Table Of Contents

Cleanflight

Welcome to CleanFlight!

Cleanflight is an community project which attempts to deliver flight controller firmware and related tools.

Primary Goals

- Community driven.
- Friendly project atmosphere.
- Focus on the needs of users.
- Great flight performance.
- Understandable and maintainable code.

Hardware

See the flight controller hardware chapter for details.

Software

There are two primary components, the firmware and the configuration tool. The firmware is the code that runs on the flight controller board. The GUI configuration tool (configurator) is used to configure the flight controller, it runs on Windows, OSX and Linux.

Feedback & Contributing

We welcome all feedback. If you love it we want to hear from you, if you have problems please tell us how we could improve things so we can make it better for everyone.

If you want to contribute please see the notes here:

https://github.com/cleanflight/cleanflight#contributing

Developers should read this:

https://github.com/cleanflight/cleanflight/blob/master/CONTRIBUTING.md

Getting Started

This is a step-by-step guide that can help a person that has never used Cleanflight before set up a flight controller and the aircraft around it for flight. Basic RC knowledge is required, though. A total beginner should first familiarize themselves with concepts and techniques of RC before using this (e.g. basic controls, soldering, transmitter operation etc). One could use RCGroups and/or the Youtube show FliteTest for this.

DISCLAIMER: This documents is a work in progress. We cannot guarantee the safety or success of your project. At this point the document is only meant to be a helping guide, not an authoritative checklist of everything you should do to be safe and successful. Always exercise common sense, critical thinking and caution.

Read the Introduction chapter for an overview of Cleanflight and how the community works.

Hardware

NOTE: Flight Controllers are typically equipped with accelerometers. These devices are sensitive to shocks. When the device is not yet installed to an aircraft, it has very little mass by itself. If you drop or bump the controller, a big force will be applied on its accelerometers, which could potentially damage them. Bottom line: Handle the board very carefully until it's installed on an aircraft!

For an overview of the hardware Cleanflight (hereby CF) can run on, see Boards.md. For information about specific boards, see the board specific documentation.

- Assuming that you have a flight controller board (hereby FC) in hand, you should first read through the manual that it came with. You can skip the details about software setup, as we'll cover that here.

- Decide how you'll connect your receiver by reading the receiver chapter, and how many pins you need on the outputs (to connect ESCs and servos) by reading about Mixers.

- If you're interested in monitoring your flight battery with CF, see Battery Monitoring.

- You may want audible feedback from your copter so skim through Buzzer and mark the pins that will be used.

- Do you want your RC Receiver's RSSI to be sent to the board? The RSSI chapter explains how. You may or may not need to make an additional connection from your Receiver to the FC.

- Would you like to try using a GPS unit to get your aircraft to Loiter or Return-To-Launch? Take a look at the GPS and GPS Tested Hardware chapters.

- You may also want to read the Serial chapter to determine what extra devices (such as Blackbox, OSD, Telemetry) you may want to use, and how they should be connected.

- Now that you know what features you are going to use, and which pins you need, you can go ahead and solder them to your board, if they are not soldered already. Soldering only the pins required for the application may save weight and contribute to a neater looking setup, but if you need to use a new feature later you may have to unmount the board from the craft and solder missing pins, so plan accordingly. Before soldering your FC please review a how-to-solder tutorial to avoid expensive mistakes, practice soldering on some scrap before soldering your FC.

- If you are going to use Oneshot125, you may need to enable that on your ESCs using a jumper or flashing them with the latest stable firmware and enable Damped Light in their settings, if it's supported. Refer to the ESCs' documentation or online discussions to determine this.

Software setup

Now that your board has pins on it, you are ready to connect it to your PC and flash it with CF. Install the Chromium browser or Google Chrome to your PC, if you don't have it already, add the Cleanflight Configurator to it, and start it.

Then follow these instructions for Installation of the firmware to the FC.

Cleanflight Configuration

Your FC should now be running CF, and you should be able to connect to it using the Configurator. If that is not the case, please go back to the previous sections and follow the steps carefully.

Now, there are two ways to configure CF; via the Configurator's tabs (in a "graphical" way, clicking through and selecting/changing values and tickboxes) and using the Command Line Interface (CLI). Some settings may only be configurable using the CLI and some settings are best configured using the GUI (particularly the ports settings, which aren't documented for the CLI as they're not human friendly).

- It is now a good time to setup your RC Receiver and Transmitter. Set the Tx so that it outputs at least 4 channels (Aileron, Elevator, Throttle, Rudder) but preferably more. E.g. you can set channels 5 and 6 to be controlled by 3-position switches, to be used later. Maybe set up EXPO on AIL/ELE/RUD, but you should know that it can also be done in CF's software later. If using RSSI over PPM or PWM, it's now time to configure your Rx to output it on a spare channel.

- Connect the Rx to the FC, and the FC to the PC. You may need to power the Rx through a BEC (its 5V rail - observe polarity!).

- On your PC, connect to the Configurator, and go to the first tab. Check that the board animation is moving properly when you move the actual board. Do an accelerometer calibration.

- Configuration tab: Select your aircraft configuration (e.g. Quad X), and go through each option in the tab to check if relevant for you.

    - E.g. you may want to enable ONESHOT125 for Oneshot-capable ESCs.
    - You may need RX_PPM if you're using an RC Receiver with PPM output etc.
    - If planning to use the battery measurement feature of the FC, check VBAT under Battery Voltage.
    - If using analog RSSI, enable that under RSSI. Do not enable this setting if using RSSI injected into the PPM stream.
    - Motors will spin by default when the FC is armed. If you don't like this, enable MOTOR_STOP.
    - Also, adjust the minimum, middle and maximum throttle according to these guidelines:
    - Minimum Throttle - Set this to the minimum throttle level that enables all motors to start reliably. If this is too low, some motors may not start properly after spindowns, which can cause loss of stability and control. A typical value would be 1100.
    - Middle Throttle - The throttle level for middle stick position. Many radios use 1500, but some (e.g. Futaba) may use 1520 or other values.
    - Maximum Throttle - The maximum throttle level that the ESCs should receive. A typical value would be 2000.
    - Minimum Command - This is the "idle" signal level that will be sent to the ESCs when the craft is disarmed, which should not cause the motors to spin. A typical value would be 1000.
    - Finally, click Save and Reboot.

