a *transmitter* or *radio*)
  - DX8 is commonly used with Snapdragon Navigator
  - DX6i, DX7s, and DX9 have also been tested
- Wi-Fi control through Qualcomm DroneController Android application
- Snapdragon Navigator API (see *Qualcomm Snapdragon Navigator Developer Guide* (80-P4698-20) for details)

For development purposes, the tablet/API and RC and can be used simultaneously to provide a backup to the Wi-Fi control. Snapdragon Navigator searches for the tablet/API and RC by default. If both are present, channel 7 of the Spektrum RC determines which device is to be used as follows:

- **CH7 LOW** corresponds to Spektrum RC control
- **CH7 HIGH** corresponds to tablet/API control

If you are using a DX6i, which does not have seven channels, the Spektrum RC always overrides the tablet/API.

2.6.1 Setting Up an Android Tablet

DroneController can be used to control a vehicle running Snapdragon Navigator. This application has virtual joysticks that mimic the joysticks of a traditional RC.

Note: DroneController can only be used in Position Hold mode (see Section 4.1) and cannot access many features accessible through a Spektrum RC, such as calibration routines (see Section 6).

1. Obtain a copy of the DroneController application and flash it onto an Android tablet.
2. Connect to the drone's access point and open DroneController.
3. In settings, set the drone IP address and UDP port appropriately (default UDP port is 14556).

Refer to https://github.com/ATLFlight/drone-controller for more details about DroneController.

2.6.1.1 Snapdragon Navigator DroneController Companion

To use Snapdragon Navigator with the Qualcomm DroneController Android application, `snav_dronecontroller_companion` must also be running. This executable receives control packets from DroneController over Wi-Fi and forwards control commands to Snapdragon Navigator.

When executing `snav_dronecontroller_companion`, the IP address of the drone must be specified if it is not the default (192.168.1.1). For example, to run the application with an IP address of 192.168.2.1:

```
$ sudo snav_dronecontroller_companion -i 192.168.2.1
```

Run the following to display `snav_dronecontroller_companion` help:

```
$ snav_dronecontroller_companion -h
```

2.6.2 Setting Up a Spektrum RC

Instructions on using a Spektrum RC are beyond the scope of this document, but the RC can be set up to control channels five and six. These channels control vehicle behavior and are used to change flight mode or enable/disable certain features.

Note: On Spektrum radios with two-position switches, it might help to use the mix to access more features on a particular channel. Refer to Spektrum documentation for instructions on assigning switches, using mixes, and changing the travel adjust.

The Spektrum RC can be bound in the field to the Spektrum satellite receiver using the built-in binding functionality of Snapdragon Navigator.

Bind the Spektrum RC as follows:

1. Power on the vehicle and place it upside down on a stationary surface.
2. Wait for Snapdragon Navigator to boot up. Once the binding sequence has been activated, the status LED displays the Binding mode LED code listed in Section 12.1.
3. Activate binding mode on the Spektrum RC. Refer to the Spektrum product-specific manual for instructions.
4. Once binding is completed successfully, a tone is issued from the vehicle to alert the user that an RC is connected.

3 Data Logging and Viewing Telemetry

3.1 Overview

By default, data logging is started automatically when Snapdragon Navigator is started. A large amount of data is available in these log files that can be very useful for a number of uses including system tuning and discovering the cause of an unexpected behavior.

This data is logged to a binary dump in the following location:

`/var/log/snav/flight/`

3.1.1 Log File Naming

Accurate time information is not possible without persistent device power. Log files are named each time Snapdragon Navigator starts, the first number in the file name (the major index) is incremented from the last time Snapdragon Navigator was started.

```
snav_00001_01.log
snav_00002_01.log
```

This convention ensures unique, identifiable names that increase monotonically. Log files continue to grow until reaching a specified size (default$\sim$ 200MB) before starting a new file. When a new log file is started, the second number (minor index) is incremented and the major number remains the same.

```
snav_00001_01.log
snav_00001_02.log
```

Log files with a minor index not equal to 1 do not contain version information because this information is only logged when Snapdragon Navigator is started.

Note: By default, Snapdragon Navigator stores a maximum of 1 GB of logs. Be sure to copy the log files before they are automatically removed.

Log files roll over to gracefully delete older log files and prevent running out of disk space.

Log files contain binary representations of data to reduce file sizes and computational requirements while ensuring data exactness. The impact of using this format is that log files are not human readable except when using the provided interpreter, `snav_inspector`.

3.1.2 Machine Vision Logs

Console output from some Snapdragon Navigator vision programs running on the applications processor is logged to the following location:

`/var/log/snav/vision`

This directory includes output of the five most recent logs from vision-based applications that might be enabled.

The Snapdragon Navigator vision-based applications can log the images consumed by the application through MV sequences. Machine vision (MV) sequence logging can be enabled by setting the `enable_srw_writer` parameter in `/etc/snav/app_params.<app_name>.xml` to 1.

MV sequences are logged to `/var/log/snav/mv_sequence`.

Caution: MV sequences are not automatically removed; be careful to avoid filling up disk space.

3.2 Live Data Viewing

The `snav_inspector` console application enables viewing the binary data from log files in a human readable form. This application works by continuously reading log files and formatting the requested data. The `snav_inspector` application runs on target on the applications processor, so no setup is required on a host computer to use this tool.

To run `snav_inspector` for live data viewing, open a terminal on the device and start the application when Snapdragon Navigator is running:

```
$ snav_inspector
```

The main screen of `snav_inspector` displays a numbered list of items with hotkey instructions at the bottom of the window. To expand an item, enter the number and press the `[enter]` key. When an item is expanded, several fields are exposed which update in real time as the flight control system runs. To collapse an item, enter the item number and press the `[enter]` key again.

The `snav_inspector` application can also be used to view debug output. To print debug output to `stdout`, start `snav_inspector` with the `-d` option:

```
$ snav_inspector -d
```

To open a previous log with `snav_inspector`, use the `-o` option to pass the log name:

```
$ snav_inspector -o /path/to/log/file
```

Run the following command to see all `snav_inspector` command line options:

```
$ snav_inspector -h
```

Note: If the viewer gets out of date with current logging message definitions, the display might be inaccurate. See Section 3.4 for version confirmation instructions.

3.3 Log Parsing

To parse a log file to human readable .csv files with `snav_inspector`, use the -w option to pass the output directory to be created and written to.

For example, to create the desired directory and comma separated value (csv) documents within the directory, use the following command:

```
$ snav_inspector -w /path/to/desired/directory
```

The -w flag can also be used with the following flags to parsing logs:

- -m – Filters output
- -o – Chooses the desired log file

For example, to only generate .csv files with “attitude” contained in the name, use the following command:

```
$ snav_inspector -o /path/to/log/file -w /path/to/desired/directory -m
attitude
```

Note: Parsing large log files might be slow with the `snav_inspector` application. For the fastest performance, compile `snav_inspector` for 64-bit Linux machines.

3.4 Additional Logging System Information

3.4.1 View Data Logging Options

Data logging can be disabled or configured to use custom file sizes and naming conventions. Use the following command on the target to see available options:

```
$ snav -h
```

Table 3-1 lists commands specific to data logging.

Table 3-1 Snapdragon Navigator data logging commands

Command	Description
-g	Enables data logging.
-s	Disables data logging.
-r	Sets the file size in MB at which to rollover the log (Default – 200 MB).
-n	Specifies the log file name. Selecting the existing name overwrites the file. Log file cleanup does not occur automatically on these files.

For example, to disable all logging, use the following command:

```
$ snav -s
```

To specify a custom log file name and file size, use the following command:

```
$ snav -r 300 -n log_name
```

3.4.2 Ensure Log Storage

The underlying operating system manages writing to files so that data is stored in memory for a while before writing it to persistent memory (disk). When power is removed, contents in memory are erased, which can cause portions of log files to be lost. To ensure a file has been completely written to disk, use the sync utility to flush data in memory to disk before removing power:

```
$ sync
```

3.4.3 Get the snav_inspector Version

The `snav_inspector` application might not be compatible with log files generated from different releases. Ensure the correct version of `snav_inspector` is used to parse log files.

Run the following command for the `snav_inspector` and log file version:

```
$ snav_inspector -v -o /path/to/log/file
```

Version information is only logged in the first log file generated when Snapdragon Navigator starts and therefore must contain a minor index equal to 1.

Caution: There is a separate clock used on the aDSP compared to the applications processor. These clocks have different offsets, so it is important to ensure timing data is used properly.

4 Flight Modes

Depending on the flight mode, the vehicle responds differently to user input from the RC. It is important to understand the differences between the flight modes and what inputs each mode is expecting.

In a Mode 2 RC configuration, moving the left stick up/down controls thrust or Z velocity and moving the left stick left/right controls yaw rate, while moving the right stick up/down controls pitch or X velocity and moving the right stick left/right controls roll or Y velocity.

Note: Not all flight modes are accessible via the DroneController application due to the difficulty of flying in certain modes with virtual joysticks. **Only Position Hold mode is supported by DroneController**.

4.1 Flight Mode Overview

4.1.1 Thrust Angle Mode

Associated color – Blue

CH1 controls – Thrust magnitude

CH2 controls – Roll angle

CH3 controls – Pitch angle

CH4 controls – Yaw rate

Description

User controls the vehicle's roll and pitch angles, yaw rate, and thrust magnitude.

In a Mode 2 RC configuration, the left stick controls thrust and yaw rate, while right stick controls roll and pitch angles. When the right stick is centered, the vehicle stabilizes its attitude to zero roll and pitch.

Note: To hover in place, this mode requires minor thrust, roll, and pitch adjustments from the pilot.

4.1.2 Thrust Angle GPS Hover Mode

Associated color – Blue and Green

CH1 controls – Thrust magnitude

CH2 controls – Roll angle

CH3 controls – Pitch angle

CH4 controls – Yaw rate

Description

User controls are identical to Thrust Angle mode. However, if the user does not input any roll or pitch commands (roll and pitch sticks are in their neutral position), the vehicle uses GPS to hover in place

(laterally). The user can still adjust thrust and yaw.

4.1.3 GPS Position Control Mode

Associated color – Purple and Green

Also known as: GPS Body Velocity Control mode

CH1 controls – Z velocity

CH2 controls – Y velocity

CH3 controls – X velocity

CH4 controls – Yaw rate

Description

User controls the vehicle's body-relative X, Y, and Z velocities and yaw rate. GPS holds the vehicle's current position if no control is commanded.

In a Mode 2 RC configuration, the left stick controls the Z velocity and yaw rate, and the right stick controls X and Y velocities. When the right stick is centered, the vehicle attempts to hold its horizontal position. When the left stick is centered, the vehicle attempts to hold its current altitude.

When the left stick is above center, the vehicle ascends; when the left stick is below center, the vehicle descends.

Note: This mode is easy for new pilots.

4.1.4 GPS Position Control Mode with Optic Flow

Associated color – Purple and Green when using GPS data, Green when using Optic Flow data

Also known as: Hybrid GPS Mode, GPS + Optic Flow Mode

CH1 controls – Z velocity

CH2 controls – Y velocity

CH3 controls – X velocity

CH4 controls – Yaw rate

Description

User controls the vehicle's body-relative X, Y, and Z velocities and yaw rate. GPS holds the vehicle's current position if no control is commanded. If the vehicle detects that it has good optic flow and sonar data, the vehicle uses optic flow data to improve hovering performance. The LED color changes to Green when the vehicle is using optic flow data.

In a Mode 2 RC configuration, the left stick controls the Z velocity and yaw rate, and the right stick controls X and Y velocities. When the right stick is centered, the vehicle attempts to hold its horizontal position. When the left stick is centered, the vehicle attempts to hold its current altitude.

When the left stick is above center, the vehicle ascends; when the left stick is below center, the vehicle descends.

Note: This mode is easy for new pilots.

Note: Ensure that Snapdragon Navigator is configured for Optic Flow (DFT). See Section 10.1.

4.1.5 Height Control Mode

Associated colors – Purple when the barometer is used, cyan when the sonar is used, red and green when the tilt angle limits are reduced.

Also known as: Z Velocity Control mode, Pressure mode, Sonar mode

CH1 controls – Z velocity

CH2 controls – Roll angle

CH3 controls – Pitch angle

CH4 controls – Yaw rate

Description

User controls the vehicle's roll and pitch angles, yaw rate, and Z velocity.

In a Mode 2 RC configuration, the left stick controls Z velocity and yaw rate, and the right stick controls roll and pitch angles. When the right stick is centered, the vehicle stabilizes its attitude to zero roll and pitch. When the left stick is centered, the vehicle attempts to hold its current altitude.

When the left stick is above center, the vehicle ascends at a velocity proportional to the stick input; when the left stick is below center, the vehicle descends at a velocity proportional to the stick input.

Note: This mode requires minor roll and pitch adjustments from the pilot to hover in place.

Note: Height Control mode is similar to Thrust Angle mode, but does not directly control thrust. Instead, the user controls the vehicle's vertical velocity, with center stick corresponding to zero vertical velocity.

4.1.6 Optic Flow (Downward-Facing Tracker) Mode

Associated color – Green

CH1 controls – Z velocity

CH2 controls – Y velocity

CH3 controls – X velocity

CH4 controls – Yaw rate

Description

User controls the vehicle's body-relative X, Y, and Z velocities and yaw rate.

In a Mode 2 RC configuration, the left stick controls the Z velocity and yaw rate, and the right stick controls X and Y velocities. When the right stick is centered, the vehicle attempts to hold its horizontal position. When the left stick is centered, the vehicle attempts to hold its current altitude.

When the left stick is above center, the vehicle ascends; when the left stick is below center, the vehicle descends.

Note: This mode is easy for new pilots.

Note: Ensure that Snapdragon Navigator is configured for Optic Flow (DFT). See Section 10.1.

4.1.7 VIO Mode

Associated color – Yellow

CH1 controls – Z velocity

CH2 controls – Y velocity

CH3 controls – X velocity

CH4 controls – Yaw rate

Description

User controls the vehicle's body-relative X, Y, and Z velocities and yaw rate.

In a Mode 2 RC configuration, the left stick controls the Z velocity and yaw rate, and the right stick controls X and Y velocities. When the right stick is centered, the vehicle attempts to hold its horizontal position. When the left stick is centered, the vehicle attempts to hold its current altitude.

When the left stick is above center, the vehicle ascends; when the left stick is below center, the vehicle descends.

Note: This mode is easy for new pilots.

Note: Ensure that Snapdragon Navigator is configured for VIO. See Section 10.2.

4.1.8 Position Hold Mode

Associated color – Yellow and Green

CH1 controls – Z velocity

CH2 controls – Y velocity

CH3 controls – X velocity

CH4 controls – Yaw rate

Description

Note: This mode is under development and should be considered experimental.

This mode uses vision data, GPS data, or both to hold the position when no input is commanded. This mode can be considered the "smartest" flight mode. Position Hold mode attempts to maintain consistent velocity-style control regardless of which sensors are used to estimate position (indoors or outdoors). Some features might only be available when the vehicle has a GPS lock, including return-to-launch, geotether, and no fly zones. Position Hold mode is accessible if any combination of GPS, VIO, or optic flow (DFT) data is available.

User controls the vehicle's body-relative X, Y, and Z velocities and yaw rate.

In a Mode 2 RC configuration, the left stick controls the Z velocity and yaw rate, and the right stick controls X and Y velocities. When the right stick is centered, the vehicle attempts to hold its horizontal position. When the left stick is centered, the vehicle attempts to hold its current altitude.

When the left stick is above center, the vehicle ascends; when the left stick is below center, the vehicle descends.

Note: This mode is easy for new pilots.

Note: See Chapter 10 for information on configuring vision data with Snapdragon Navigator.

4.1.9 Rate Mode

Associated color – Red

CH1 controls – Thrust magnitude

CH2 controls – Roll rate

CH3 controls – Pitch rate

CH4 controls – Yaw rate

Description

User controls vehicle's roll, pitch, yaw rates, and thrust magnitude.