- Receiver tab:

    - Check that the channel inputs move according to your Tx inputs.
    - Check that the Channel map is correct along with the RSSI Channel, if you use that.
    - Verify the range of each channel goes from ~1000 to ~2000. See also controls. and `rx_min_usec` and `rx_max_usec`.
    - You can also set EXPO here instead of your Tx.
    - Click Save!

- Modes tab: Setup the desired modes. See the modes chapter for what each mode does, but for the beginning you mainly need HORIZON, if any.

- Before finishing this section, you should calibrate the ESCs, install the FC to the frame, and connect the RSSI cable, buzzer and battery if you have chosen to use those.

Final testing and safety

It's important that you have configured CF properly, so that your aircraft does not fly away, or even worse fly into property and people! This is an important step that you should NOT postpone until after your maiden flight. Please do this now, before you head off to the flying field.

- First, learn how to arm your FC, and about other controls.
- Next up, setup Failsafe. Take your time, do it properly.
- Now, on the bench, without props, test that failsafe works properly, according to the above doc.
- Additionally, test the effect of AIL/ELE input of your Tx. Is the aircraft responding properly? Do the same for RUD input.
- Test the direction of AIL/ELE auto correction. Raise throttle at 30% (no blades!); when you tilt the aircraft, do the motors try to compensate momentarily? This should simulate random wind forces that the FC should counteract
- Test the direction of AIL/ELE auto correction in HORIZON mode. With throttle at 30%, if you tilt the aircraft so that one motor is lowered towards the ground, does it spin up and stay at high RPM until you level it off again? This tests the auto-leveling direction.

If one of these tests fail, do not attempt to fly, but go back to the configuration phase instead. Some channel may need reversing, or the direction of the board is wrong.

Using it (AKA: Flying)

Go to the field, turn Tx on, place aircraft on the ground, connect flight battery and wait. Arm and fly. Good luck!

Advanced Matters

Some advanced configurations and features are documented in the following pages, but have not been touched-upon earlier:

- Profiles
- PID tuning
- In-flight Adjustments
- Blackbox logging
- Using a Sonar
- Spektrum Bind
- Telemetry
- Using a Display
- Using a LED strip
- Migrating from baseflight # Safety

As many can attest, multirotors and RC models in general can be very dangerous, particularly on the test bench. Here are some simple golden rules to save you a trip to the local ER: * **NEVER** arm your model with propellers fitted unless you intend to fly! * **Always** remove your propellers if you are setting up for the first time, flashing firmware, or if in any doubt.

Before Installing

Please consult the Cli, Controls, Failsafe and Modes pages for further important information.

You are highly advised to use the Receiver tab in the CleanFlight Configurator, making sure your Rx channel values are centered at 1500 (1520 for Futaba RC) with minimum & maximums of 1000 and 2000 (respectively) are reached when controls are operated. Failure to configure these ranges properly can create problems, such as inability to arm (because you can't reach the endpoints) or immediate activation of failsafe.

You may have to adjust your channel endpoints and trims/sub-trims on your RC transmitter to achieve the expected range of 1000 to 2000.

The referenced values for each channel have marked impact on the operation of the flight controller and the different flight modes.

Props Spinning When Armed

With the default configuration, when the controller is armed, the propellers *WILL* begin spinning at low speed. We recommend keeping this setting as it provides a good visual indication the craft is armed.

If you wish to change this behavior, see the MOTOR_STOP feature in the Configurator and relevant docuemntation pages. Enabling this feature will stop the props from spinning when armed.

Installation

Using the configurator

This is a generic procedure to flash a board using the configurator. The configurator does not yet support all boards, so please check the documentation corresponding to your board before proceeding.

Make sure you have the Cleanflight Configurator installed, then:

- Connect the flight controller to the PC.
- Start the Cleanflight Configurator.
- Click on "Disconnect" if the configurator connected to the board automatically.
- Click on the "Firmware Flasher" tab.
- Make sure you have internet connectivity and click on the "Load Firmware [Online]" button.
- Click on the "Choose a Firmware / Board" dropdown menu, and select the latest stable version for your flight controller.
- IMPORTANT: Read and understand the release notes that are displayed. When upgrading review all release notes since your current firmware.
- If this is the first time Cleanflight is flashed to the board, tick the "Full Chip Erase" checkbox.
- Connect the flight controller board to the PC. Ensure the correct serial port is selected.
- Click on the "Flash Firmware" button and hold still (do not breathe, too).
- When the progress bar becomes green and reads "Programming: SUCCESSFUL" you are done!

Manually

See the board specific flashing instructions.

Upgrading

When upgrading be sure to backup / dump your existing settings. Some firmware releases are not backwards compatible and default settings are restored when the FC detects an out of date configuration.

Backup/Restore process

See the CLI section of the docs for details on how to backup and restore your configuration via the CLI.

Configuration

Cleanflight is configured primarily using the Cleanflight Configurator GUI.

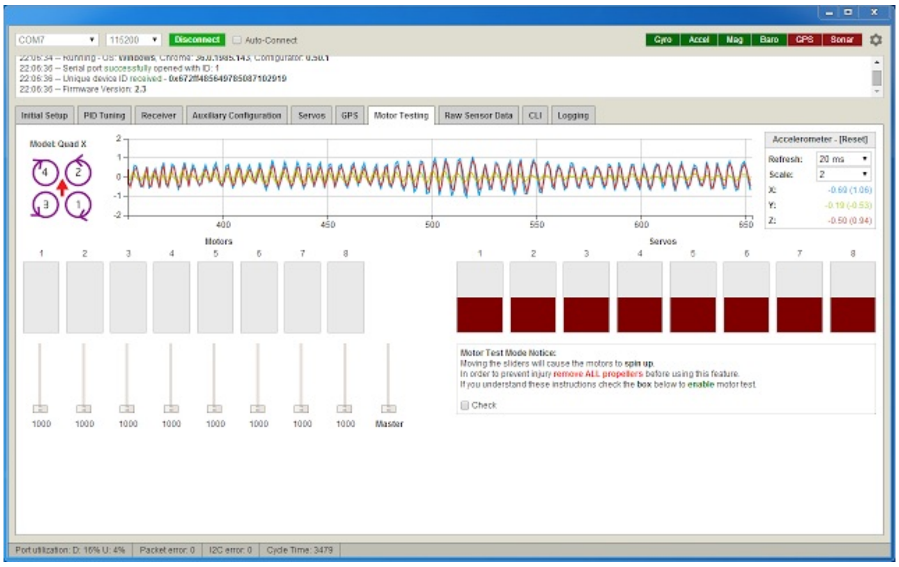
Both the command line interface and GUI are accessible by connecting to a serial port on the target, be it a USB virtual serial port, physical hardware UART port or a SoftSerial port.

See the Serial section for more information and see the Board specific sections for details of the serial ports available on the board you are using.

The GUI cannot currently configure all aspects of the system, the CLI must be used to enable or configure some features and settings.

Due to ongoing development, the fact that the GUI cannot yet backup all your settings and automatic chrome updates of the GUI app it is highly advisable to backup your settings (using the CLI) so that when a new version of the configurator or firmware is released you can re-apply your settings.

GUI



```
22:06:34 -- Running - OS: Windows, Chrome: 36.0.1985.143, Configurator: 0.50.1
22:06:36 -- Serial port successfully opened with ID: 1
22:06:36 -- Unique device ID received - 0x672ff485649785087102919
22:06:36 -- Firmware Version: 2.3
```

Accelerometer - [Reset]
Refresh: 20 ms
Scale: 2
X: -0.69 (1.06)
Y: -0.19 (-0.53)
Z: -0.50 (0.94)

Motor Test Mode Notice:
Moving the sliders will cause the motors to **spin up**.
In order to prevent injury **remove ALL propellers** before using this feature.
If you understand these instructions check the **box** below to **enable** motor test.

Check

1000 1000 1000 1000 1000 1000 1000 1000

Port utilization: D: 16% U: 4% | Packet error: 0 | I2C error: 0 | Cycle Time: 3479

The GUI tool is the preferred way of configuration. The GUI tool also includes a terminal which can be used to interact with the CLI.

[Cleanflight Configurator on Chrome store]

If you cannot use the latest version of the GUI to access the FC due to firmware compatibility issues you can still access the FC via the CLI to backup your settings, or you can install an old version of the configurator.

Old versions of the configurator can be downloaded from the configurator releases page: https://github.com/cleanflight/cleanflight-configurator/releases See the README file that comes with the configurator for installation instructions.

CLI

Cleanflight can also be configured by a command line interface.

See the CLI section of the documentation for more details.

Command Line Interface (CLI)

Cleanflight has a command line interface (CLI) that can be used to change settings and configure the FC.

Accessing the CLI.

The CLI can be accessed via the GUI tool or via a terminal emulator connected to the CLI serial port.

1. Connect your terminal emulator to the CLI serial port (which, by default, is the same as the MSP serial port)
2. Use the baudrate specified by msp_baudrate (115200 by default).
3. Send a `#` character.

To save your settings type in 'save', saving will reboot the flight controller.

To exit the CLI without saving power off the flight controller or type in 'exit'.

To see a list of other commands type in 'help' and press return.

To dump your configuration (including the current profile), use the 'dump' command.

See the other documentation sections for details of the cli commands and settings that are available.

Backup via CLI

Disconnect main power, connect to cli via USB/FTDI.

dump using cli

```
rateprofile 0
profile 0
dump
```

dump profiles using cli if you use them

```
profile 1
dump profile
profile 2
dump profile
```

dump rate profiles using cli if you use them

```
rateprofile 1
dump rates
rateprofile 2
dump rates
```

copy screen output to a file and save it.

Restore via CLI.

Use the cli `defaults` command first.

When restoring from a backup it is a good idea to do a dump of the latest defaults so you know what has changed - if you do this each time a firmware release is created youwill be able to see the cli changes between firmware versions. For instance, in December 2014 the default GPS navigation PIDs changed. If you blindly restore your backup you would not benefit from these new defaults.

Use the CLI and send all the output from the saved backup commands.

Do not send the file too fast, if you do the FC might not be able to keep up when using USART adapters (including built in ones) since there is no hardware serial flow control.

You may find you have to copy/paste a few lines at a time.

Repeat the backup process again!

Compare the two backups to make sure you are happy with your restored settings.

Re-apply any new defaults as desired.

CLI Command Reference

`Command`	Description
`1wire <esc>`	passthrough 1wire to the specified esc
`adjrange`	show/set adjustment ranges settings
`aux`	show/set aux settings
`mmix`	design custom motor mixer
`smix`	design custom servo mixer
`color`	configure colors
`defaults`	reset to defaults and reboot
`dump`	print configurable settings in a pastable form
`exit`	
`feature`	list or -val or val
`get`	get variable value
`gpspassthrough`	passthrough gps to serial
`help`	
`led`	configure leds
`map`	mapping of rc channel order
`mixer`	mixer name or list
`motor`	get/set motor output value
`play_sound`	index, or none for next
`profile`	index (0 to 2)
`rateprofile`	index (0 to 2)
`rxrange`	configure rx channel ranges (end-points)
`save`	save and reboot
`set`	name=value or blank or * for list
`status`	show system status
`version`	

CLI Variable Reference

`Variable`	Description/Units	Min	Max	Default	Type	Datatype
`looptime`	This is the main loop time (in us). Changing this affects PID effect with some PID controllers (see PID section for details). Default of 3500us/285Hz should work for everyone. Setting it to zero does not limit loop time, so it will go as fast as possible.	0	9000	3500	Master	UINT16
`emf_avoidance`	Default value is 0 for 72MHz processor speed. Setting this to 1 increases the processor speed, to move the 6th harmonic away from 432MHz.	OFF	ON	OFF	Master	UINT8
`i2c_overclock`	Default value is 0 for disabled. Enabling this feature speeds up IMU speed significantly and faster looptimes are possible.	OFF	ON	OFF	Master	UINT8
`mid_rc`	This is an important number to set in order to avoid trimming receiver/transmitter. Most standard receivers will have this at 1500, however Futaba transmitters will need this set to 1520. A way to find out if this needs to be changed, is to clear all trim/subtrim on transmitter, and connect to GUI. Note the value most channels idle at - this should be the number to choose. Once midrc is set, use subtrim on transmitter to make sure all channels (except throttle of course) are centered at midrc value.	1200	1700	1500	Master	UINT16