In a Mode 2 RC configuration, the left stick controls the thrust and yaw rate, and the right stick controls the roll and pitch rates. This mode does not automatically stabilize its roll and pitch angles to zero when the right stick is centered.

Controlling the roll and pitch rates enables an experienced pilot to perform acrobatic maneuvers, such as flips.

Caution: This mode requires a skilled pilot and is not recommended for beginners.

4.2 Accessing Flight Modes

A Spektrum RC can be used to access all flight modes with the appropriate channel value.

CH5 is used to switch between flight modes

$+50\% <= \text{CH5} \rightarrow$ Thrust Angle mode (**Blue** mode)
$\rightarrow$ Thrust Angle GPS Hover mode (GPS enabled) (**Blue** and **Green** modes)

$-50\% <= \text{CH5} < +50\% \rightarrow$ Height Control mode (No GPS) (**Purple** mode OR **Cyan** mode)
$\rightarrow$ GPS Position Control mode (GPS enabled) (**Purple** and **Green** modes)
$\rightarrow$ GPS Position Control mode with Optic Flow (GPS with Optic Flow enabled)
(**Purple** and **Green** modes OR **Green** mode only)

With `enable_vio_mode` set to 0[1]:

$-120\% <= \text{CH5} < -50\% \rightarrow$ Optic Flow (DFT) Mode (**Green** mode)

With `enable_vio_mode` set to 1:

$-106\% <= \text{CH5} < -50\% \rightarrow$ Optic Flow (DFT) mode (**Green** mode)
$-120\% <= \text{CH5} < -106\% \rightarrow$ VIO mode (**Yellow** mode)

$-135\% <= \text{CH5} < -120\% \rightarrow$ Position Hold mode (**Yellow** and **Green** mode)

$\text{CH5} < -135\% \rightarrow$ Rate mode (**Red** mode)

CH6 is used to enable/disable GPS and to trigger landing

CH6 HIGH $\rightarrow +50\% <= \text{CH6} \rightarrow$ Trigger landing (see Section 5 for details)

CH6 MID $\rightarrow -50\% <= \text{CH6} < +50\% \rightarrow$ GPS enabled

[1] See Section 8.28 for details

CH6 LOW $\rightarrow$ CH6 $< -50\%$ $\rightarrow$ GPS disabled

4.3 Starting the Propellers

Before attempting to start the propellers, be sure that the propellers are clear of any obstructions and that the status LED is displaying a code that corresponds to a flight mode, not an error code. The propellers do not spin up if the LED is showing an error code. See Section 12.1 for a guide to LED color codes.

To start the propellers:

1. Move the thrust stick down while keeping it centered (CH1 must be between -100% and -94%)
2. Move the yaw stick all the way to the left or right (CH1 must remain between -100% and -94%, CH4 must reach either less than -62% or greater than 62% depending on the direction); vehicle emits a tone to indicate it received the first part of the sequence
3. Move the yaw stick all the way to the opposite side (CH1 must remain between -100% and -94%, CH4 must reach either less than -62% or greater than 62% depending on the previous step)
4. Move the yaw stick back to center (CH1 must remain between -100% and -94%, CH4 must enter the range of -9% to 9%); propellers start spinning

This sequence corresponds to lowering the thrust and yawing quickly left then right, or lowering the thrust and yawing quickly right then left.

If the RC is trimmed properly, a tone sounds when the throttle is reduced and the yaw stick reaches the far left or right side. The propellers start spinning when the yaw stick returns to center.

4.4 Stopping the Propellers

Set up the throttle cut on your radio. The recommended procedure is to use the bind/trainer button on the radio as throttle cut, so that pressing it reduces CH1 to the minimum value (-150%).

To stop the propellers manually:

1. Hold down throttle cut (CH1 must be less than -105%)
2. While holding throttle cut, move the yaw stick all the way to the left OR all the way to the right (CH4 must reach either less than -62% or greater than 62% depending on the direction); propellers immediately stop spinning

This sequence corresponds to holding throttle cut and yawing left or right. Alternatively, the vehicle automatically stops the propellers after it has been on the ground for a brief period.

5 Landing Behavior

If CH6 is high or the radio link is lost, Snapdragon Navigator puts the vehicle in a landing mode. The landing mode is chosen based on the available sensor data.

5.1 Return to Home

This mode is entered if the vehicle had GPS lock at takeoff, is receiving good GPS data, and one of the following conditions occurs:

- The radio link drops out during flight
- CH6 is set to HIGH

Description

Vehicle returns to the position where it took off from and and lands automatically. Pilot has a small amount of control in lateral position during the vehicle's descent if desired.

5.2 Landing With GPS

This mode is entered if the vehicle did not have GPS lock at takeoff, is receiving good GPS data, and one of the following conditions occurs:

- The radio link drops out during flight
- CH6 is set to HIGH

Description

Vehicle descends in place, attempting to land directly below the position where the landing was activated. Pilot has a small amount of control in lateral position during the vehicle's descent if desired.

5.3 Landing With Barometer

This mode is entered if the GPS is unavailable and one of the following conditions occurs:

- The radio link drops out during flight
- CH6 is set to HIGH

Description

The vehicle descends at a fixed velocity using barometer data.

Note: The vehicle drifts during descent.

5.4 Emergency Landing

This mode is entered if the barometer is unavailable and one of the following conditions occurs:

- The radio link drops out during flight
- CH6 is set to HIGH

Description

The vehicle stabilizes to zero roll and pitch angles while issuing a fixed low thrust command so that it descends rapidly.

Caution: This landing mode is a last resort. The descent rate might be fast enough to damage the vehicle. The vehicle drifts during descent.

6 Sensor Calibration Procedures

Snapdragon Navigator leverages several sensor calibration procedures to get the best performance out of each sensor. Follow the instructions for each calibration procedure closely. Most calibrations occur with the vehicle stationary on the ground, but the in-flight accelerometer offset and trim estimation procedure is one exception.

Note: Calibration procedures are not currently accessible from DroneController. They can only be triggered with a Spektrum RC or through the Snapdragon Navigator API (refer to *Qualcomm Snapdragon Navigator Developer Guide* (80-P4698-20)).

6.1 Gyroscope and Accelerometer Temperature Calibration

6.1.1 Important Notes

- This calibration is not required, but it can improve performance when the IMU undergoes large temperature variations.
- This calibration removes any accel offset calibration as the accel offset calibration references the temperature calibration.
- This mode can only be entered if the propellers are not spinning.
- The vehicle must be kept completely still during this procedure. If motion is detected, the software causes the calibration to fail.

6.1.2 Self Heating

Self-heating functionality is available via the `snav_calibration_manager` application. When enabled, the application loads CPU cores to hold the temperature gradient at the specified linear rate. The specified rate can be positive or negative. If the rate is negative, the device is preheated by full CPU loading for five minutes prior to starting calibration.

The self-heating application does not limit the magnitude of the specified rate, but a limit of +/-6°C/min was measured on reference devices. The recommended rate is +5°C/min.

Note: If self-heating functionality is enabled, it is important that other active processes have minimal CPU loading.

6.1.3 Instructions

1. Place the vehicle upright on a flat stationary surface free from vibration. The vehicle can also be calibrated at an arbitrary orientation if **require_level_temp_calibration** is disabled as described in Section 8.15.
2. Set the initial IMU temperature:
   - Self-heating enabled – Initial IMU temperature must be near the ambient room temperature.
   - Self-heating disabled – Initial IMU temperature must be set as low as possible (ideally around 10°C).
3. Enter the calibration mode:
   - Self-heating enabled – Run the snav_calibration_manager application with the -r [X] option, where [X] is the desired temperature gradient in units of °C/min. See Section 6.1.2.
   - Self-heating disabled – Hold the roll stick right with pitch centered and toggle CH5 between low and high values quickly. Alternatively, run the snav_calibration_manager application with the -t option.

   The status LED indicates that the vehicle is in the Gyro and Accelerometer Temperature Calibration mode.
4. Begin heating the IMU.
5. The calibration takes approximately five minutes, over which time the temperature should increase from the initial temperature by at least 15°C.
   - If the calibration completes successfully, the linear fit is written to `/usr/share/data/adsp/imu_linear_calib.txt` and the LED displays the Calibration Success LED code (see Chapter 12).
   - If **require_level_temp_calibration** is enabled and the calibration completes successfully, accelerometer offset calibration parameters are stored to `/usr/share/data/adsp/offset_calib.txt`.
   - The calibration fails and the LED displays the Calibration Failure LED code (see Chapter 12) if any the following conditions occur:
     – There is any motion
     – The fit has an out of range residual
     – The temperature during the calibration is not to specification
     – The write fails
6. Reboot the vehicle when the procedure is completed.
7. Run either offset accelerometer calibration routine in Section 6.2 or Section 6.3.

6.2 Accelerometer Offset Estimation – On Ground

6.2.1 Important Note

This mode can only be entered with the propellers not spinning.

6.2.2 Instructions

1. Place the vehicle on a flat, stationary surface as level as possible. The calibration will fail if the accelerometer values are not within the ranges described in Section 8.11.
2. Hold the pitch stick down with roll centered and toggle CH5 between low and high values quickly.
3. The status LED indicates that the vehicle has entered Accelerometer Offset Estimation – On Ground mode.
4. Wait for approximately ten seconds until LED reports either success or failure (see Chapter 12).
5. If successful, calibration parameters are calculated and the result is stored to `/usr/share/data/adsp/offset_calib.txt`.
6. Reboot the vehicle after the calibration is completed.

6.3 Accelerometer Offset and Trim Estimation – In Flight

6.3.1 Important Notes

- This mode can only be entered during flight.
- This mode attempts to identify any biases in sensors and the weight distribution of the vehicle. To avoid measuring the ground effect, fly the vehicle as close to stationary as possible, a few feet off of the ground, and away from walls.
- The instructions are for running this calibration during a manual flight. This procedure can also be performed automatically without a pilot using the Snapdragon Navigator API. Refer to *Qualcomm Snapdragon Navigator Developer Guide* (80-P4698-20) for details.

6.3.2 Instructions

1. Using a Spektrum RC, put the vehicle in any flight mode and take off (see Section 4.1).
2. Fly the vehicle so that it remains as stationary as possible.
3. Rapidly toggle CH6 between low and mid values (do not toggle too high or the vehicle might enter a Landing Mode, see Section 4.2).
4. The status LED indicates that the Accelerometer Offset and Trim Estimation – In Flight calibration is in progress.
5. Continue to fly the vehicle so that it remains as stationary as possible. Avoid quick attitude adjustments.
6. The system finishes gathering data and the status LED reports success or failure (see Chapter 12).
7. Land the vehicle and stop the propellers. If successful, calibration parameters are calculated and the

result is stored to `/usr/share/data/adsp/offset_calib.txt`. The LED signals success or failure (see Chapter 12) depending on the status of the file write operation. Note the result.

8. Run the propeller startup sequence to exit the calibration (see Section 4.3). Status LED changes indicate the current flight mode. If the calibration fails, repeat steps 1-5. If the calibration is successful, the vehicle is ready to fly with the new parameters.

6.4 Magnetometer (Compass) Calibration

6.4.1 Important Notes

- This procedure calibrates the compass to the current environment. A calibration may be necessary if the vehicle is in a new environment, or if the compass displays a warning LED.
- The procedure works best outdoors in a metal-free environment.
- When the procedure is complete, the vehicle LED blinks green if the calibration passed, or blinks red LED if the calibration failed.
- After the calibration mode is entered, the entire procedure must be completed within 30 seconds.
- A successful calibration is stored in `/usr/share/data/adsp/compass_calib.txt`.
- This mode can only be entered with the propellers not spinning.

6.4.2 Instructions

1. In an open, metal-free area, hold the vehicle level with the front facing directly north.
2. With the vehicle on and bound to an RC controller, enter the Compass Calibration mode by holding the pitch stick forward, roll stick centered, and rapidly toggling CH5 values between low and high.
    - LED shows the **Compass Calibration Prep Mode** LED code
    - Ensure the vehicle is in position and wait 5 seconds until calibration begins and the LED shows the **Compass Calibration Mode** LED code (see Section 12.1.3)
3. Rotate the vehicle about the east-west axis, so that the vehicle does a full-front flip around the vehicle pitch axis.
4. Once the flip is complete and the vehicle is level, yaw the vehicle 45 degrees so that it is facing northeast.
5. Rotate the vehicle about the east-west axis. The vehicle flips diagonally around the front-right and back-left motors.
6. Once the flip is complete and the vehicle is level, yaw the vehicle 45 degrees so that it is facing east.
7. Rotate the vehicle about the east-west axis. The vehicle flips about the vehicle roll axis.
8. Once the flip is completed and the vehicle is level, yaw the vehicle 45 degrees so that it is facing southeast.
9. Rotate the vehicle once more about the east-west axis. The vehicle should flip diagonally around the front-left and back-right motors.
10. Wait for the LED to indicate whether the calibration passed or failed.

- Blinking green indicates a passing calibration
- Blinking red indicates a failed calibration

11. Restart the vehicle for the calibration to take effect.

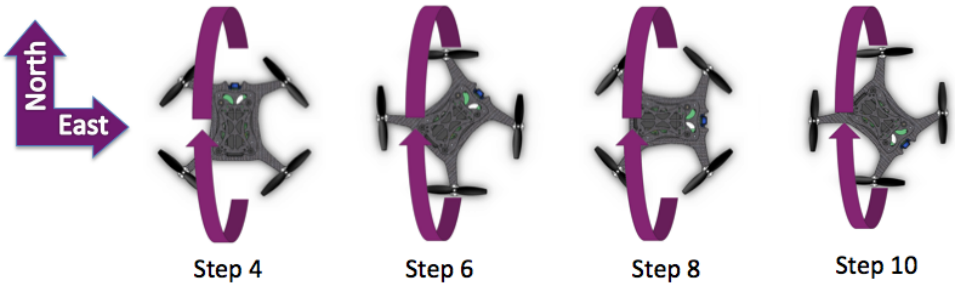


Figure 6-1 Compass calibration

6.5 Required/Recommended Calibration Procedures

Some calibration procedures must be performed before flight is permitted. In this release, the following calibration is required:

- Accelerometer Offset Estimation – On Ground

Some calibration procedures are recommended to achieve the best possible flight performance. In this release, the following calibration is recommended and triggers a warning chime at startup if it has not been performed:

- Accelerometer Offset and Trim Estimation – In Flight

The warning chime is a sequence of six tones from high to low frequency and is emitted at startup, before the normal initialization chime occurs.

A valid compass calibration is required to fly in any GPS mode.

- Magnetometer (Compass) Calibration

7 Electronic Speed Controllers

7.1 ESC Firmware Updates

Snapdragon Navigator supports ESC firmware updates without connecting or disconnecting any cables. The firmware is loaded from a known location on the applications processor and flashed on the connected ESCs.

7.1.1 Parameter Configuration

The following configuration block example specifies parameters for the ESC firmware update:

```
<esc_firmware_params>
<param name="firmware_file_name" value="esc_firmware.bin"/>
<param name="config_file_name" value="esc_params.eep"/>
<param name="force_firmware_update" value="1"/>
<param name="force_config_update" value="1"/>
<param name="config_bit_rate" value="57600"/>
</esc_firmware_params>
```

See Section 8.30 for more information.

7.1.2 Internal Firmware Update Sequence

With the parameters configured, Snapdragon Navigator internally updates the ESC firmware in the following sequence:

1. Locate (or first copy to the applications processor if need be) the firmware file and make sure `force_config_update` is set to 1, otherwise exit the procedure.
2. Repeat the following steps for *n* ESCs (where *n* is the number of motors):
   (a) Configure the UART baud rate to the normal ESC communication baud rate (250 K default).
   (b) Send a reset command to one ESC to exit the firmware and enter the bootloader.
   (c) Configure the UART baud rate to the bootloader baud rate (38400).
   (d) Transfer the ESC firmware to the bootloader.
   (e) When complete, the bootloader automatically exits and firmware execution starts.
3. Set the UART baud rate back to the standard ESC communication baud rate.
4. Locate the configuration file and make sure `force_config_rate` is set to 1, otherwise exit the procedure.
5. Set the UART baud rate to `config_bit_rate`.

6. Push the firmware configuration to all ESCs at the same time.
7. During the configuration installation, all ESCs flash with small status LEDs for 3 seconds.
8. Send restart command to exit the configuration mode and restart the ESC firmware.
9. Set the UART baud rate back to standard ESC communication baud rate.
10. Exit the procedure.

7.1.3 Update During Initialization

Enable ESC firmware updates during initialization as follows:

1. Set the `force_firmware_update` parameter to 1.
2. Store the firmware file on the applications processor at a known location and create a symlink to it called `esc_firmware.bin` to retain the original firmware filename.
3. Specify the firmware file name in the `firmware_file_name` parameter.

During initialization, Snapdragon Navigator searches the `/usr/share/data/adsp` directory for the firmware file name specified in the `firmware_file_name` configuration parameter.

- If the firmware file is found, Snapdragon Navigator automatically begins the firmware update process for each of the connected ESCs.
- If the firmware file is not found or `force_firmware_update` is not set to 1, the firmware is not updated.

Note: Snapdragon Navigator updates the ESC firmware during every initialization. To prevent subsequent updates, set the `force_firmware_update` parameter to 0.

7.1.4 ESC Firmware Configuration

ESC firmware configuration is separate from the firmware and updates separately. Use the `config_file_name` parameter to specify the ESC binary configuration.

Firmware configuration is uploaded separately and is not always required (allows firmware updates while maintaining the application-specific configuration).

- If `force_config_update` is set to 1 and `config_file_name` file is located, Snapdragon Navigator installs the firmware configuration to all ESCs.
- If a firmware update is in progress, the configuration updates after the firmware update.

Note: Do not upload the firmware unless the configuration file is available. See Section 7.1.5.

7.1.5 Firmware Upgrades

If upgrading from version of Snapdragon Navigator firmware without separate configuration files, the firmware requires an initial installation of the configuration before using the firmware. See Section 7.1.2.

After the configuration has been installed, configuration updates are no longer necessary with subsequent firmware updates.

Note: If the ESC has firmware but no configuration files, the ESC firmware accepts configuration at 57600

baud rate.

To update an existing configuration, set the `config_bit_rate` parameter to the regular ESC communication baud rate (i.e., 250000).

7.1.6 Update Procedure

Update the ESC firmware as follows:

1. Power on the vehicle and copy the ESC firmware file and ESC configuration file to a known location on the applications processor.
2. Add the `esc_firmware_params` section to Snapdragon Navigator parameters and make sure the filenames are correct and the files exist
3. Create a symlink to the firmware file (and configuration file) if needed, for example:
   ```
   $ ln -s /home/linaro/my_esc_firmware.bin
   /usr/share/data/adsp/esc_firmware.bin
   ```
4. Call `sync` from the applications processor console.
5. Reboot the vehicle. The ESC firmware update starts automatically as Snapdragon Navigator initializes.
6. Use `snav_inspector` to view the output from the firmware update process (see Section 3.2):
   ```
   $ snav_inspector -d
   ```
   The ESCs update sequentially. Each ESC flashes via status LED on the ESC board and emits a beep to indicate when the firmware update is complete.
7. After each ESC firmware update, the ESC configuration file is pushed to all ESCs at the same time. During this process (3 sec), all ESCs flash their status LED and reboot together.

   When the final ESC update is complete, Snapdragon Navigator continues as normal.
8. Do one of the following to prevent the firmware update during reboot:
   - Remove or rename `/usr/share/data/adsp/esc_firmware.bin`.
   - Set the `force_firmware_update` and `force_config_update` parameters to 0.

8 Parameters Description

Each vehicle must have the runtime parameters file in the proper location for Snapdragon Navigator to function properly. When creating a parameters file for a new vehicle, it is easiest to start with an existing parameters file from a similar vehicle and modify the parameters.

Parameters labeled *required* must be specified within the runtime parameters file in order for Snapdragon Navigator to initialize. The default values reported are for the small sample drone and must be modified for any other vehicle. For example, to change the **basethrust** parameter in Section 8.8, the corresponding XML lines are in the following format:

```
<position_control_params>
    <param name="basethrust" value="240.0"/>
</position_control_params>
```

The XML group is the subsection heading and parameters are listed within the group with attributes `name` and `value`.

Parameters are grouped into the same subsections as organized in the XML runtime parameter files. The notation is in the following format:

parameter – *required*, `type`, units, default value, [min, max]

8.1 imu_conditioner_params

require_initial_motionless_period – `int`, unitless, 1, [0, 1]

1 is enabled, 0 is disabled. If enabled, an initial motionless period is required for the default attitude estimate initialization method. If disabled, the default attitude estimate initialization method times out after a period of **initial_motionless_period_timeout** microseconds and the software attempts to initialize with an alternative attitude initialization method with less strict requirements on how much motion is allowed. Setting this parameter to disabled might reduce vehicle performance.

initial_motionless_period_timeout – `int64_t`, microseconds, 3000000

If **require_initial_motionless_period** is disabled, the initial attempt to use the default attitude estimate initialization method times out after this many microseconds.

8.2 optic_flow_data_params

min_optic_flow_sample_size – `int32_t`, 20, [0, –]

See comments for **optic_flow_sample_timeout_time** and **optic_flow_sample_relock_time**.

optic_flow_sample_timeout_time – `int64_t`, us, 500000, [0, –]

Optic flow data is deemed invalid if fewer than **min_optic_flow_sample_size** samples are observed for this period of time.

optic_flow_sample_relock_time – `int64_t`, us, 1000000, [0, –]

After optic flow data is deemed invalid due to low sample size, it can become valid again if at least **min_optic_flow_sample_size** samples are observed for this period of time.

8.3 optic_flow_estimate_data_params

bad_range_timeout_time – `int64_t`, us, 3000000, [0, –]

The optic flow estimate is deemed invalid if no good sonar range measurements are observed for at least this period of time once the vehicle is in flight.

8.4 optic_flow_estimator_params

min_sample_size – `int16_t`, 10

Minimum optic flow sample size (minimum number of features) required for data to be used in estimator.

max_sample_size – `int16_t`, 600

Maximum optic flow sample size (maximum number of features) allowed for data to be used in estimator.

camera_offset_x – `float`, meters, 0.0

Distance at which the optic flow camera is offset in the forward (+x) direction.

camera_offset_y – `float`, meters, 0.0

Distance at which the optic flow camera is offset in the forward (+y) direction.

max_pixel_velocity – `float`, pixels/second, 1500.0

Maximum pixel velocity that is used by the optic flow estimator. Pixel velocities above this value (or below this negative value) are capped at this magnitude.

limit_max_xy_vel_using_range – `int`, 1, [0, 1]

Limit the maximum lateral velocity magnitude that can be commanded as a function of the sonar range measurement. When this parameter is enabled (1), the maximum allowed lateral velocity magnitude is reduced as the range decreases.

min_range_at_max_vel – `float`, m, 1.2, [0, –]

The minimum range where the maximum lateral velocity magnitude given by **optic_flow_mode_xy_gain** is achievable. The value given by this parameter might be overwritten if the flight controller determines that it is unsafe.

max_vel_at_min_range – `float`, m/s, 0.2, [0, –]

The maximum lateral velocity magnitude that can be commanded at the minimum sonar range. The value given by this parameter may be overwritten if the flight controller determines that it is unsafe.

8.5 voa_params

enable_voa – `int`, 1, [0, 1]

Set **enable_voa** to 0 to disable visual obstacle avoidance (VOA). VOA is currently only functional if the primary sensor mode is VIO.

distance_bound – `float`, m, 1.5, [0, –]

The distance from which to avoid a detected object. This value must be larger than the minimum sensor range to prevent collision.

allow_sliding_motion – `int`, 1, [0, 1]

Set **allow_sliding_motion** to 0 to enforce a constraint that the output velocity must lay on the vector of the desired velocity. If set to 1, the direction of user velocities can be modified, including being pushed away from objects. Setting to 0 is more conservative, but does not feel as intuitive.

enable_voa_xy_stationary – `int`, 0, [0, 1]

Flag determining whether VOA is active when there is no commanded velocity or acceleration in X and Y plane. It is safest to set this parameter to 0 to ensure the vehicle cannot be pushed back when no planar velocity is commanded.

wn_converge – `float`, 1/s, 1.5, [0.0, –]

Natural frequency of the controller to slow the vehicle toward detected obstacles. Higher values stop more abruptly and lower values have an impact over a longer distance. If the maximum allowed velocity is very high, increasing this value too far can cause collisions.

enable_voa_running_led – `int`, 1, [0, 1]

Flag indicating whether VOA uses LEDs while processing to alert the user that the system can modify control output. Set this flag to 0 to disable modified LEDs while VOA is running.

enable_voa_active_led – `int`, 1, [0, 1]

Flag indicating whether VOA uses LEDs while modifying control output. This set of LED colors overrides the running LEDs if VOA is active (except battery warning LEDs).

8.6 obstacle_output_params

angle_per_bin – `float`, radians, 0.3

Angle in radians for each number of the relative distances to correspond to. If this angle is wider than the depth sensor's field of view, the corresponding bins are invalid.

8.7 velocity_smoother_params

Parameters in this section only apply to VIO mode.

max_jerk_allowed – `float`, m/s/s/s, 100, [1, –]

Maximum allowable jerk to move the desired position in an X-Y plane, in which jerk is the time derivative of acceleration. In ideal conditions near horizontal level flight, this parameter is approximately 9.81 times the roll or pitch angular rate in rad/sec.

max_acc_allowed – `float`, m/s/s, 6, [.1, –]

Maximum allowable acceleration to move the desired position in an X-Y plane, in which acceleration is the time derivative of velocity. In ideal conditions, this parameter is directly related to the maximum roll or pitch angle through: **max_acc_allowed** $= 9.81 * \tan(\text{max angle})$. This parameter must be lower in high-wind conditions.

acc_converge_bound – `float`, m/s/s, 6, [0, –]

Used in conjunction with **max_vel_error** to determine how the system tracks desired velocities. Increasing **acc_converge_bound** increases the rate of convergence to the desired velocity. The **acc_converge_bound** value must be less than or equal to **max_acc_allowed**.

max_vel_error – `float`, m/s, 6, [0.001, –]

Used in conjunction with **acc_converge_bound** to determine how the system tracks desired velocities. **max_vel_error** is the cap for velocity tracking error that corresponds to the acceleration **acc_converge_bound**. Increasing **max_vel_error** decreases the rate of convergence to the desired velocity.

8.8 position_control_params

max_gps_control_tilt_angle – `float`, radians, 0.610865238198, [0.0, 1.23]

Maximum total roll and pitch tilt angle command for the GPS position controller.

max_optic_flow_control_tilt_angle – `float`, radians, 0.4, [0.0, 1.23]

Maximum total roll and pitch tilt angle command for the Optic Flow position controller.

max_vio_control_tilt_angle – `float`, radians, 0.6, [0.0, 1.23]

Maximum total roll and pitch tilt angle command for the VIO position controller.

max_pos_control_tilt_angle – `float`, radians, 0.610865238198, [0.0, 1.23]

Maximum total roll and pitch tilt angle command for Position Hold mode.

basethrust – *required*, `float`, grams, 240.0

Constant thrust added to Z position controller. Total controller output is this value plus feedback terms. The basethrust value must be close to the mass of the vehicle.

kp_z_factor – `float`, 1/meter, 0.917, [0.001, –]

The proportional gain for the Z (vertical) position controller is equal to this value multiplied by the **basethrust** parameter.

kd_z_factor – `float`, 1/(meter/sec), 0.734

The derivative gain for the Z (vertical) position controller is equal to this value multiplied by the **basethrust** parameter.

kp_xy – `float`, radians/meter, 0.15207125

Proportional gain for the XY (lateral) GPS position controller. Controller output is the tilt angle in radians.

kd_xy – `float`, radians/(meter/sec), 0.3486215

Derivative gain for the XY (lateral) GPS position controller. Controller output is the tilt angle in radians.

kp_xy_optic_flow – `float`, radians/meter, 0.55033
Proportional gain for the XY (lateral) Optic Flow position controller. Controller output is the tilt angle in radians.

kd_xy_optic_flow – `float`, radians/(meter/sec), 0.55033
Derivative gain for the XY (lateral) Optic Flow position controller. Controller output is the tilt angle in radians.

kp_xy_vio – `float`, radians/meter, 0.8
Proportional gain for the XY (lateral) VIO position controller. Controller output is the tilt angle in radians.

kd_xy_vio – `float`, radians/(meter/sec), 0.671275
Derivative gain for the XY (lateral) VIO position controller. Controller output is the tilt angle in radians.

force_landing_speed – `float`, meters/sec, -0.75, [–, 0.0]
Descent velocity when the vehicle is forced to land due to low voltage. This value must be less than 0.

max_relative_pressure_height – `float`, meters, 50.0
Maximum relative commanded height in Z velocity control modes.

8.9 state_machine_params

prop_startup_time_1us – `int64_t`, 4000000
Length of time in microseconds that propellers are given to start spinning before timing out.

enable_hybrid_gps_outdoor_mode – `int`, unitless, 0, [0, 1]
Enables (1) or disables (0) a hybrid mode when the GPS is enabled and transitions to an Optic Flow mode (if conditions allow). This flag causes the reported mode to change between multiple modes for the same command.

max_ang_vel_component_for_starting_props – `float`, rad/second, 0.5236
Maximum X, Y, or Z angular velocity component in rad/sec for which starting the propellers is allowed.

max_accel_mag_for_starting_props – `float`, Gravity, 1.3
Maximum acceleration magnitude for which starting the propellers is allowed.

min_accel_mag_for_starting_props – `float`, Gravity, 0.7
Minimum acceleration magnitude for which starting the propellers is allowed.

max_tilt_angle_for_starting_props – `float`, radians, 1.047
Maximum total roll and total pitch tilt angles in radians for which starting the propellers is allowed.

8.10 height_estimator_params

use_min_pressure_height_limit_feature – `uint8_t`, unitless, 0
Enables (1) or disables (0) a specific height control feature. This feature attempts to fix Z position control issues due to high pressure from propeller ground effects when the vehicle is close to the ground. Only

use this feature if it is necessary.

min_sonar_range – `float`, meters, 0.25
Minimum sonar range in meters used by the height estimator, sonar ranges below this value are not used.

max_sonar_range – `float`, meters, 6.5
Maximum sonar range in meters used by the height estimator, sonar ranges above this value are not used.

initial_barometer_rate – `float`, hertz, 49.1
Initial barometer rate in Hz, this estimate is updated as barometer data is received.

min_barometer_rate – `float`, hertz, 20
Minimum barometer rate that can be estimated in Hz.

max_barometer_rate – `float`, hertz, 150
Maximum barometer rate that can be estimated in Hz.

initial_sonar_rate – `float`, hertz, 10.15
Initial sonar rate in Hz, this estimate is updated as sonar data is received.

min_sonar_rate – `float`, hertz, 5
Minimum sonar rate that can be estimated in Hz.

max_sonar_rate – `float`, hertz, 25
Maximum sonar rate that can be estimated in Hz.

8.11 imu_accel_offset_params

The imu_accel_offset_params parameters are used for accelerometer offset estimation on ground calibration. See Section 6.2 for the procedure.

max_xy_accel_magnitude – `float`, Gravity, 0.5
Maximum X or Y accelerometer value magnitude value allowed for successful calibration.

min_z_accel_value – `float`, Gravity, 0.75
Minimum Z accelerometer value allowed for successful calibration.

max_z_accel_value – `float`, Gravity, 1.25
Maximum Z accelerometer value allowed for successful calibration.