`min_check`	These are min/max values (in us) which, when a channel is smaller (min) or larger (max) than the value will activate various RC commands, such as arming, or stick configuration. Normally, every RC channel should be set so that min = 1000us, max = 2000us. On most transmitters this usually means 125% endpoints. Default check values are 100us above/below this value.	0	2000	1100	Master	UINT16
`max_check`	These are min/max values (in us) which, when a channel is smaller (min) or larger (max) than the value will activate various RC commands, such as arming, or stick configuration. Normally, every RC channel should be set so that min = 1000us, max = 2000us. On most transmitters this usually means 125% endpoints. Default check values are 100us above/below this value.	0	2000	1900	Master	UINT16
`rssi_channel`		0	18	0	Master	INT8
`rssi_scale`		1	255	30	Master	UINT8
`rssi_ppm_invert`		0	1	0	Master	UINT8
`input_filtering_mode`		0	1	0	Master	INT8
`rc_smoothing`	Interpolation of Rc data during looptimes when there are no new updates. This gives smoother RC input to PID controller and cleaner PIDsum	OFF	ON	ON	Master	INT8
`min_throttle`	These are min/max values (in us) that are sent to esc when armed. Defaults of 1150/1850 are OK for everyone, for use with AfroESC, they could be set to 1064/1864.	0	2000	1150	Master	UINT16
`max_throttle`	These are min/max values (in us) that are sent to esc when armed. Defaults of 1150/1850 are OK for everyone, for use with AfroESC, they could be set to 1064/1864.	0	2000	1850	Master	UINT16
`min_command`	This is the PWM value sent to ESCs when they are not armed. If ESCs beep slowly when powered up, try decreasing this value. It can also be used for calibrating all ESCs at once.	0	2000	1000	Master	UINT16
`servo_center_pulse`		0	2000	1500	Master	UINT16
`3d_deadband_low`		0	2000	1406	Master	UINT16
`3d_deadband_high`		0	2000	1514	Master	UINT16
`3d_neutral`		0	2000	1460	Master	UINT16
`3d_deadband_throttle`		0	2000	50	Master	UINT16
`motor_pwm_rate`	Output frequency (in Hz) for motor pins. Defaults are 400Hz for motor. If setting above 500Hz, will switch to brushed (direct drive) motors mode. For example, setting to 8000 will use brushed mode at 8kHz switching frequency. Up to 32kHz is supported. Note, that in brushed mode, minthrottle is offset to zero. Default is 16000 for boards with brushed motors.	50	32000	400	Master	UINT16
`servo_pwm_rate`	Output frequency (in Hz) servo pins. Default is 50Hz. When using tricopters or gimbal with digital servo, this rate can be increased. Max of 498Hz (for 500Hz pwm period), and min of 50Hz. Most digital servos will support for example 330Hz.	50	498	50	Master	UINT16
`servo_lowpass_freq`	Selects the servo PWM output cutoff frequency. Valid values range from 10 to 400. This is a fraction of the loop frequency in 1/1000ths. For example, `40` means `0.040`. The cutoff frequency can be determined by the following formula: `Frequency = 1000 * servo_lowpass_freq / looptime`	10	400	400	Master	INT16
`servo_lowpass_enable`	Disabled by default.	OFF	ON	OFF	Master	INT8
`retarded_arm`	Disabled by default, enabling (setting to 1) allows disarming by throttle low + roll. This could be useful for mode-1 users and non-acro tricopters, where default arming by yaw could move tail servo too much.	OFF	ON	OFF	Master	UINT8
`disarm_kill_switch`	Enabled by default. Disarms the motors independently of throttle value. Setting to 0 reverts to the old behaviour of disarming only when the throttle is low. Only applies when arming and disarming with an AUX channel.	OFF	ON	ON	Master	UINT8
`auto_disarm_delay`		0	60	5	Master	UINT8
`small_angle`	If the copter tilt angle exceed this value the copter will refuse to arm. default is 25°.	0	180	25	Master	UINT8
	If enabled, the copter will process					

`pid_at_min_throttle`	the pid algorithm at minimum throttle. Cannot be used when `retarded_arm` is enabled.	OFF	ON	ON	Master	UINT8
`flaps_speed`		0	100	0	Master	UINT8
`fixedwing_althold_dir`		-1	1	1	Master	INT8
`reboot_character`		48	126	82	Master	UINT8
`gps_provider`	NMEA, UBLOX			NMEA	Master	UINT8
`gps_sbas_mode`	EGNOS, WAAS, MSAS, GAGAN			AUTO	Master	UINT8
`gps_auto_config`		OFF	ON	ON	Master	UINT8
`gps_auto_baud`		OFF	ON	OFF	Master	UINT8
`gps_pos_p`		0	200	15	Profile	UINT8
`gps_pos_i`		0	200	0	Profile	UINT8
`gps_pos_d`		0	200	0	Profile	UINT8
`gps_posr_p`		0	200	34	Profile	UINT8
`gps_posr_i`		0	200	14	Profile	UINT8
`gps_posr_d`		0	200	53	Profile	UINT8
`gps_nav_p`		0	200	25	Profile	UINT8
`gps_nav_i`		0	200	33	Profile	UINT8
`gps_nav_d`		0	200	83	Profile	UINT8
`gps_wp_radius`		0	2000	200	Profile	UINT16
`nav_controls_heading`		OFF	ON	ON	Profile	UINT8
`nav_speed_min`		10	2000	100	Profile	UINT16
`nav_speed_max`		10	2000	300	Profile	UINT16
`nav_slew_rate`		0	100	30	Profile	UINT8
`serialrx_provider`	When feature SERIALRX is enabled, this allows connection to several receivers which output data via digital interface resembling serial. See RX section.	0	6	0	Master	UINT8
`spektrum_sat_bind`		0	10	0	Master	UINT8
`telemetry_switch`	Which aux channel to use to change serial output & baud rate (MSP / Telemetry). It disables automatic switching to Telemetry when armed.	OFF	ON	OFF	Master	UINT8
`telemetry_inversion`		OFF	ON	OFF	Master	UINT8
`frsky_default_lattitude`		-90	90	0	Master	FLOAT
`frsky_default_longitude`		-180	180	0	Master	FLOAT
`frsky_coordinates_format`		0	1	0	Master	UINT8
`frsky_unit`				*	Master	UINT8
`battery_capacity`		0	20000	0	Master	UINT16
`vbat_scale`	Result is Vbatt in 0.1V steps. 3.3V = ADC Vref, 4095 = 12bit adc, 110 = 11:1 voltage divider (10k:1k) x 10 for 0.1V. Adjust this slightly if reported pack voltage is different from multimeter reading. You can get current voltage by typing "status"" in cli."	0	255	110	Master	UINT8
`vbat_max_cell_voltage`	Maximum voltage per cell, used for auto-detecting battery voltage in 0.1V units, default is 43 (4.3V)	10	50	43	Master	UINT8
`vbat_min_cell_voltage`	Minimum voltage per cell, this triggers battery out alarms, in 0.1V units, default is 33 (3.3V)	10	50	33	Master	UINT8
`vbat_warning_cell_voltage`		10	50	35	Master	UINT8
`current_meter_scale`	This sets the output voltage to current scaling for the current sensor in 0.1 mV/A steps. 400 is 40mV/A such as the ACS756 sensor outputs. 183 is the setting for the Überdistro with a 0.25mOhm shunt.	-10000	10000	400	Master	INT16
`current_meter_offset`	This sets the output offset voltage of the current sensor in millivolts.	0	3300	0	Master	UINT16
`multiwii_current_meter_output`	Default current output via MSP is in 0.01A steps. Setting this to 1 causes output in default multiwii scaling (1mA steps).	OFF	ON	OFF	Master	UINT8
`current_meter_type`		0	2	1	Master	UINT8
`align_gyro`	When running on non-default hardware or adding support for new sensors/sensor boards, these values are used for sensor orientation. When carefully understood, these values can also be used to rotate (in 90deg steps) or flip the board.			DEFAULT	Master	UINT8

align_acc	When running on non-default hardware or adding support for new sensors/sensor boards, these values are used for sensor orientation. When carefully understood, these values can also be used to rotate (in 90deg steps) or flip the board.			DEFAULT	Master	UINT8
align_mag	When running on non-default hardware or adding support for new sensors/sensor boards, these values are used for sensor orientation. When carefully understood, these values can also be used to rotate (in 90deg steps) or flip the board.			DEFAULT	Master	UINT8
align_board_roll	Arbitrary board rotation in degrees, to allow mounting it sideways / upside down / rotated etc	-180	360	0	Master	INT16
align_board_pitch	Arbitrary board rotation in degrees, to allow mounting it sideways / upside down / rotated etc	-180	360	0	Master	INT16
align_board_yaw	Arbitrary board rotation in degrees, to allow mounting it sideways / upside down / rotated etc	-180	360	0	Master	INT16
max_angle_inclination	This setting controls max inclination (tilt) allowed in angle (level) mode. default 500 (50 degrees).	100	900	500	Master	UINT16
gyro_lpf	Hardware lowpass filter for gyro. Allowed values depend on the driver - For example MPU6050 allows 5,10,20,42,98,188,256Hz, while MPU3050 doesn't allow 5Hz. If you have to set gyro lpf below 42Hz generally means the frame is vibrating too much, and that should be fixed first. Values outside of supported range will usually be ignored by drivers, and will configure lpf to default value of 42Hz.	0	256	42	Master	UINT16
moron_threshold	When powering up, gyro bias is calculated. If the model is shaking/moving during this initial calibration, offsets are calculated incorrectly, and could lead to poor flying performance. This threshold (default of 32) means how much average gyro reading could differ before re-calibration is triggered.	0	128	32	Master	UINT8
gyro_cmpf_factor	This setting controls the Gyro Weight for the Gyro/Acc complementary filter. Increasing this value reduces and delays Acc influence on the output of the filter.	100	1000	600	Master	UINT16
gyro_cmpfm_factor	This setting controls the Gyro Weight for the Gyro/Magnetometer complementary filter. Increasing this value reduces and delays the Magnetometer influence on the output of the filter.	100	1000	250	Master	UINT16
alt_hold_deadband		1	250	40	Profile	UINT8
alt_hold_fast_change		OFF	ON	ON	Profile	UINT8
deadband	These are values (in us) by how much RC input can be different before it's considered valid. For transmitters with jitter on outputs, this value can be increased. Defaults are zero, but can be increased up to 10 or so if rc inputs twitch while idle.	0	32	0	Profile	UINT8
yaw_deadband	These are values (in us) by how much RC input can be different before it's considered valid. For transmitters with jitter on outputs, this value can be increased. Defaults are zero, but can be increased up to 10 or so if rc inputs twitch while idle.	0	100	0	Profile	UINT8
throttle_correction_value	The throttle*correction*value will be added to the throttle input. It will be maximal at the throttle*correction*angle and over, null when the copter is leveled and proportional in bewteen. The angle is set with 0.1 deg steps from 1 to 900, ie : 300 = 30.0 deg, 225 = 22.5 deg.	0	150	0	Profile	UINT8
throttle_correction_angle	The throttle*correction*value will be added to the throttle input. It will be maximal at the throttle*correction*angle and over, null when the copter is leveled and proportional in bewteen. The angle is set with 0.1 deg steps from 1 to 900, ie : 300 = 30.0 deg, 225 = 22.5 deg.	1	900	800	Profile	UINT16
yaw_control_direction		-1	1	1	Master	INT8
yaw_motor_direction		-1	1	1	Profile	INT8
yaw_jump_prevention_limit	Prevent yaw jumps during yaw stops. To disable set to 500.	80	500	200	Master	UINT16
tri_unarmed_servo	On tricopter mix only, if this is set to 1, servo will always be correcting regardless of armed state. to disable this, set it to 0.	OFF	ON	ON	Profile	INT8
default_rate_profile	Default = profile number	0	2		Profile	UINT8