8.12 voltage_monitor_params

voltage_warn_threshold – *required*, `float`, volts, 7.05
Below this threshold the status LED displays the Low Voltage warning.

voltage_warn_critical – *required*, `float`, volts, 6.95
Below this threshold the status LED displays the Critical Voltage warning. This value must be less than **voltage_warn_threshold**.

voltage_force_landing – *required*, `float`, volts, 6.9

If the voltage is below this value for a certain amount of time the vehicle is forced to descend. If the vehicle is in a Z position controlled mode the vehicle is forced to descend at a rate specified by the **force_landing_speed** parameter. If the vehicle is in a thrust-controlled mode, the maximum commanded thrust value is lowered according to the slope defined by the **low_voltage_thrust_slope** parameter.

voltage_no_start – *required*, `float`, volts, 6.7

Below this voltage the propellers are not allowed to spin up.

voltage_warn_threshold_external – `float`, volts, -1.0

If this value is greater than 0 and the voltage is measured by an external driver, this value is used in place of **voltage_warn_threshold**. If this value is less than 0, **voltage_warn_threshold** is always used no matter how the voltage is sensed.

voltage_warn_critical_external – `float`, volts, -1.0

See **voltage_warn_threshold_external**.

voltage_force_landing_external – `float`, volts, -1.0

See **voltage_warn_threshold_external**.

voltage_no_start_external – `float`, volts, -1.0

See **voltage_warn_threshold_external**.

low_voltage_thrust_slope – *required*, `float`, grams/volts, 700

If the voltage is below the **voltage_force_landing** parameter for a certain duration, the vehicle is forced to descend. This slope value determines how much the maximum commanded vehicle thrust is lowered when the sensed voltage is below the **voltage_force_landing** value. For example, if the sensed voltage is 1 V below **voltage_force_landing**, the maximum thrust is lowered by **low_voltage_thrust_slope**.

voltage_landing_reset_threshold – `float`, volts, 30.0

The vehicle is forced to land if the sensed voltage is below a certain voltage. The forced landing logic resets if the voltage is above **voltage_landing_reset_threshold** if the propellers are not spinning. Setting this parameter to a large value disables this reset feature.

8.13 att_control_params

kp_roll_moment – *required*, `float`, grams*meters/rad, 23.084

Proportional gain for roll (+x) axis attitude controller. This gain multiplies the error in roll angle. The output of the controller is the roll axis moment in units of grams*meters.

kp_pitch_moment – *required*, `float`, grams*meters/rad, 23.084

Proportional gain for pitch (+y) axis attitude controller. This gain multiplies the error in pitch angle. The output of the controller is the pitch axis moment in units of grams*meters.

kpy – *required*, `float`, grams/rad, 64.8

Proportional gain for yaw (+z) axis attitude controller. This gain multiplies the error in yaw angle. The output of the controller is the differential individual motor thrust that is added to each clockwise-rotating motor and subtracted from each counterclockwise-rotating motor.

kd_roll_moment – *required*, `float`, grams*meters/(rad/sec), 2.6508

Derivative gain for roll (+x) axis attitude controller. This gain multiplies the error in roll angular velocity. The output of the controller is the roll axis moment in units of grams*meters.

kd_pitch_moment – *required*, `float`, grams*meters/(rad/sec), 2.6508

Derivative gain for pitch (+y) axis attitude controller. This gain multiplies the error in pitch angular velocity. The output of the controller is the pitch axis moment in units of grams*meters.

kdy – *required*, `float`, grams/(rad/sec), 10.8

Derivative gain for yaw (+z) axis attitude controller. This gain multiplies the error in yaw angular velocity. The output of the controller is the differential individual motor thrust that is added to each clockwise-rotating motor and subtracted from each counterclockwise-rotating motor.

prop_rpm_thrust_curve_a0 – *required*, `float`, grams, -2.1723

Propeller constants to get the thrust curve of a single propeller using the following format:

$thrust = a_2 * rpm^2 + a_1 * rpm + a_0$

prop_rpm_thrust_curve_a1 – *required*, `float`, grams/rpm, 0.00047332

See **prop_rpm_thrust_curve_a0**.

prop_rpm_thrust_curve_a2 – *required*, `float`, grams/rpm/rpm, 4.0172e-07

See **prop_rpm_thrust_curve_a0**.

min_rpm – *required*, `int16_t`, rpm, 5500

Minimum commanded motor RPM.

max_rpm – *required*, `int16_t`, rpm, 16400

Maximum commanded motor RPM.

number_of_props – *required*, `int16_t`, unitless, 4

Number of propellers on vehicle. The supported number of propellers is 4 or 6.

use_max_rpm_curve – `int`, unitless, 0, [0, 1]

1 is enabled, 0 is disabled. This parameter allows the vehicle to command higher RPMs as a function of the vehicle's measured voltage. If disabled, the maximum commanded motor RPM is simply **max_rpm**. If enabled, the maximum RPM is the minimum of **max_rpm** and the rpm calculated by (**max_rpm_curve_a0** + **max_rpm_curve_a1** × voltage).

max_rpm_curve_a0 – `float`, rpm, 12200

See **use_max_rpm_curve**.

max_rpm_curve_a1 – `float`, rpm/volt, 0

See **use_max_rpm_curve**.

standard_motor_directions – `int`, unitless, 1, [0, 1]

If true (1) the standard convention for motor directions is used. The standard directions for a quadrotor are clockwise for motors 0 and 2 and counterclockwise for motors 1 and 3. If false (0) the reverse convention for motor directions is used. The reverse directions for a quadrotor are counterclockwise for motors 0 and 2 and clockwise for motors 1 and 3.

prop_config_type – *required*, `int16_t`, unitless, 0, [0, 2]

Specifies the propeller configuration type and how the **prop_config_Dx**, **prop_config_Dy**, and **prop_config_beta** parameters are interpreted. 0 is the quadrotor rectangle configuration, 1 is the quadrotor rhombus configuration, and 2 is the hexrotor regular hexagon configuration. See Section 9.2.6 for details.

prop_config_Dx – *required*, `float`, meters, 0.071, [0.001, –]

See **prop_config_type**.

prop_config_Dy – *required*, `float`, meters, 0.071, [0.001, –]

See **prop_config_type**.

prop_config_beta – `float`, rad, 0

See **prop_config_type**.

ang_vel_z_ff_factor – `float`, unitless, 1.0

Desired angular velocity feedforward factor for yaw control (Z-axis control). Value of zero eliminates desired yaw angular velocity feedforward component for attitude controller. Value of 1.0 corresponds to full feedforward control.

8.14 compass_calib_params

The compass_calib_params parameters are used for compass calibration. The software fits a three-dimensional ellipsoid to magnetometer data that is collected while the vehicle is rotated through a certain sequence described in Section 6.4. After the data is collected, several checks on the collected magnetometer data and the ellipsoid fit are performed to determine if the calibration passed. These checks are controlled by the following parameters.

min_range_each_axis – `float`, microteslas, 35, [0.0, –]

Raw compass field data collected during the compass calibration procedure must have a minimum range (i.e., max value - min value) for the X, Y, and Z axes.

min_corrected_magnitude – `float`, calibrated compass units, 0.7, [0.0, 1.0]

The ellipsoid fitting algorithm attempts to map the raw compass field data to a unit sphere (i.e., a sphere with a radius of 1.0). This check requires that no collected magnetometer data is smaller in vector magnitude than this value after the calculated calibration is applied.

max_corrected_magnitude – `float`, calibrated compass units, 1.3, [1.0, –]

The ellipsoid fitting algorithm attempts to map the raw compass field data to a unit sphere (i.e., a sphere with a radius of 1.0). This check requires that no collected magnetometer data is larger in vector magnitude than this value after the calculated calibration is applied.

max_axes_ratio – `float`, unitless, 2.0, [1.0, –]

The ratio of the length of longest ellipsoid axis to the length of the shortest ellipsoid axis must be less than this value. If the ellipsoid is too oblong, this check fails.

8.15 imu_linear_calib_params

The imu_linear_calib_params parameters are used for gyroscope and accelerometer temperature calibration. See Section 6.1 for the procedure.

group_size – `int`, number of samples, 2000

Number of gyroscope and accelerometer samples per group for the temperature calibration, one sample is collected per loop at a rate of 500 Hz. For example, if this value is set to 2000 then data for each group is collected over a period of 4 seconds.

num_groups – `int`, number of groups, 70, [2, 70]

In conjunction with **group_size**, this parameter determines the total time for the gyroscope and accelerometer temperature calibration. For example, if this parameter is 70 and **group_size** is 2000, the total calibration time is 70 x 4 = 280 seconds. The value of **num_groups** must be less than or equal to 70.

min_req_temp_diff – `float`, degrees Celsius, 10.0

Minimum temperature change required for calibration to be successful.

max_x_acc_residual – `float`, Gravity^2, 2.5e-06

Maximum residual on X accelerometer data allowed for calibration to be successful.

max_y_acc_residual – `float`, Gravity^2, 2.5e-06

Maximum residual on Y accelerometer data allowed for calibration to be successful.

max_z_acc_residual – `float`, Gravity^2, 1.5e-05

Maximum residual on Z accelerometer data allowed for calibration to be successful.

max_x_gyro_residual – `float`, (rad/sec)^2, 2.5e-06

Maximum residual on X gyroscope data allowed for calibration to be successful.

max_y_gyro_residual – `float`, (rad/sec)^2, 2.5e-06

Maximum residual on Y gyroscope data allowed for calibration to be successful.

max_z_gyro_residual – `float`, (rad/sec)^2, 1.5e-06

Maximum residual on Z gyroscope data allowed for calibration to be successful.

require_level_temp_calibration – `int`, unitless, 1, [0, 1]

1 is enabled, 0 is disabled. If enabled, the temperature calibration must be run with the vehicle level and in the upright position. If disabled, the temperature calibration can be performed at an arbitrary orientation.

max_xy_accel_magnitude – `float`, Gravity, 0.5

If **require_level_temp_calibration** is enabled, the maximum X or Y accelerometer value magnitude allowed for successful calibration.

min_z_accel_value – `float`, Gravity, 0.75

If **require_level_temp_calibration** is enabled, the minimum Z accelerometer value magnitude allowed for successful calibration.

max_z_accel_value – `float`, Gravity, 1.25

If **require_level_temp_calibration** is enabled, the maximum Z accelerometer value magnitude allowed for successful calibration.

8.16 in_flight_detector_params

in_flight_thrust_threshold_factor – `float`, unitless, 0.5, [0, 0.8]

The vehicle can be considered to be in a landing state if the commanded thrust is less than this value multiplied the **basethrust** value in the **position_control_params** group. Note that additional sensors are used to confirm that the vehicle is in a landing state.

8.17 no_fly_zone_detector_params

For more information on the no-fly zone feature, see Section 11.4.

enable_safe_circle – `int`, unitless, 0, [0, 1]

Enables (1) or disables (0) the safe circle. Use the other parameters in this section to define the safe circle zone.

safe_lat_center – `int32_t`, degrees*1e7, 399421032

Coordinate of the center of the safe circle in degrees of latitude in which the vehicle is allowed to fly using the safe circle feature.

safe_lon_center – `int32_t`, degrees*1e7, -752008578

Coordinate of the center of the safe circle in degrees of longitude in which the vehicle is allowed to fly using the safe circle feature.

safe_circle_radius – `float`, meters, 5000.0

Radius of safe circle in meters where vehicle is allowed to fly.

8.18 vio_data_params

vio_data_is_invalid_after_critical_error – `int`, 1, [0, 1]

If true (1) VIO data is considered invalid for the remainder of a flight if a critical VIO error occurs during the flight. Critical VIO errors include significant data timeouts, detection of poor VIO performance, and other errors defined in this section. VIO data can be considered valid again after the propellers are stopped. If false (0), it is possible for VIO data to be considered valid during a flight after a critical VIO error occurs.

min_vio_feature_count – `uint16_t`, , 4

See comments for **vio_feature_timeout_time** and **vio_feature_relock_time**.

vio_feature_timeout_time – `uint32_t`, us, 1000000

VIO data is deemed invalid if fewer than **min_vio_feature_count** are observed for this period of time.

vio_feature_relock_time – `uint32_t`, us, 1000000

After VIO data is deemed invalid due to tracking too few features, VIO data can become valid if at least **min_vio_feature_count** features are observed for this period of time.

max_velocity_covariance – `float`, $(m/s)^2$, 0.12

VIO data is deemed invalid if any diagonal term of the velocity covariance matrix exceeds this value. This is considered a critical VIO error, see **vio_data_is_invalid_after_critical_error** parameter description.

max_extended_velocity_covariance – `float`, $(m/s)^2$, 0.01

VIO data is deemed invalid if any diagonal term of the velocity covariance matrix exceeds this value for the period of time defined by **max_extended_velocity_covariance_time**. This is considered a critical VIO error, see **vio_data_is_invalid_after_critical_error** parameter description.

max_extended_velocity_covariance_time – `uint32_t`, us, 1000000

See description for **max_extended_velocity_covariance**.

8.19 attitude_estimator_params

config_mag_use_for_yaw_drift – `int`, unitless, 1, [0, 2]

The estimate of the yaw angle might drift slightly over time due to biases in gyroscope data. Use the magnetometer to correct the small drift in the yaw angle estimate. This feature requires the magnetic field in which the vehicle operates to be constant. If the field is changing or if the magnetometer is not well-calibrated, using magnetometer data to estimate the yaw drift results in poor yaw angle estimation and control.

This parameter configures how the magnetometer data is used to correct yaw drift as follows:

- 0 – Magnetometer data is never used to correct yaw drift
- 1 – Magnetometer data is used to correct yaw drift only in GPS modes
- 2 – Magnetometer data is used to correct yaw drift whenever magnetometer data is available

config_vio_use_for_yaw_drift – `int`, unitless, 1, [0, 2]

The estimate of the yaw angle may drift slightly over time due to biases in gyroscope data. VIO, which uses vision data, can be used to correct for the small drift in the yaw angle estimate.

This parameter configures how the VIO data is used to correct yaw drift as follows:

- 0 – VIO data is never used to correct yaw drift
- 1 – VIO data is used to correct yaw drift only in VIO modes
- 2 – VIO data is used to correct yaw drift whenever VIO data is available

mag_has_priority_over_vio – `int`, unitless, 1, [0, 1]

VIO data or magnetometer data can be used to correct yaw drift. See documentation for **config_mag_use_for_yaw_drift** and **config_vio_use_for_yaw_drift**. This parameter configures which data is used to correct yaw drift. The following are true if both VIO data and magnetometer data are available, and the configuration options indicate that both should be used:

- 0 – VIO data has priority over magnetometer data to correct yaw drift
- 1 – Magnetometer data has priority over VIO data to correct yaw drift

8.20 orientation_params

imu_R00 – *required*, `float`, unitless, 1.0

Description applies to parameters **imu_R00** through **imu_R22**. Rotation matrix elements (row, column) to specify accelerometer orientation and gyroscope on the vehicle.

The axis orientation is as follows:

- +x is forward on the vehicle between the two front propellers.
- +y is to the left on the vehicle.
- +z is vertically up on the vehicle.

This rotation matrix is from the IMU frame to the vehicle frame. That is, if this rotation matrix is post-multiplied by a vector in the IMU frame, the result is in the vehicle frame.