`rc_rate`		0	250	90	Rate Profile	UINT8
`rc_expo`		0	100	65	Rate Profile	UINT8
`rc_yaw_expo`		0	100	0	Rate Profile	UINT8
`thr_mid`		0	100	50	Rate Profile	UINT8
`thr_expo`		0	100	0	Rate Profile	UINT8
`roll_pitch_rate`		0	100	0	Rate Profile	UINT8
`yaw_rate`		0	100	0	Rate Profile	UINT8
`tpa_rate`	Throttle PID attenuation reduces influence of P on ROLL and PITCH as throttle increases. For every 1% throttle after the TPA breakpoint, P is reduced by the TPA rate.	0	100	0	Rate Profile	UINT8
`tpa_breakpoint`	See tpa_rate.	1000	2000	1500	Rate Profile	UINT16
`failsafe_delay`	Time in deciseconds to wait before activating failsafe when signal is lost. See Failsafe documentation.	0	200	10	Profile	UINT8
`failsafe_off_delay`	Time in deciseconds to wait before turning off motors when failsafe is activated. See Failsafe documentation.	0	200	200	Profile	UINT8
`failsafe_throttle`	Throttle level used for landing when failsafe is enabled. See Failsafe documentation.	1000	2000	1000	Profile	UINT16
`rx_min_usec`	Defines the shortest pulse width value used when ensuring the channel value is valid. If the receiver gives a pulse value lower than this value then the channel will be marked as bad and will default to the value of `mid_rc`.	100	2000	885	Profile	UINT16
`rx_max_usec`	Defines the longest pulse width value used when ensuring the channel value is valid. If the receiver gives a pulse value higher than this value then the channel will be marked as bad and will default to the value of `mid_rc`.	100	3000	2115	Profile	UINT16
`gimbal_mode`	When feature SERVO_TILT is enabled, this can be either NORMAL or MIXTILT			NORMAL	Profile	UINT8
`acc_hardware`	This is used to suggest which accelerometer driver should load, or to force no accelerometer in case gyro-only flight is needed. Default (0) will attempt to auto-detect among enabled drivers. Otherwise, to force a particular device, set it to 2 for ADXL345, 3 for MPU6050 integrated accelerometer, 4 for MMA8452, 5 for BMA280, 6 for LSM303DLHC, 7 for MPU6000, 8 for MPU6500 or 1 to disable accelerometer alltogether - resulting in gyro-only operation.	0	9	0	Master	UINT8
`acc_lpf_factor`	This setting controls the Low Pass Filter factor for ACC. Increasing this value reduces ACC noise (visible in GUI), but would increase ACC lag time. Zero = no filter	0	250	4	Profile	UINT8
`accxy_deadband`		0	100	40	Profile	UINT8
`accz_deadband`		0	100	40	Profile	UINT8
`accz_lpf_cutoff`		1	20	5	Profile	FLOAT
`acc_unarmedcal`		OFF	ON	ON	Profile	UINT8
`acc_trim_pitch`		-300	300	0	Profile	INT16
`acc_trim_roll`		-300	300	0	Profile	INT16
`baro_tab_size`		0	48	21	Profile	UINT8
`baro_noise_lpf`		0	1	0.6	Profile	FLOAT
`baro_cf_vel`		0	1	0.985	Profile	FLOAT
`baro_cf_alt`		0	1	0.965	Profile	FLOAT
`mag_hardware`	0 = Default, use whatever mag hardware is defined for your board type ; 1 = None, disable mag ; 2 = HMC5883 ; 3 = AK8975 (for versions <= 1.7.1: 1 = HMC5883 ; 2 = AK8975 ; 3 = None, disable mag)	0	3	0	Master	UINT8
`mag_declination`	Current location magnetic declination in format. For example, -6deg 37min, = for Japan. Leading zero in ddd not required. Get your local magnetic declination here: http://magnetic-declination.com/	-18000	18000	0	Profile	INT16
`pid_controller`	MW23, MWREWRITE, LUX			MWREWRITE	Profile	UINT8
`p_pitch`		0	200	40	Profile	UINT8

i_pitch		0	200	30	Profile	UINT8
d_pitch		0	200	23	Profile	UINT8
p_roll		0	200	40	Profile	UINT8
i_roll		0	200	30	Profile	UINT8
d_roll		0	200	23	Profile	UINT8
p_yaw		0	200	85	Profile	UINT8
i_yaw		0	200	45	Profile	UINT8
d_yaw		0	200	0	Profile	UINT8
p_pitchf		0	100	1.5	Profile	FLOAT
i_pitchf		0	100	0.4	Profile	FLOAT
d_pitchf		0	100	0.03	Profile	FLOAT
p_rollf		0	100	1.5	Profile	FLOAT
i_rollf		0	100	0.4	Profile	FLOAT
d_rollf		0	100	0.03	Profile	FLOAT
p_yawf		0	100	2.5	Profile	FLOAT
i_yawf		0	100	1.0	Profile	FLOAT
d_yawf		0	100	0.00	Profile	FLOAT
level_horizon		0	10	3	Profile	FLOAT
level_angle		0	10	5	Profile	FLOAT
sensitivity_horizon		0	250	75	Profile	UINT8
p_alt		0	200	50	Profile	UINT8
i_alt		0	200	0	Profile	UINT8
d_alt		0	200	0	Profile	UINT8
p_level		0	200	90	Profile	UINT8
i_level		0	200	10	Profile	UINT8
d_level		0	200	100	Profile	UINT8
p_vel		0	200	120	Profile	UINT8
i_vel		0	200	45	Profile	UINT8
d_vel		0	200	1	Profile	UINT8
dterm_cut_hz	Lowpass cutoff filter for Dterm for all PID controllers	0	200	0	Profile	UINT8
pterm_cut_hz	Lowpass cutoff filter for Pterm for all PID controllers	0	200	0	Profile	UINT8
gyro_cut_hz	Lowpass cutoff filter for gyro input	0	200	0	Profile	UINT8
yaw_jump_prevention_limit	Yaw Jump Prevention Limit, adjust this if your aircraft 'skids out'	80	500	200	Profile	UINT16
yaw_p_limit	Limiter for yaw P term. This parameter is only affecting PID controller MW23. To disable set to 500 (actual default).	100	500	500	Profile	UINT16
blackbox_rate_num		1	32	1	Master	UINT8
blackbox_rate_denom		1	32	1	Master	UINT8

Serial

Cleanflight has enhanced serial port flexibility but configuration is slightly more complex as a result.

Cleanflight has the concept of a function (MSP, GPS, Serial RX, etc) and a port (VCP, UARTx, SoftSerial x). Not all functions can be used on all ports due to hardware pin mapping, conflicting features, hardware, and software constraints.

Serial port types

- USB Virtual Com Port (VCP) - USB pins on a USB port connected directly to the processor without requiring a dedicated USB to UART adapter. VCP does not 'use' a physical UART port.
- UART - A pair of dedicated hardware transmit and receive pins with signal detection and generation done in hardware.
- SoftSerial - A pair of hardware transmit and receive pins with signal detection and generation done in software.

UART is the most efficient in terms of CPU usage. SoftSerial is the least efficient and slowest, SoftSerial should only be used for low-bandwidth usages, such as telemetry transmission.

UART ports are sometimes exposed via on-board USB to UART converters, such as the CP2102 as found on the Naze and Flip32 boards. If the flight controller does not have an on-board USB to UART converter and doesn't support VCP then an external USB to UART board is required. These are sometimes referred to as FTDI boards. FTDI is just a common manufacturer of a chip (the FT232RL) used on USB to UART boards.