The nine elements of this matrix must have the properties of a rotation matrix to be accepted by the flight controller: $R^\top R = I$ and $|det(R)| = 1$.

imu_R01 – *required*, `float`, unitless, 0.0

imu_R02 – *required*, `float`, unitless, 0.0

imu_R10 – *required*, `float`, unitless, 0.0

imu_R11 – *required*, `float`, unitless, -1.0

imu_R12 – *required*, `float`, unitless, 0.0

imu_R20 – *required*, `float`, unitless, 0.0

imu_R21 – *required*, `float`, unitless, 0.0

imu_R22 – *required*, `float`, unitless, -1.0

mag_R00 – *required*, `float`, unitless, -1.0

Description applies to all parameters **mag_R00** through **mag_R22**. Nine rotation matrix elements (row, column) that specify the magnetometer orientation on the vehicle. The functionality is equivalent to the **imu_R00** through **imu_R22** parameters.

mag_R01 – *required*, `float`, unitless, 0.0

mag_R02 – *required*, `float`, unitless, 0.0

mag_R10 – *required*, `float`, unitless, 0.0

mag_R11 – *required*, `float`, unitless, 1.0

mag_R12 – *required*, `float`, unitless, 0.0

mag_R20 – *required*, `float`, unitless, 0.0

mag_R21 – *required*, `float`, unitless, 0.0

mag_R22 – *required*, `float`, unitless, -1.0

downward_cam_R00 – `float`, unitless, 0.0

Description applies to all parameters **downward_cam_R00** through **downward_cam_R11**. Four rotation matrix elements (row, column) that specify the downward-facing camera orientation on the vehicle. This rotation matrix is from the camera pixel flow to the vehicle frame. That is, if this rotation matrix is post-multiplied with by a vector containing the X and Y pixel flow, the result is the X and Y velocity in the vehicle frame.

The camera frame is defined as follows:

- +x is right in the image.
- +y is down in the image.

The four elements of this matrix must have the properties of a rotation matrix to be accepted by the flight controller: $R^\top R = I$ and $|det(R)| = 1$.

If the camera orientation is unknown, one could determine this transform experimentally by observing the `pixel_flow` field in the `optic_flow_0_raw` group in `snav_inspector`. For example:

- Vehicle translates in the positive X direction with respect to the vehicle frame (forward). The X component of `pixel_flow` is close to zero and the Y component of `pixel_flow` is a negative number. Since the `pixel_flow` is in the opposite direction of the camera's motion, this corresponds to a positive Y translation of the camera in the camera frame as estimated by optic flow.
- Vehicle translates in the positive Y direction with respect to the vehicle frame (left). The X component of `pixel_flow` is a negative number and the Y component of `pixel_flow` is close to zero. Since the `pixel_flow` is in the opposite direction of the camera's motion, this corresponds to a positive X translation of the camera in the camera frame as estimated by optic flow.
- The positive X axis of the camera is aligned to the positive Y axis of the vehicle frame and the positive Y axis of the camera frame is aligned to the positive X axis of the vehicle frame as follows:
  - **downward_cam_R00** = 0.
  - **downward_cam_R01** = 1.
  - **downward_cam_R10** = 1.
  - **downward_cam_R11** = 0.

downward_cam_R01 – `float`, unitless, -1.0

downward_cam_R10 – `float`, unitless, -1.0

downward_cam_R11 – `float`, unitless, 0.0

vehicle_center_to_imu_x – `float`, meters, 0.0

Distance from the geometric center of the vehicle to the IMU along the +x axis. The +x axis is forward on the vehicle.

vehicle_center_to_imu_y – `float`, meters, 0.0

Distance from the geometric center of the vehicle to the IMU along the +y axis. The +y axis is left on the vehicle.

vehicle_center_to_imu_z – `float`, meters, 0.0

Distance from the geometric center of the vehicle to the IMU along the +z axis. The +z axis is vertically up on the vehicle.

8.21 esc_interface_params

sequential_spinup – `int`, unitless, 1, [0, 1]

1 is enabled, 0 is disabled. If enabled, each motor starts sequentially (e.g., 0, 1, 2, 3) when propeller spinup is commanded. If disabled, all motors start simultaneously.

esc_protocol – *required*, `EscProtocol`, unitless, 3

Protocol used for ESC communication. The ESC_200QC_PROTOCOL is 3.

require_same_sw_versions – `int`, unitless, 1, [0, 1]

Enables (1) or disables (0) the requirement of all ESCs with matching software versions. If this value is set to 1 and the ESC software versions do not match, Snapdragon Navigator does not initialize successfully and reports `MOTORS_SW_VERSION_ERROR`.

8.22 esc_mapping_params

Flight controller internally assumes that the front left (FL) of the vehicle has motor ID=0, rear left (RL) -> ID=1, rear right (RR) -> ID=2, and front right (FR) -> ID=3. To avoid confusion, it is recommended to maintain this mapping by connecting the ESCs with the correct IDs to the motors in this specific order.

If the actual mapping is not 0,1,2,3 (CCW starting from front left), esc_mapping_params can be used to correct the ESC mapping. Parameters id0, id1, id2, id3 specify the ID of the ESC that is connected to the motor in location specified by number in the id parameter. For example, if the front two ESCs switch places, the mapping is id0=3, id1=1, id2=2, id3=0 because ESC3 is connected to motor0 (front left) and ESC0 is connected to motor3 (front right).

Similarly, dir0, dir1, dir2, dir3 specify whether direction should be 1 or -1 for the motor in position 0,1,2 or 3, independently of which ESC id is actually driving this motor. If the motor in position 0 is spinning in the wrong direction, reverse the dir0 parameter.

The following parameters affect the order and sign of ESC commands sent to ESCs in the UART packet:

id0 – `int8_t`, 0, [-1, 7]

id1 – `int8_t`, 1, [-1, 7]

id2 – `int8_t`, 2, [-1, 7]

id3 – `int8_t`, 3, [-1, 7]

dir0 – `int8_t`, 1, [-1, 1]

dir1 – `int8_t`, 1, [-1, 1]

dir2 – `int8_t`, 1, [-1, 1]

dir3 – `int8_t`, 1, [-1, 1]

8.23 baro_data_params

max_valid_barometer_pressure – `float`, Pascals, 120000.0
Barometer data is invalid if the barometer pressure is greater than this value.

min_valid_barometer_pressure – `float`, Pascals, 20000.0
Barometer data is invalid if the barometer pressure is less than this value.

max_valid_barometer_temperature – `float`, degrees Celsius, 120.0
Barometer data is invalid if the barometer temperature is greater than this value.

min_valid_barometer_temperature – `float`, degrees Celsius, -40.0
Barometer data is invalid if the barometer temperature is less than this value.

8.24 rc_receiver_params

is_binding_supported – `int`, 0, [0, 1]
Flag (0 or 1) to enable binding of RC by holding the vehicle upside down during initialization

bind_mode – `SnRcReceiverMode`, SN_SPEKTRUM_MODE_DSM2_11
Desired binding mode for Spektrum RC. Choose from: `SN_SPEKTRUM_MODE_DSM2_22(1)`, `SN_SPEKTRUM_MODE_DSM2_11(2)`, `SN_SPEKTRUM_MODE_DSMX_22(3)`, `SN_SPEKTRUM_MODE_DSMX_11(4)`. Use integer values to specify the desired bind mode in the parameter file.

power_ctrl_pin – `int32_t`, 69
GPIO pin for power control of the RC receiver (power cycling of RC receiver is required for binding purposes).

bind_pin – `int32_t`, 50
GPIO pin for binding of the RC receiver. This pin is the same as that used for the data RX into the DSP. For binding purposes, this pin is temporarily used as a GPIO to send the binding pulse sequence to the Spektrum receiver.

8.25 rc_params

max_angle_mode_tilt_angle – `float`, radians, 0.872664625997, [0.0, 1.23]
Maximum total roll and pitch tilt angle command for angle control modes.

max_low_angle_mode_tilt_angle – `float`, radians, 0.4, [0.0, 1.23]
Maximum total roll and pitch tilt angle command for "low angle" angle control modes. This value must be smaller than **max_tilt_angle**.

landing_height_threshold_1 – `float`, meters, 7.0
See **landing_speed_1**, **landing_speed_2**, and **landing_speed_3**.

landing_height_threshold_2 – `float`, meters, 4.0
See **landing_speed_1**, **landing_speed_2**, and **landing_speed_3**.

landing_speed_1 – `float`, meters/second, -2.0, [–, -0.1]

This parameter is relevant when landing after Return to Home and when Landing with GPS. The vehicle descends at this speed when the vehicle is above the altitude defined by the **landing_height_threshold_1** parameter as measured by the barometer.

landing_speed_2 – `float`, meters/second, -0.75, [–, -0.1]

This parameter is relevant when landing after Return to Home and when Landing with GPS. The vehicle descends at this speed when the vehicle is above the altitude defined by the **landing_height_threshold_2** parameter and below the altitude defined by the parameter **landing_height_threshold_1** as measured by the barometer.

landing_speed_3 – `float`, meters/second, -0.5, [–, -0.1]

This parameter is relevant when landing after Return to Home and when Landing with GPS. The vehicle descends at this speed when the vehicle is below the altitude defined by the **landing_height_threshold_2** parameter as measured by the barometer.

go_home_speed – `float`, meters/second, 3.0, [0.2, –]

This parameter is relevant for the Return to Home process. The vehicle is commanded to fly at this lateral speed towards the home location when the vehicle is at a distance greater than **go_home_slow_radius** from the home location.

go_home_speed_slow – `float`, meters/second, 2.0, [0.2, –]

This parameter is relevant for the Return to Home process. The vehicle is commanded to fly at this lateral speed towards the home location when the vehicle is at a distance less than **go_home_slow_radius** from the home location.

go_home_slow_radius – `float`, meters, 7.0

See documentation for **go_home_speed** and **go_home_speed_slow**.

go_home_max_advance_distance – `float`, meters, 9.0

This parameter is relevant for the Return to Home process. The position control set point only advances if the lateral position error is less than this value. This feature prevents the set point from getting too far ahead of the vehicle if the vehicle cannot keep up with the desired position.

enable_go_home_ascending – `int`, unitless, 1, [0, 1]

If enabled (1), the vehicle ascends to **go_home_altitude** at **go_home_ascent_rate** if below **go_home_altitude** during Return to Home. If disabled (0), the vehicle stays at the current altitude while returning to home.

go_home_altitude – `float`, meters, 3.0

The vehicle ascends to this height during return to home if **enable_go_home_ascending** is enabled.

go_home_ascent_rate – `float`, meters/second, 0.5

During return to home, the vehicle ascends at this rate to **go_home_altitude** if **enable_go_home_ascending** is enabled.

max_roll_pitch_rate – `float`, rad/sec, 6.28318530718, [0, –]

Maximum rate of change of total roll and pitch tilt angle command in angle control modes.

stop_props_rc_value – `uint16_t`, rc units, 154, [0, 1023]

When using RC control input, propellers are commanded to stop when the RC thrust channel value is less than this parameter and the yaw stick is moved all the way to the left OR all the way to the right. This low RC thrust command is normally achieved by using the "throttle cut" feature on the RC. The vehicle does not respond to yaw commands when the RC thrust channel value is less than this parameter.

yaw_acc_limit – `float`, rad/sec/sec, 5.5

Maximum change in the rate of desired yaw angular velocity. This parameter limits the acceleration rate of the desired yaw angle setpoint.

min_thrust – *required*, `float`, grams, 60.0

Minimum commanded total vehicle thrust. To increase the actual minimum vehicle thrust limitation, use the minimum RPM value.

max_thrust – *required*, `float`, grams, 460.0

Maximum commanded total vehicle thrust. To increase the actual maximum vehicle thrust limitation, use the maximum RPM value.

emergency_landing_thrust_factor – `float`, unitless, 0.4, [0, 0.75]

A low fixed thrust value is applied when the vehicle enters Emergency Landing mode. The fixed thrust value is equal to this value multiplied by the **basethrust** parameter in the **position_control_params** group.

pressure_landing_mode_z_velocity – `float`, meters/second, -0.75, [–, 0.0]

The fixed velocity used when the vehicle enters Landing with Barometer mode. This value must be less than 0.

max_rc_ch_val_dimensionless – `uint16_t`, rc units, 852

Maximum value of the RC channel used to convert to +1.0 non-dimensional input.

min_rc_ch_val_dimensionless – `uint16_t`, rc units, 171

Minimum value of the RC channel used to convert to -1.0 non-dimensional input.

geotether_radius – `float`, meters, 150.0

In GPS Position Control mode, this parameter is the maximum radial displacement in meters from the startup location for the commanded X and Y position. Users cannot fly outside of this circular region. See Section 11.3 for more information on this feature.

8.26 input_interpreter_params

rate_mode_roll_pitch_gain – `float`, rad/s, 10.0, [0.0, –]

Rate mode roll rate and pitch rate gain for non-dimensional input.

rate_mode_roll_pitch_deadband – `float`, unitless, 0.023, [0.0, 0.5]

Rate mode roll rate and pitch rate deadband for non-dimensional input.

rate_mode_roll_pitch_enable_expo – `int`, unitless, 1, [0, 1]

Enables (1) or disables (0) use of expo on roll and pitch non-dimensional inputs in rate mode. See the description for **rate_mode_roll_pitch_expo_value**.

rate_mode_roll_pitch_expo_value – `float`, unitless, 1.3, [0.0, –]

Modifies roll and pitch non-dimensional inputs in rate mode when enabled by the **rate_mode_roll_pitch_enable_expo** parameter. The magnitude of the roll and pitch non-dimensional inputs are modified according to:

$$\text{output\_value} = \text{input\_value}^{\textbf{rate\_mode\_roll\_pitch\_expo\_value}}$$

If this parameter is set to 1.0, there is no change in the output value. Setting this parameter to a value greater than 1.0 results in a "flatter" response near an input value of 0. After this remapping, non-dimensional input remains a value between -1.0 and 1.0.

rate_mode_yaw_gain – `float`, rad/s, 3.14159265359, [0.0, –]
Rate mode yaw rate gain for non-dimensional input.

rate_mode_yaw_deadband – `float`, unitless, 0.023, [0.0, 0.5]
Rate mode yaw rate deadband for non-dimensional input.

angle_mode_roll_gain – `float`, rad, 0.872664625997, [0.0, –]
Angle mode roll angle gain for non-dimensional input.

angle_mode_roll_deadband – `float`, unitless, 0.0, [0.0, 0.5]
Angle mode roll angle deadband for non-dimensional input.

angle_mode_pitch_gain – `float`, rad, 0.872664625997, [0.0, –]
Angle mode pitch angle gain for non-dimensional input.

angle_mode_pitch_deadband – `float`, unitless, 0.0, [0.0, 0.5]
Angle mode pitch angle deadband for non-dimensional input.

angle_mode_yaw_gain – `float`, rad/s, 3.14159265359, [0.0, –]
Yaw rate gain for non-dimensional input for Angle mode, GPS mode, Optic Flow mode, and VIO mode.

angle_mode_yaw_deadband – `float`, unitless, 0.023, [0.0, 0.5]
Yaw angle deadband for non-dimensional input for Angle mode, GPS mode, Optic Flow mode, and VIO mode.

low_angle_mode_roll_gain – `float`, rad, 0.4, [0.0, –]
Low angle mode roll angle gain for non-dimensional input.

low_angle_mode_roll_deadband – `float`, unitless, 0.0, [0.0, 0.5]
Low angle mode roll angle deadband for non-dimensional input.

low_angle_mode_pitch_gain – `float`, rad, 0.4, [0.0, –]
Low angle mode pitch angle gain for non-dimensional input.

low_angle_mode_pitch_deadband – `float`, unitless, 0.0, [0.0, 0.5]
Low angle mode pitch angle deadband for non-dimensional input.

gps_mode_xy_gain – `float`, m/s, 4.7, [0.0, –]
GPS mode horizontal velocity gain for non-dimensional input. This value is also the maximum magnitude of the total X-Y GPS velocity command, larger magnitudes are capped at this value.

gps_mode_xy_deadband – `float`, unitless, 0.023, [0.0, 0.5]
GPS mode horizontal velocity deadband for non-dimensional input.

optic_flow_mode_xy_gain – `float`, m/s, 1.5, [0.0, –]

Optic Flow mode horizontal velocity gain for non-dimensional input. This value is also the maximum magnitude of the total X-Y Optic Flow velocity command, larger magnitudes are capped at this value.

optic_flow_mode_xy_deadband – `float`, unitless, 0.023, [0.0, 0.5]

Optic Flow mode horizontal velocity deadband for non-dimensional input.

vio_mode_xy_gain – `float`, m/s, 3.3, [0.0, –]

VIO mode horizontal velocity gain for non-dimensional input. This value is also the maximum magnitude of the total X-Y VIO velocity command, larger magnitudes are capped at this value.

vio_mode_xy_deadband – `float`, unitless, 0.023, [0.0, 0.5]

VIO mode horizontal velocity deadband for non-dimensional input.

pos_hold_mode_xy_gain – `float`, m/s, 4.7, [0.0, –]

Position Hold mode horizontal velocity gain for non-dimensional input.

pos_hold_mode_xy_deadband – `float`, unitless, 0.023, [0.0, 0.5]

Position Hold mode horizontal velocity deadband for non-dimensional input.

height_control_z_vel_gain – `float`, m/s, 1.0, [0.0, –]

Height control mode vertical velocity gain for non-dimensional input. This value is applied to optic flow, GPS, and other height control modes.

height_control_z_vel_deadband – `float`, unitless, 0.25

Height control mode vertical velocity deadband for non-dimensional input. This value is applied to optic flow, GPS, and other height control modes.

enable_non_dim_thrust_curve – `int`, unitless, 0, [0, 1]

This parameter controls how the thrust command is calculated from the non-dimensional thrust input (i.e., cmd2) in thrust controlled modes. If enabled (1), user-specified polynomial constants are used:

$$\text{thrust command} = a2 * \text{cmd2}^2 + a1 * \text{cmd2} + a0$$

In which a2, a1, a0 correspond to parameters **non_dim_input_thrust_curve_a2**, **non_dim_input_thrust_curve_a1**, and **non_dim_input_thrust_curve_a0**.