When selecting a USB to UART converter choose one that has DTR exposed as well as a selector for 3.3v and 5v since they are more useful.

Examples:

- FT232RL FTDI USB To TTL Serial Converter Adapter
- USB To TTL / COM Converter Module buildin-in CP2102

Both SoftSerial and UART ports can be connected to your computer via USB to UART converter boards.

Serial Configuration

Serial port configuration is best done via the configurator.

Configure serial ports first, then enable/disable features that use the ports. To configure SoftSerial ports the SOFTSERIAL feature must be also be enabled.

Constraints

If the configuration is invalid the serial port configuration will reset to its defaults and features may be disabled.

- There must always be a port available to use for MSP/CLI.
- There is a maximum of 2 MSP ports.
- To use a port for a function, the function's corresponding feature must be also be enabled. e.g. after configuring a port for GPS enable the GPS feature.
- If SoftSerial is used, then all SoftSerial ports must use the same baudrate.
- Softserial is limited to 19200 buad.
- All telemetry systems except MSP will ignore any attempts to override the baudrate.
- MSP/CLI can be shared with EITHER Blackbox OR telemetry. In shared mode blackbox or telemetry will be output only when armed.
- Smartport telemetry cannot be shared with MSP.
- No other serial port sharing combinations are valid.
- You can use as many different telemetry systems as you like at the same time.
- You can only use each telemetry system once. e.g. FrSky telemetry cannot be used on two port, but MSP Telemetry + FrSky on different ports is fine.

Configuration via CLI

You can use the CLI for configuration but the commands are reserved for developers and advanced users.

The `serial` CLI command takes 6 arguments.

1. Identifier
2. Function bitmask (see serialPortFunction_e in the source)
3. MSP baud rate
4. GPS baud rate
5. Telemetry baud rate (auto baud allowed)
6. Blackbox baud rate

Baud Rates

The allowable baud rates are as follows:

Identifier	Baud rate
0	Auto
1	9600
2	19200
3	38400
4	57600
5	115200
6	230400
7	250000

Receivers (RX)

A receiver is used to receive radio control signals from your transmitter and convert them into signals that the flight controller can understand.

There are 3 basic types of receivers:

1. Parallel PWM Receivers
2. PPM Receivers
3. Serial Receivers

Parallel PWM Receivers

8 channel support, 1 channel per input pin. On some platforms using parallel input will disable the use of serial ports and SoftSerial making it hard to use telemetry or GPS features.

PPM Receivers

PPM is sometimes known as PPM SUM or CPPM.

12 channels via a single input pin, not as accurate or jitter free as methods that use serial communications, but readily available.

These receivers are reported working:

FrSky D4R-II http://www.frsky-rc.com/product/pro.php?pro_id=24

Graupner GR24 http://www.graupner.de/en/products/33512/product.aspx

R615X Spektrum/JR DSM2/DSMX Compatible 6Ch 2.4GHz Receiver w/CPPM http://orangerx.com/2014/05/20/r615x-spektrumjr-dsm2dsmx-compatible-6ch-2-4ghz-receiver-wcppm-2/

FrSky D8R-XP 8ch telemetry receiver, or CPPM and RSSI enabled receiver http://www.frsky-rc.com/product/pro.php?pro_id=21

Serial Receivers

Spektrum

8 channels via serial currently supported.

These receivers are reported working:

Lemon Rx DSMX Compatible PPM 8-Channel Receiver + Lemon DSMX Compatible Satellite with Failsafe http://www.lemon-rx.com/shop/index.php?route=product/product&product_id=118

S.BUS

16 channels via serial currently supported. See below how to set up your transmitter.

- You probably need an inverter between the receiver output and the flight controller. However, some flight controllers have this built in (the main port on CC3D, for example), and doesn't need one.
- Softserial ports cannot be used with SBUS because it runs at too high of a bitrate (1Mbps). Refer to the chapter specific to your board to determine which port(s) may be used.
- You will need to configure the channel mapping in the GUI (Receiver tab) or CLI (`map` command). Note that channels above 8 are mapped "straight", with no remapping.

These receivers are reported working:

FrSky X4RSB 3/16ch Telemetry Receiver http://www.frsky-rc.com/product/pro.php?pro_id=135

FrSky X8R 8/16ch Telemetry Receiver http://www.frsky-rc.com/product/pro.php?pro_id=105

Futaba R2008SB 2.4GHz S-FHSS http://www.futaba-rc.com/systems/futk8100-8j/

OpenTX S.BUS configuration

If using OpenTX set the transmitter module to D16 mode and ALSO select CH1-16 on the transmitter before binding to allow reception of all 16 channels.

OpenTX 2.09, which is shipped on some Taranis X9D Plus transmitters, has a bug - issue:1701. The bug prevents use of all 16 channels. Upgrade to the latest OpenTX version to allow correct reception of all 16 channels, without the fix you are limited to 8 channels regardless of the CH1-16/D16 settings.

XBUS

The firmware currently supports the MODE B version of the XBus protocol. Make sure to set your TX to use "MODE B" for XBUS in the TX menus! See here for info on JR's XBUS protocol: http://www.jrpropo.com/english/propo/XBus/

These receivers are reported working:

XG14 14ch DMSS System w/RG731BX XBus Receiver http://www.jramericas.com/233794/JRP00631/

There exist a remote receiver made for small BNF-models like the Align T-Rex 150 helicopter. The code also supports using the Align DMSS RJ01 receiver directly with the cleanflight software. To use this receiver you must power it with 3V from the hardware, and then connect the serial line as other serial RX receivers. In order for this receiver to work, you need to specify the XBUS*MODE*B*RJ01 for serialrx* provider. Note that you need to set your radio mode for XBUS "MODE B" also for this receiver to work. Receiver name: Align DMSS RJ01 (HER15001)

SUMD

16 channels via serial currently supported.

These receivers are reported working:

GR-24 receiver HoTT http://www.graupner.de/en/products/33512/product.aspx

Graupner receiver GR-12SH+ HoTT http://www.graupner.de/en/products/870ade17-ace8-427f-943b-657040579906/33565/product.aspx

SUMH

8 channels via serial currently supported.

SUMH is a legacy Graupner protocol. Graupner have issued a firmware updates for many recivers that lets them use SUMD instead.