If disabled (0), the flight controller internally calculates a linear mapping in which non-dimensional thrust inputs of -1.0 and +1.0 respectively correspond to **min_thrust** and **max_thrust** from the **rc_params** group.

non_dim_input_thrust_curve_a0 – `float`, grams, 0.0

Parameter is active only if **enable_non_dim_thrust_curve** is enabled. See the documentation for **enable_non_dim_thrust_curve**.

non_dim_input_thrust_curve_a1 – `float`, grams/non-dimensional input, 0.0

Parameter is active only if **enable_non_dim_thrust_curve** is enabled. See the documentation for **enable_non_dim_thrust_curve**.

non_dim_input_thrust_curve_a2 – `float`, grams/(non-dimensional input)^2, 0.0

Parameter is active only if **enable_non_dim_thrust_curve** is enabled. See the documentation for **enable_non_dim_thrust_curve**.

8.27 calib_optic_flow_cam_yaw_params

max_calib_time_us – `int64_t`, us, 60000000, [0, –]

Maximum amount of time allowed for the calibration procedure. If the calibration procedure is not completed in this amount of time, it fails.

enforce_max_ang_vel_z_limit – `int`, 1, [0, 1]

If enabled (1), a limit on the Z component of angular velocity is enforced during the calibration procedure, defined by **max_ang_vel_z**.

max_ang_vel_z – `float`, rad/s, 1.5, [0, –]

Maximum angular velocity magnitude allowed for the Z axis. If the absolute value of the Z component of angular velocity exceeds this threshold and **enforce_max_ang_vel_z_limit** is set to 1, the calibration procedure fails.

enforce_xy_movement_checker – `int`, 1, [0, 1]

If enabled (1), a limit on the X and Y components of angular velocity is enforced during the calibration procedure to prevent undesired movement about these axes. The limit is defined by **movement_checker_max_rate**.

movement_checker_max_rate – `float`, rad/s, 0.1, [0, –]

Defines the threshold on the absolute value of the X and Y components of angular velocity for the movement checker. If the movement checker detects motion about these axes and **enforce_xy_movement_checker** is set to 1, the calibration procedure fails.

required_rotation_z – `float`, radians, 12.5663706144, [0, –]

Required rotation magnitude about Z for the calibration to be completed.

max_valid_x_calib – `float`, px/rad, 50, [0, –]

Maximum magnitude of the X result of the calibration to be considered valid. If the absolute value of the X result exceeds this value, the calibration fails.

max_valid_y_calib – `float`, px/rad, 50, [0, –]

Maximum magnitude of the Y result of the calibration to be considered valid. If the absolute value of the Y result exceeds this value, the calibration fails.

8.28 user_input_handler_params

require_spektrum – `int`, 0, [0, 1]

If enabled (1), a Spektrum RC must be connected for flight to be permitted, even if commands are being received through the API.

spektrum_auto_override – `int`, 0, [0, 1]

If enabled (1), a connected Spektrum RC always takes precedence over any commands received through the API.

low_thrust_bound – `float`, unitless, -0.94

Propellers can be commanded to stop automatically when vehicle has landed and non-dimensional thrust/vertical speed command (i.e., cmd2) is less than this value.

enable_vio_mode – `int`, 1, [0, 1]
Controls whether VIO mode is accessible.

8.29 device_drivers_params

The flight controller dynamically loads software drivers for the following devices: imu0 (main flight IMU), baro0 (barometer), mag0 (magnetometer), sonar0 (ultrasonic sensor), gnss0 (GNSS receiver), rc0 (RC receiver), vsense0 and isense0 (voltage and current sensing). ESC driver is built into the flight controller.

The device_drivers_params parameters specify file names of dynamic libraries (.so) to be loaded for each device (parameters that end with _file_name). Multiple devices can be handled by a single driver, so they use the same library (for example, baro0, vsense0, isense0 in default configuration). To disable a device from starting, provide an empty string or non-existent file name as the device driver file name.

Similarly, each device has a device path, mapped to a I2C, SPI, or UART port (parameters tha end with _device_path). These parameters must specify the correct device string, depending on the electrical connections.

The communication bit rate can also be specified using the _bit_rate parameters for each device.

The device path and bit rate parameters are passed directly to the device driver and handled there.

enable_external_drivers – `int`, 1, [0, 1]
This parameter is obsolete and will be removed in the next release. Dynamically loadable drivers are always used. Do not change.

imu0_driver_file_name – `string`, libimu0_mpu_spi_driver.so
File name of the driver for the flight IMU (imu0).

rc0_driver_file_name – `string`, librc0_spektrum_driver.so
File name of the RC receiver driver.

baro0_driver_file_name – `string`, libbaro0_bmp280_vsense0_ltc2946_driver.so
File name of the barometer driver.

mag0_driver_file_name – `string`, libmag0_hmc5883l_driver.so
File name of the magnetometer driver.

sonar0_driver_file_name – `string`, libsonar0_maxbotix_driver.so
File name of the ultrasonic sensor driver.

gnss0_driver_file_name – `string`, libgnss0_ublox_driver.so
File name of the GNSS0 receiver driver.

vsense0_driver_file_name – `string`, libbaro0_bmp280_vsense0_ltc2946_driver.so
File name of the voltage sensor driver.

isense0_driver_file_name – `string`, libbaro0_bmp280_vsense0_ltc2946_driver.so
File name of the current sensor driver.

imu0_device_path – `string`, /dev/spi-1
Device path for imu0.

rc0_device_path – `string`, /dev/tty-1
Device path for the RC receiver.

esc0_device_path – `string`, /dev/tty-2
Device path for ESC communication.

baro0_device_path – `string`, /dev/iic-3
Device path for the barometer.

mag0_device_path – `string`, /dev/iic-8
Device path for the magnetometer.

sonar0_device_path – `string`, /dev/tty-4
Device path for the ultrasonic sensor.

gnss0_device_path – `string`, /dev/tty-3
Device path for the GNSS0 receiver.

vsense0_device_path – `string`, /dev/iic-3
Device path for the voltage sensor.

isense0_device_path – `string`, /dev/iic-3
Device path for the current sensor.

imu0_bit_rate – `int32_t`, 10000000
Communication bit rate for imu0.

rc0_bit_rate – `int32_t`, 115200
Communication bit rate for the RC receiver.

esc0_bit_rate – `int32_t`, 250000
Communication bit rate for the ESC.

baro0_bit_rate – `int32_t`, 400000
Communication bit rate for the barometer.

mag0_bit_rate – `int32_t`, 400000
Communication bit rate for the magnetometer.

sonar0_bit_rate – `int32_t`, 9600
Communication bit rate for the ultrasonic sensor.

gnss0_bit_rate – `int32_t`, 9600
Communication bit rate for the GNSS0 receiver.

vsense0_bit_rate – `int32_t`, 400000
Communication bit rate for the voltage sensor.

isense0_bit_rate – `int32_t`, 400000
Communication bit rate for the current sensor.

8.30 esc_firmware_params

firmware_file_name – `string`, esc_firmware.bin

Filename of the ESC firmware to be used for firmware update.

config_file_name – `string`, no_file.bin

Filename of the ESC configuration to be used for configuration update.

bootloader_bit_rate – `uint32_t`, 38400

UART bit rate for ESC firmware update.

config_bit_rate – `uint32_t`, 57600

UART bit rate for updating ESC configuration (firmware bit rate). This is normally the same as the standard ESC communication bit rate, but if there is no configuration file uploaded to firmware, it defaults to 57600.

force_firmware_update – `int`, 0, [0, 1]

Set this parameter to 1 to execute the firmware update (disabled by default).

force_config_update – `int`, 0, [0, 1]

Set this parameter to 1 to execute the configuration update (disabled by default).

8.31 soft_landing_params

enabled – `int`, unitless, 1, [0, 1]

Controls whether or not the Soft Landing feature is enabled (1) or disabled (0). The Soft Landing feature limits the maximum descent speed as a function of the estimated height above ground level (AGL) by applying a constant deceleration.

max_descent_speed_at_min_height_agl – `float`, m/s, 0.15, [0.05, 0.75]

The maximum allowed descent speed at **min_height_agl** and below.

min_height_agl – `float`, m, 0.5, [0, 1]

The height above ground level at which the Soft Landing feature attempts to reduce the maximum allowed descent speed to **max_descent_speed_at_min_height_agl**.

acceleration_magnitude – `float`, m/s/s, 0.75, [0.1, –]

Specifies the magnitude of the constant deceleration (negative acceleration) applied to the descent speed as a function of height AGL.

9 Basic Parameter Tuning

This Chapter includes guidelines for modifying runtime parameters for a vehicle. These runtime parameters are defined in the `snav_params.xml` configuration file, see Section 2.4.3. Each runtime parameter must be defined with the correct group of parameters in the `snav_params.xml` file.

9.1 Critical Tuning Parameters

Table 9-1 lists parameters to adjust when tuning a new vehicle. If required, these parameter adjustments are critical. See Chapter 8 for parameter descriptions.

Table 9-1 Snapdragon Navigator Critical tuning parameters

Parameter group name	Parameter name(s)
orientation_params	***imu_R00*** through ***imu_R22***
orientation_params	***mag_R00*** through ***mag_R22***
orientation_params	***downward_cam_R00*** through ***downward_cam_R11***
position_control_params	***basethrust***
optic_flow_estimator_params	***camera_offset_x***, ***camera_offset_y***
voltage_monitor_params	***voltage_warn_threshold***
voltage_monitor_params	***voltage_warn_critical***
voltage_monitor_params	***voltage_no_start***
voltage_monitor_params	***voltage_force_landing***
att_control_params	***prop_config_type***, ***prop_config_Dx***, ***prop_config_Dy***, ***prop_config_beta***
att_control_params	***kp_roll_moment***, ***kp_pitch_moment***, ***kpy***, ***kd_roll_moment***, ***kd_pitch_moment***, ***kdy***
att_control_params	***prop_rpm_thrust_curve_a0, 1, 2***
att_control_params	***min_rpm***, ***max_rpm***
rc_params	***min_thrust***, ***max_thrust***

9.2 Tuning Process

When creating a parameters file for a new vehicle type it is recommended to use an existing parameters file of a similar vehicle and modify the required parameters. The example parameters in this section are for the small sample drone, which weighs approximately 240 grams, measures 200 mm from motor-to-motor across the diagonal, and uses 10 cm diameter propellers.

9.2.1 Sensor Orientation

The orientation of the IMU, magnetometer, and downward-facing camera must be specified using the parameters in the **orientation_params** group. This first step requires defining the vehicle body coordinate system. See Section 8.20 for parameter descriptions.

9.2.2 Electronic Speed Controller (ESC) Tuning

The ESC software does not work with a generic motor and propeller. The ESC software requires specific parameters for each specific motor and propeller combination. If using a new propeller or motor, the ESC software must be tuned. The ESC tuning process must currently be performed by Qualcomm and is TBD for future documentation.

9.2.3 Propeller and Motor Properties

The thrust vs. RPM relationship for the propeller must be specified. A quadratic equation is used to specify this relationship. Data to determine this curve can be measured experimentally with a suitable propeller testing rig.

The curve is specified using the following parameters in the **att_control_params** group:

- ***prop_rpm_thrust_curve_a0***
- ***prop_rpm_thrust_curve_a1***
- ***prop_rpm_thrust_curve_a2***

These parameters are used to define the individual propeller curve per the following equation, which provides thrust in grams:

$thrust = a_2 * rpm^2 + a_1 * rpm + a_0$.

Thrust for the small sample drone is as follows:

```
<param name="prop_rpm_thrust_curve_a0" value="-2.1723"/>
<param name="prop_rpm_thrust_curve_a1" value="0.00047332"/>
<param name="prop_rpm_thrust_curve_a2" value="4.0172e-07"/>
```

The maximum and minimum RPM in the **att_control_params** groups must also be specified. Minimum and maximum thrusts are calculated using maximum and minimum RPM values with the propeller thrust curve. A total thrust to weight ratio of at least 2:1 is ideal for best performance.

Total thrust for the small sample drone is as follows:

```
<param name="min_rpm" value="5500"/>
<param name="max_rpm" value="16400"/>
```

9.2.4 Parameter to Specify Total Vehicle Mass

The nominal vehicle takeoff weight in grams must be measured and the ***basethrust*** parameter in the **position_control_params** group must be set based on this value.

For the small sample drone:

```
<param name="basethrust" value="240.0" />
```

9.2.5 Minimum and Maximum Thrust Commands

The minimum and maximum thrust commands must be specified with the ***min_thrust*** and ***max_thrust*** parameters in the **rc_params** group.

Minimum and maximum thrust values for the small sample drone are as follows:

```
<param name="min_thrust" value="60.0"/>
<param name="max_thrust" value="460.0"/>
```

9.2.6 Propeller Configuration

The propeller configuration relative to the chosen vehicle body coordinate system must be specified. See Section 8.13 for a description of the propeller configuration parameters.

The parameters for the small sample drone are as follows:

```
<param name="prop_config_type" value="0"/>
<param name="prop_config_Dx" value="0.071"/>
<param name="prop_config_Dy" value="0.071"/>
<param name="prop_config_beta" value="0"/>
```

For a quadrotor, rectangle and rhombus configurations are supported. A regular hexagon configuration is supported for hexrotors. In Figures 9-1 and 9-2, the coordinate system represents the body frame coordinates. The numbers at the vertices represent the ESC motor IDs.

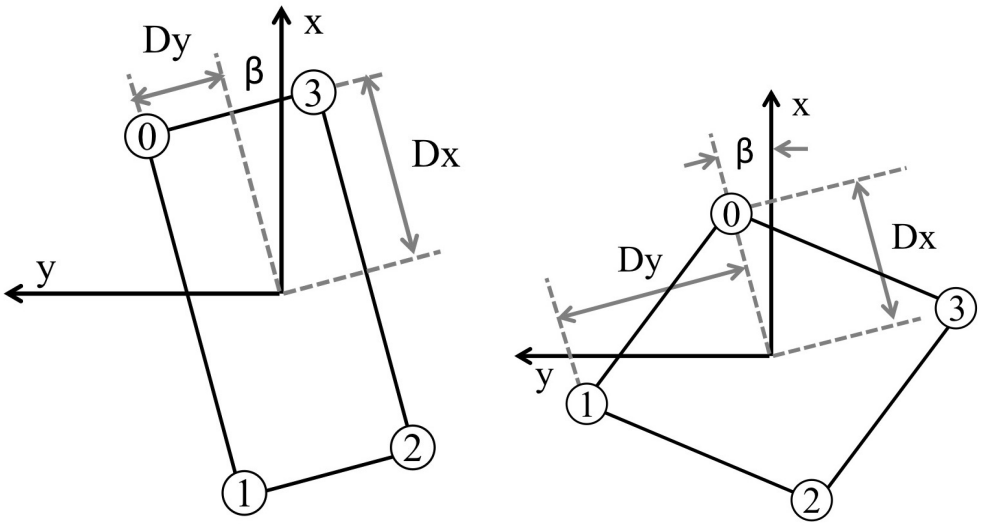


Figure 9-1 Quadrotor rectangle and rhombus propeller configurations

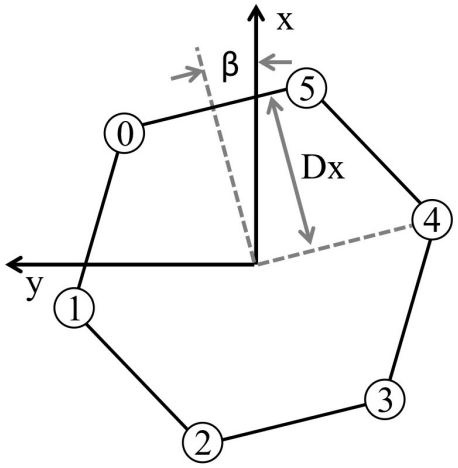


Figure 9-2 Hexrotor regular hexagon propeller configuration

9.2.7 Attitude Control Gains

The attitude controller gains must be tuned using the following parameters in the **att_control_params** group.

Parameters for proportional gains are as follows:

- ***kp_roll_moment***
- ***kp_pitch_moment***
- ***kpy***

Parameters for derivative gains are as follows:

- ***kd_roll_moment***
- ***kd_pitch_moment***

- ***kdy***

See Section 8.13 for more details on these parameters.

Attitude controller outputs are the commanded body frame moments or torques. The proportional gains increase the moment that is created as a function of the attitude or angle errors. The derivative gains increase the moment that is created as a function of the angular velocity errors.

Increasing the proportional terms increases the stiffness of the controller while increasing the derivative terms increases the damping of the system.

Roll and pitch gain adjustments

Calculate initial values for the parameters using the moments of vehicle inertia as follows:

kp_roll_moment = 36.80 * I_{xx}
kp_pitch_moment = 36.80 * I_{yy}
kd_roll_moment = 4.261 * I_{xx}
kd_pitch_moment = 4.261 * I_{yy}

Note: Fine adjustments of roll and pitch gains are likely to be required.

I_{xx} and I_{yy} are the moment of inertia values about the vehicle body x and y axes in grams*meters^2.

Moment of inertia values can be calculated using mechanical CAD software. These gains result in an attitude controller that is slightly overdamped with a settling time of approximately 0.4 seconds to a step input change in roll or pitch angle. The gains must be adjusted from the initial parameter values. Future documentation will provide a similar method for calculating initial values for yaw attitude gains.

Attitude controller adjustments

To create an attitude controller that is responsive and stiff, increase the control gains. Increasing gains too much can cause high frequency vibrations and instability.

To make sure the system does not oscillate when a change in angle is commanded, increase the derivative gains.

Attitude controller gain tuning aids

Experienced tuners use a variety of criteria to aid in tuning, such as the following:

- Flight modes and flight regimes
- Vehicle sound
- Response to step inputs of various magnitudes
- Flight log data
- Visual performance
- Data collected from flight control log files

Small sample drone example

This example uses CAD software to calculate x and y moment of inertia values as $I_{xx} = 0.623$ grams*meters^2 and $I_{yy} = 0.678$ grams*meters^2.

The roll and pitch attitude gains are calculated and specified in the parameters file as follows:

```
<param name="kp_roll_moment" value="22.926"/>
```

```
<param name="kp_pitch_moment" value="24.950"/>
<param name="kd_roll_moment" value="2.655"/>
<param name="kd_pitch_moment" value="2.889"/>
```

The yaw gains for the small sample drone are manually tuned and specified as follows:

```
<param name="kpy" value="64.8"/>
<param name="kdy" value="10.8"/>
```

9.2.8 Voltage Warning Thresholds

Parameters in **voltage_monitor_params** must be adjusted after collecting the voltage discharge curve for the given vehicle and battery during a flight. The parameters must be adjusted so that the voltage warning lights turn on and the vehicle is forced to land with the desired amount of flight time remaining on a given vehicle flight.

The parameters for the small sample drone are as follows:

```
<param name="voltage_warn_threshhold" value="7.05"/>
<param name="voltage_warn_critical" value="6.95"/>
<param name="voltage_no_start" value="6.7"/>
<param name="voltage_force_landing" value="6.9"/>
```

Note: The small sample drone uses a 2-cell LiPo battery. Vehicles using 2-cell LiPo batteries should use similar parameters to those listed above. Vehicles using 3-cell LiPo batteries use parameters approximately 50% larger than the parameters listed for the small sample drone.

9.2.9 Downward Facing Camera Position

The displacement of the downward facing camera from the vehicle center must be specified using the ***camera_offset_x*** and ***camera_offset_y*** parameters in the **optic_flow_estimator_params** group. These parameters are described in Section 8.4.

9.3 Known Limitations

- If the moment of inertia for the X and Y axes are substantially different, the vehicle might not perform well.
- Guidance on all of the parameters that can be tuned is not in the scope of this document.
- A method for tuning the ESC software for a new motor or propeller are TBD.

10 Vision-based Applications

This chapter provides further details on the vision-based applications integrated with Snapdragon Navigator.

10.1 Optic Flow (DFT)

To enter any mode that has a dependency on optic flow, `snav_dft_app` must be running on the applications processor. This application gets frame data from the camera driver, processes optic flow on the frames, and passes the resulting data to Snapdragon Navigator for the flight controller.

Run the following command to configure Snapdragon Navigator to use Optic Flow:

```
$ sudo /etc/snav/configure_dft.sh
```

Snapdragon Navigator restarts with Optic Flow enabled on the downward camera.

To see all available options:

```
$ sudo /etc/snav/configure_dft.sh -h
```

Caution: Optic Flow (DFT) is only supported for downward-facing cameras. Do not try to run this process with a camera tilted at 45 degrees.

Note: Optic Flow (DFT) is only supported for vehicles with a working sonar.

Camera parameter adjustment is enabled by default, so camera exposure and gain are adjusted while running in response to the environment. As a result, the vehicle can fly with optic flow in different lighting scenarios, from dim indoor lighting to bright outdoor sunlight. Performance might be degraded when flying over surfaces with very little texture.

Note: Optic Flow (DFT) is automatically configured upon installation of Snapdragon Navigator for APQ8x74 boards.

10.2 Visual Inertial Odometry (VIO)

To enter VIO mode, `snav_dft_vio_app` must be running on the applications processor. This application gets frame data from the camera driver, processes VIO data on the frames, and passes the resulting data to Snapdragon Navigator for the flight controller.

Run the following command to configure Snapdragon Navigator to use VIO:

```
$ sudo /etc/snav/configure_vio.sh
```

Snapdragon Navigator restarts with VIO enabled on the downward camera tilted at 45 degrees.

To see all available options:

```
$ sudo /etc/snav/configure_vio.sh -h
```

It is possible to use VIO on a strictly downward-facing camera by running the following command:

```
$ sudo /etc/snav/configure_vio.sh -c downward
```

Note: VIO performance improves when using the downward camera tilted at 45 degrees.

Camera parameter adjustment is enabled by default, so camera exposure and gain are adjusted while running in response to the environment. As a result, the vehicle can fly with VIO in different lighting scenarios, from dim indoor lighting to bright outdoor sunlight. Performance might be degraded when flying over surfaces with very little texture.

Caution: Ensure the translation and rotation between the camera and IMU are accurately captured by the parameters in `/etc/snav/mount.snav_dft_vio_app.xml` before attempting to fly in VIO mode. This file is a symlink that is set when the system is configured for VIO.

10.3 Visual Obstacle Avoidance (VOA)

To use VOA, `snav_voa_app` must be running on the applications processor. In addition, this application depends on a valid pose estimate from VIO, so `snav_dft_vio_app` must also be running on the applications processor. The VOA application receives images from the stereo camera and computes a depth estimate. The resulting depth map and the pose from VIO are used to determine safe velocities when navigating in the proximity of obstacles.

Run the following command to configure Snapdragon Navigator to use VOA:

```
$ sudo /etc/snav/configure_voa.sh
```

Snapdragon Navigator restarts with VOA processing stereo images and VIO enabled on the downward camera tilted at 45 degrees.

To see all available options:

```
$ sudo /etc/snav/configure_voa.sh -h
```

It is possible to use VOA with VIO on a strictly downward-facing camera by running the following command:

```
$ sudo /etc/snav/configure_voa.sh -c downward
```

Note: VIO performance improves when using the downward camera tilted at 45 degrees.

Caution: VOA requires an accurate stereo calibration. Do not try to run this application without first performing a sufficient stereo camera calibration. Once calibrated, edit `/etc/snav/app_params.snav_voa_app.xml` and/or `/etc/snav/calibration.stereo.xml` with the calibration parameters.

10.4 Parameters for Vision-related Applications

Snapdragon Navigator has multiple vision-related applications, including `snav_dft_app`, `snav_dft_vio_app`, and `snav_voa_app`. These applications have corresponding parameter files for setting parameters associated with each application, located in `/etc/snav/app_params.<app_name>.xml`.

10.4.1 General Application Parameters

- ***enable_dft*** – Enable (1) or disable (0) the DFT process.
- ***enable_vio*** – Enable (1) or disable (0) the VIO process.
- ***enable_srw_writer*** – Enable (1) or disable (0) logging image sequences. See Section 3.1.2.

10.4.2 Parameters Specific to DFT

- ***mv_dft_max_nr*** – Maximum number of features forced as input to optical flow. More features provides more stability, but runs slower.
- ***mv_dft_min_nr*** – Minimum number of features forced as input to optical flow. Fewer features results in less stability in texture-poor areas.

10.4.3 Parameters Specific to VIO

These parameters will be documented at a later time.

10.4.4 Parameters Specific to VOA

These parameters will be documented at a later time.

10.4.5 Camera Configuration Parameters

The following parameters are found in the corresponding camera file, `/etc/snav/camera.<camera_name>.xml`.

- ***pixel_width*** – Camera resolution width used for configuration.
- ***pixel_height*** – Camera resolution height used for configuration.
- ***memory_stride*** – Camera image stride used for configuration.
- ***fps*** – Number of frames per second.
- ***default_exposure*** – Normalized to a range of $[0.0, 1.0]$.
- ***default_gain*** – Normalized to a range of $[0.0, 1.0]$.

The following parameters are related to automatic camera parameter adjustment (CPA).

- ***enable_cpa*** – Enable (1) or disable (0) the CPA process.
- ***mv_cpa_type*** – Legacy (0) or cost-based (1).
- ***mv_cpa_filter_size*** – Internal filter size for exposure and gain changes (the larger the filter size, the slower the convergence (0 = no filtering)).

- ***mv_cpa_exposure_cost*** – Cost to increase exposure.
- ***mv_cpa_gain_cost*** – Cost to increase gain.

The sum of gain cost and exposure cost influences how much brightness cost is weighted. The MV SDK defines guidelines for the cost-based approach as follows:

- If gain cost plus exposure cost is greater than 1.0, minimizing gain and exposure values is weighted higher than the brightness goal.
- If the sum is less than 1.0, the brightness goal is weighted higher.

The ratio of ***mv_cpa_gain_cost*** and ***mv_cpa_exposure_cost*** is the ratio between gain and exposure values.

10.4.6 Camera Calibration Parameters

These parameters are found in the corresponding calibration file, `/etc/snav/calibration.<camera_name>.xml`.

- ***calib_pixel_width*** – Camera resolution width used for calibration.
- ***calib_pixel_height*** – Camera resolution height used for calibration.
- ***calib_memory_stride*** – Camera image stride used for calibration.
- ***principal_point_x*** – Camera principal point offset on the X axis.
- ***principal_point_y*** – Camera principal point offset on the Y axis.
- ***focal_length_x*** – Camera focal length on the X axis in pixels.
- ***focal_length_y*** – Camera focal length on the Y axis in pixels.
- ***distortion_model*** – None (0), polynomial model (4, 5, 8), or fisheye model (10).
- ***distortion_<0-7>*** – Distortion coefficients.

11 Additional Features

Snapdragon Navigator has features to increase safety, help the vehicle communicate warnings to the user, and perform special tasks (such as binding an RC).

11.1 Propeller Obstruction Detection

Snapdragon Navigator uses bi-directional ESC communication to send commands to the ESCs and receive feedback about the state of the motors. This feedback is used to detect obstructions to the propellers.

11.1.1 During Propeller Spin-up

If one or more of the motors is detected to have stalled during the initial propeller spin-up phase and does not begin spinning, all the motors stop. Control returns to the user to attempt another takeoff.

11.1.2 During Flight

If one or more of the motors is detected to have stalled during flight, the vehicle enters Emergency Kill mode, in which all motors immediately stop, the status LED shows solid white, and the vehicle issues loud beeps. This safety feature is intended to prevent the propellers from harming someone or something in the event of a crash.

Note: Once in Emergency Kill mode, the vehicle cannot change modes until the pilot completes the sequence to start the propellers. Once the sequence is complete, Snapdragon Navigator exits Emergency Kill mode. See Section 4.3 for the start up sequence.

11.2 Low Voltage Warnings and Forced Landing

When the battery voltage reaches a certain threshold, the status LED overlays the Low Voltage warning code over the presently displayed code. The pilot must immediately begin planning for landing when the Low Voltage warning is displayed.

If the battery voltage continues to decrease until reaching a second threshold:

- Status LED displays the Critical Voltage warning code
- Depending on the flight mode, one of the following actions occur:
  - Vehicle starts forcing the pilot to land by limiting the maximum possible thrust
  - Vehicle lowers the height setpoint

Note: The pilot must avoid flying the vehicle to this stage.

11.3 Geotether in GPS Modes

The geotether feature enables Snapdragon Navigator to use GPS data to limit the allowable flight region of the vehicle. In GPS Position Control Mode, the maximum radial translation from the takeoff location is limited. The maximum radius is defined by the **geotether_radius** parameter described in Section 8.25. The user cannot command the desired X and Y position outside of this circular region.

The maximum altitude is limited in any mode in which the user controls the Z velocity. The user is not allowed to command the desired Z position above a certain height. The height above the takeoff location is limited by the **max_relative_pressure_height** parameter described in Section 8.8. This feature has no impact in modes in which the user controls the thrust magnitude.

11.4 No Fly Zone Feature

Snapdragon Navigator can use GPS data to restrict flight outside of a certain safe region. The parameters that define this feature are described in Section 8.17. A user can specify a safe circle zone using a radius and a latitude and longitude coordinate.

The following occur if the feature is enabled and GPS data indicates that the vehicle is outside of the safe zone:

- LED warning light flashes, see Section 12.1
- Vehicle propellers do not start up
- If the vehicle is in a flight mode in which the user controls the Z velocity, the vehicle is forced to descend

In a future release, the no-fly zone feature will be extended to add exclusion zones (e.g., airports) with similar restrictions on flight.

11.5 Sensor Checks That Do Not Allow Propeller Spinup

Snapdragon Navigator checks certain sensor values before the propellers are allowed to spin up. The angular velocity components must be below a certain threshold, the acceleration magnitude must be within a certain range, and the total roll and pitch tilt angle must be below a threshold. These parameters are defined in Chapter 8. If the vehicle is commanded to start up and is not allowed to by this feature, the vehicle makes a "sad" tone.

11.6 Simulation Mode

11.6.1 Introduction

Simulation mode is a feature that enables hardware-in-the-loop flight simulation and pre-flight testing of Snapdragon Navigator API programs. In this mode, Snapdragon Navigator takes sensor inputs from a simulation engine that models physics and sensors (the vehicle modeled is the small sample drone). User commands can be input using a Spektrum RC, DroneController, or API programs. In simulation, all sensor inputs are noiseless and are generated with zero latency at their expected rates.

The position and pose of the vehicle can be monitored using the logged `sim_ground_truth` field. This

field can be queried by an API program for the purposes of 3D visualization of the simulation. The world is modeled as obstruction free with a flat ground plane at `z=0`. The roll and pitch are zeroed when the vehicle hits the ground plane. Simulation always starts with position and roll/pitch/yaw set to `(0,0,0)` (yaw is defined with respect to due East).

11.6.2 Usage

Run the `sim_time` command to launch a simulation in which `sim_time` is the minimum number of seconds to simulate. If the props are on at `t=sim_time`, the simulation continues to run until the props are no longer spinning for at least 1 second.

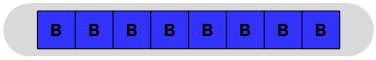
```
$ snav -w sim_time
```

12 Status LED Reference

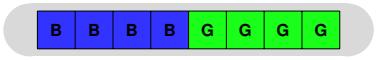
12.1 LED Color Codes

Snapdragon Navigator uses LED color codes to communicate vehicle status. Each code is defined as a sequence of eight repeating colors.

12.1.1 Flight Mode Color Codes



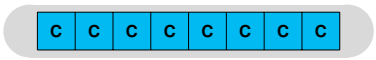
Thrust Angle mode



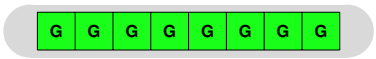
Thrust Angle mode (GPS Hover Position Hold)

GPS Position Control mode

Height Control mode (using barometer)

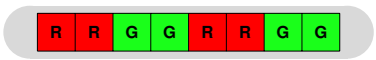


Height Control mode (using sonar)

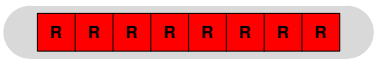


Optic Flow mode (using sonar)

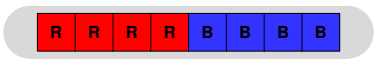
Optic Flow mode (not using sonar)



Height Control mode with lower roll/pitch angle limits



Rate mode



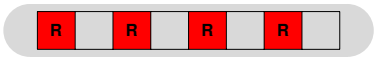
Thrust Attitude Angular Velocity mode (API only)

VIO mode

Position Hold mode

Position Hold mode with GPS lock

12.1.2 Landing Mode Color Codes

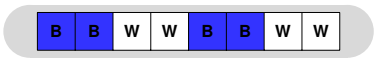


Go Home mode and Landing with GPS mode

Pressure Landing mode

Emergency Landing mode

12.1.3 Calibration Color Codes



Gyro/Accel Temperature Calibration mode

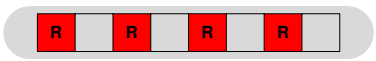
Accelerometer Static Offset Calibration mode

Accelerometer Dynamic Offset Calibration in progress (overlay)

Compass Calibration Prep mode

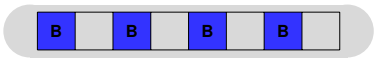
Compass Calibration mode

Calibration success

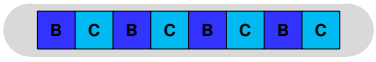


Calibration failure

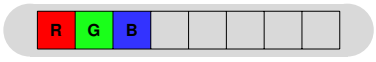
12.1.4 Miscellaneous Color Codes



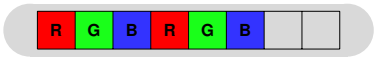
Waiting for RC or DroneController to connect



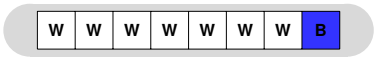
Binding mode



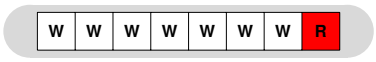
Low Voltage warning (overlay)



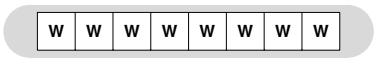
Critical Voltage warning (overlay)



ESC RPM mode



ESC PWM mode



Emergency Kill mode

IMU error

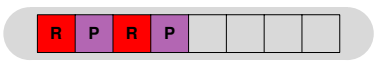
Motors error

Attitude Estimator error

Barometer error

Accelerometer Static Offset calibration is required

Gyro/Accel Temperature calibration is required



Vehicle is in a no-fly zone (overlay)

Magnetometer (compass) warning (overlay)

Generic error

12.2 Boot Sequence Example

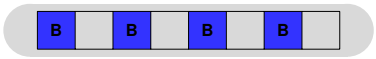
Turn on the vehicle power. The ESCs emit a beeping noise.

State Machine has not been initialized

IMU begins initializing.

IMU error

Vehicle is stationary and IMU initialization finishes successfully. ESCs emit startup chime, which is a sequence of tones from low to high frequency.



Waiting for RC

Turn on RC. ESCs emit a tone indicating that RC data is being received. LED status code now depends on channel values of RC. Vehicle is ready to fly.

13 Troubleshooting

13.1 Snapdragon Navigator Is Not Initializing

The status LED can help determine why the vehicle is not booting up. See Chapter 12 for a comprehensive list of status LED codes.

Has the correct platform build been installed on the applications processor?

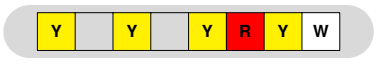
Because Snapdragon Navigator depends on some of the included libraries, it is important to install the correct platform image on the applications processor. If a wrong library version is installed, a runtime error might occur. This type of error can be difficult to debug using Snapdragon Navigator tools.

Is the runtime parameter file present?

Make sure that `snav_params.xml` is present at `/usr/share/data/adsp` , otherwise Snapdragon Navigator does not initialize. See Section 2.4.3 for instructions.

Have you run the static accelerometer offset calibration?

Run Accelerometer Offset Estimation calibration if the following LED code displays:



Accelerometer static offset calibration is required

See Section 6.2 for instructions.

Power cycle the vehicle after running the calibration.

Is the vehicle resting on a stationary surface after you turn on the power?

The following LED code indicates that the IMU is attempting to initialize but is unable to complete due to detected movement.

Imu error

Set the vehicle on a different surface.

Communication issue

The following LED code indicates a communication issue between the ESCs and the flight controller.

Motors error

Make sure the ESCs are connected to the proper port and that no connectors are damaged.

13.2 I Cannot Start the Propellers

Confirm that the status LED is not displaying an error code. See Chapter 12 for a comprehensive list of error codes. The propellers do not start in the desired mode if the status LED displays an error code.

The mode can be changeable depending on what the error code is; otherwise, the error requires troubleshooting (that is, if there is a motor error, IMU error, or attitude estimate error).

When the status LED displays a code corresponding to a flight mode, attempt the startup sequence. Ensure that none of the propellers is obstructed, which results in automatic stoppage of all propellers.

If a tone does not sound when the throttle stick is lowered and yawed left or right, adjust the throttle trim on your RC. Adjust the trim until the tone is emitted.

The sensor values could be out of the range in which starting the propellers is allowed. See Section 11.5.

The vehicle could be in a no-fly zone, see Section 11.4.

13.3 Vehicle Makes a "Sad" Tone After Powering On

If a sequence of tones from high frequency to low frequency is emitted after powering on (before the initialization chime occurs, if at all), the vehicle is warning that some recommended calibrations have not been loaded successfully. Flight might be permitted, but performance might be degraded. See Section 6.5 for recommended calibration procedures, and perform them to suppress the warning chime.

13.4 Snapdragon Navigator Cannot Enter Optic Flow Mode

Is the optic flow process running on the applications processor? If not, the LED code displays as follows:

Generic error

13.5 Vehicle Does Not Fly With the DroneController Application

Confirm the following:

- The device running DroneController is connected to the drone's Wi-Fi network
- The port is set to 14556 in the application settings
- The IP address matches the drone's IP address

If there is no position estimate available (see Section 4.1.8), the LED code displays as follows:

Generic error

13.6 Optic Flow Mode Not Working Properly

In the following conditions, Optic Flow mode might transition to Height Control mode with reduced roll/pitch angle limits:

- The vehicle is over surfaces with little texture

- The vehicle is in very dark environments

Caution: The vehicle is unable to maintain its position in Height Control mode.

A Terms and Conditions

THIS TERMS AND CONDITIONS OF USE (THE “AGREEMENT”) IS A LEGALLY BINDING AGREEMENT BETWEEN QUALCOMM TECHNOLOGIES, INC. ("QTI") AND YOU OR THE LEGAL ENTITY YOU REPRESENT (“You” or “Your”). QTI IS WILLING TO PROVIDE THESE INSTRUCTION SETS AND ANY ASSOCIATED DOCUMENTATION (COLLECTIVELY REFERRED TO AS THE “INSTRUCTIONS”) TO YOU ONLY ON THE CONDITION THAT YOU ACCEPT AND AGREE TO ALL OF THE TERMS AND CONDITIONS IN THIS AGREEMENT. BY DOWNLOADING AND USING THE INSTRUCTIONS YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. IF YOU DO NOT AGREE TO THESE TERMS, QTI IS UNWILLING TO AND DOES NOT LICENSE THE INSTRUCTIONS TO YOU. IF YOU DO NOT AGREE TO THESE TERMS YOU MUST DISCONTINUE THE USE OF THE INSTRUCTIONS AND YOU SHALL NOT USE THE INSTRUCTIONS OR RETAIN ANY COPIES OF THE INSTRUCTIONS. ANY USE OR POSSESSION OF THE INSTRUCTIONS BY YOU IS SUBJECT TO THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT.

BY USING THE INSTRUCTIONS, YOU REPRESENT, WARRANT AND CERTIFY THAT: YOU ARE AN AUTHORIZED REPRESENTATIVE OF THE LEGAL ENTITY YOU REPRESENT; YOU HAVE READ THIS AGREEMENT AND UNDERSTAND IT, INCLUDING THE CIVIL CODE SECTION BELOW; YOU HAVE THE AUTHORITY TO BIND THE LEGAL ENTITY YOU REPRESENT TO THE TERMS AND CONDITIONS OF THIS AGREEMENT; AND YOU AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS.

1. **DESIGN OF YOUR PRODUCTS.** You acknowledge that QTI has had and will have no participation in or control over – and no responsibility for – the design or assembly of Your products, including drones, the integration of any of the attached Materials into Your products, including drones, or the sale or marketing of Your products, including drones.
2. **ASSUMPTION OF RISK.** You acknowledge that the operation of the Materials, alone or in a product, including drones, is a potentially dangerous activity and may result in significant harm to property or injury or death to persons. You agree to include on Your products prominent warnings of such risks, as may be required by law or regulation and as may be necessary or prudent to advise users of such risks. You, and not QTI, assume all risks and liabilities that may result from the use of the Materials, whether or not modified by You and whether or not implemented in connection with a reference design provided by QTI or any of its Affiliates.
3. **SAFETY PRECAUTIONS AND PROCEDURES.** You will operate Your products, including drones, in compliance with any and all safety procedures and precautions as are reasonable for the operation of Your products, including drones. This may include using blade guards on drones or operating any drone sufficiently far away from people, property or other hazardous structures e.g., electricity lines. In addition, You will operate Your products, including drones, in compliance with all applicable laws and regulations, including safety and operational guidelines, that apply to the use of Your products, including drones.
4. **WARRANTY.** THE MATERIALS ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAWS, QTI AND ITS AFFILAITES EXPRESSLY DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, TITLE, FITNESS FOR A PARTICULAR PURPOSE AND WARRANTIES THAT THE MATERIALS ARE FREE FROM THE RIGHTFUL CLAIM OF ANY THIRD PARTY, BY WAY OF INFRINGEMENT OR THE LIKE. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY QTI OR ITS AUTHORIZED REPRESENTATIVES SHALL CREATE OR EXTEND ANY WARRANTY.

5. **LIMITATION OF LIABILITY.** IN NO EVENT SHALL QTI OR ANY OF ITS AFFILIATES BE LIABLE TO YOU FOR ANY INCIDENTAL, CONSEQUENTIAL OR SPECIAL DAMAGES, INCLUDING, BUT NOT LIMITED TO, ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL DAMAGES ARISING OUT OF OR IN CONNECTION WITH THIS AGREEMENT OR THE USE OR INABILITY TO USE, OR THE DELIVERY OF OR FAILURE TO DELIVER THE MATERIALS EVEN IF QTI OR ITS AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. QTI'S TOTAL, CUMULATIVE LIABILITY FOR DIRECT DAMAGES ARISING FROM OR IN CONNECTION WITH THIS AGREEMENT, WILL BE LIMITED TO A TOTAL AMOUNT OF ONE HUNDRED UNITED STATES DOLLARS (US $100). MULTIPLE CLAIMS WILL BE AGGREGATED TO DETERMINE THE SATISFACTION OF THIS LIMIT.
6. **INDEMNITY.** You agree to defend, indemnify and hold QTI, its Affiliates, employees, directors, agents, licensors, successors and assignees (each an "Indemnified Party") harmless from any and all claims, penalties, demands, causes of action, liabilities, lawsuits, or damages, including attorneys' fees and costs, that result from or relate to the Materials or any product, including drones, made, used, sold, imported, exported, or distributed by You which uses the Materials or any part or derivative work thereof, even where such product uses the Materials without modification and even where the design of such product is identical to the design of any reference product, including drones. This indemnification includes, without limitation, any claims for damages to property or injury or death to persons and any investigation, enforcement action, civil penalty, or other action conducted or cost imposed by the United States Federal Aviation Administration (FAA) or any governmental entity of the United States or any other government. If any third party asserts a claim or initiates an action against an Indemnified Party for which You are responsible under this Section, QTI shall promptly notify You when it becomes aware of such claim or action, provided, however, that any delay in notification shall not relieve You from Your indemnification obligations under this Agreement. QTI shall have the right to participate in the defense of such claim or action, including any related settlement negotiations. No such claim or action may be settled or compromised without QTI's express written consent, which may be conditioned upon the execution of a release of all claims against the Indemnified Parties by the party bringing such claim or action.
7. **GENERAL RELEASE.**
   (a) Release. In consideration of QTI's allowing You to use the Materials, You on behalf of Yourself, Your, affiliates, agents, successors and assigns, hereby fully and forever release and discharge QTI (and all of its officers, directors, employees, agents, successors, assigns, control persons, subsidiaries and Affiliated companies, together the "Released Parties"), from any and all claims, demands, liabilities, obligations, responsibilities, suits, actions and causes of action against any of the Released Parties, whether liquidated or unliquidated, fixed or contingent, known or unknown, presently existing or which, through the passage of time, might arise in the future, related to or arising out of Your use of the Materials.
   (b) Waiver and Acknowledgement. You hereby expressly waives all rights under Section 1542 of the Civil Code of the State of California, and under any and all similar laws of any governmental entity. You hereby confirm that you are aware that said Section 1542 of the Civil Code provides as follows: "A general release does not extend to claims which the creditor does not know or suspect to exist in his favor at the time of executing the release, which if known by him must have materially affected his settlement with the debtor."
8. **INSURANCE.** If You make or distribute drones, or participate in making or distributing drones, You agree to and will maintain (throughout the time You are using any of the Materials in connection with such drones) insurance providing adequate coverage for potential product liability, personal injury, property damage, and privacy claims and litigation associated with such drones and/or Materials.

B References

B.1 Related Documents

Title	Number
Qualcomm Technologies	
Qualcomm Snapdragon Navigator Developer Guide	80-P4698-20
Qualcomm Snapdragon Flight™ Reference Platform User Guide	80-H9631-1

B.2 Acronyms and Terms

Acronym or term	Definition
AGL	Above ground level
aDSP	Applications digital signal processor
CPA	Camera parameter adjustment
DFT	Downward-facing tracker
ESC	Electronic speed controller
IMU	Inertial-measurement unit
MV	Machine vision
UART	Universal asynchronous receiver/transmitter
VIO	Visual inertial odometry
VOA	Visual obstacle avoidance