MultiWii serial protocol (MSP)

Allows you to use MSP commands as the RC input. Only 8 channel support to maintain compatibility with MSP.

Configuration

There are 3 features that control receiver mode:

```
RX_PPM
RX_SERIAL
RX_PARALLEL_PWM
RX_MSP
```

Only one receiver feature can be enabled at a time.

RX signal-loss detection

The software has signal loss detection which is always enabled. Signal loss detection is used for safety and failsafe reasons.

The `rx_min_usec` and `rx_max_usec` settings helps detect when your RX stops sending any data, enters failsafe mode or when the RX looses signal.

By default, when the signal loss is detected the FC will set pitch/roll/yaw to the value configured for `mid_rc`. The throttle will be set to the value configured for `rx_min_usec` or `mid_rc` if using 3D feature.

Signal loss can be detected when:

1. no rx data is received (due to radio reception, recevier configuration or cabling issues).
2. using Serial RX and receiver indicates failsafe condition.
3. using any of the first 4 stick channels do not have a value in the range specified by `rx_min_usec` and `rx_max_usec`.

RX loss configuration

The `rxfail` cli command is used to configure per-channel rx-loss behaviour. You can use the `rxfail` command to change this behaviour. A flight channel can either be AUTOMATIC or HOLD, an AUX channel can either be SET or HOLD.

- AUTOMATIC - Flight channels are set to safe values (low throttle, mid position for yaw/pitch/roll).
- HOLD - Channel holds the last value.
- SET - Channel is set to a specific configured value.

The default mode is AUTOMATIC for flight channels and HOLD for AUX channels.

The rxfail command can be used in conjunction with mode ranges to trigger various actions.

The `rxfail` command takes 2 or 3 arguments. * Index of channel (See below) * Mode ('a' = AUTOMATIC, 'h' = HOLD, 's' = SET) * A value to use when in SET mode.

Channels are always specified in the same order, regardless of your channel mapping.

- Roll is 0
- Pitch is 1
- Yaw is 2
- Throttle is 3.
- Aux channels are 4 onwards.

Examples:

To make Throttle channel have an automatic value when RX loss is detected:

`rxfail 3 a`

To make AUX4 have a value of 2000 when RX loss is detected:

`rxfail 7 s 2000`

To make AUX8 hold it's value when RX loss is detected:

`rxfail 11 h`

WARNING: Always make sure you test the behavior is as expected after configuring rxfail settings!

`rx_min_usec`

The lowest channel value considered valid. e.g. PWM/PPM pulse length

`rx_max_usec`

The highest channel value considered valid. e.g. PWM/PPM pulse length

Serial RX

See the Serial chapter for some some RX configuration examples.

To setup spectrum on the Naze32 or clones in the GUI: 1. Start on the "Ports" tab make sure that UART2 has serial RX. If not set the checkbox, save and reboot. 2. Move to the "Configuration" page and in the upper lefthand corner choose Serial RX as the receiver type. 3. Below that choose the type of serial receiver that you are using. Save and reboot.

Using CLI: For Serial RX enable `RX_SERIAL` and set the `serialrx_provider` CLI setting as follows.

Serial RX Provider	Value
SPEKTRUM1024	0
SPEKTRUM2048	1
SBUS	2
SUMD	3
SUMH	4
XBUS*MODE*B	5
XBUS*MODE*B_RJ01	6

PPM/PWM input filtering.

Hardware input filtering can be enabled if you are experiencing interference on the signal sent via your PWM/PPM RX.

Use the `input_filtering_mode` CLI setting to select a mode.

Value	Meaning
0	Disabled
1	Enabled

Receiver configuration.

FrSky D4R-II

Set the RX for 'No Pulses'. Turn OFF TX and RX, Turn ON RX. Press and release F/S button on RX. Turn off RX.

Graupner GR-24 PWM

Set failsafe on the throttle channel in the receiver settings (via transmitter menu) to a value below `rx_min_usec` using channel mode FAILSAFE. This is the prefered way, since this is *much faster* detected by the FC then a channel that sends no pulses (OFF).

NOTE: One or more control channels may be set to OFF to signal a failsafe condition to the FC, all other channels *must* be set to either HOLD or OFF. Do **NOT USE** the mode indicated with FAILSAFE instead, as this combination is NOT handled correctly by the FC.

Receiver Channel Range Configuration.

If you have a transmitter/receiver, that output a non-standard pulse range (i.e. 1070-1930 as some Spektrum receivers) you could use rx channel range configuration to map actual range of your transmitter to 1000-2000 as expected by Cleanflight.

The low and high value of a channel range are often referred to as 'End-points'. e.g. 'End-point adjustments / EPA'.

All attempts should be made to configure your transmitter/receiver to use the range 1000-2000 *before* using this feature as you will have less preceise control if it is used.

To do this you should figure out what range your transmitter outputs and use these values for rx range configuration. You can do this in a few simple steps:

If you have used rc range configuration previously you should reset it to prevent it from altering rc input. Do so by entering the following command in CLI:

```
rxrange reset
save
```

Now reboot your FC, connect the configurator, go to the `Receiver` tab move sticks on your transmitter and note min and max values of first 4 channels. Take caution as you can accidentally arm your craft. Best way is to move one channel at a time.

Go to CLI and set the min and max values with the following command:

```
rxrange <channel_number> <min> <max>
```

For example, if you have the range 1070-1930 for the first channel you should use `rxrange 0 1070 1930` in the CLI. Be sure to enter the `save` command to save the settings.

After configuring channel ranges use the sub-trim on your transmitter to set the middle point of pitch, roll, yaw and throttle.

You can also use rxrange to reverse the direction of an input channel, e.g. `rxrange 0 2000 1000`.

Spektrum bind support

Spektrum bind with hardware bind plug support.

The Spektrum bind code is actually enabled for the NAZE, NAZE32PRO, CJMCU, EUSTM32F103RC, SPARKY, CC3D, ALIENWIIF1, ALIENWIIF3 targets.

Configure the bind code

The following parameters can be used to enable and configure this in the related target.h file:

```
SPEKTRUM_BIND          Enables the Spektrum bind code
BIND_PORT  GPIOA       Defines the port for the bind pin
BIND_PIN   Pin_3       Defines the bind pin (the satellite receiver is connected to)
```

This is to activate the hardware bind plug feature

```
HARDWARE_BIND_PLUG     Enables the hardware bind plug feature
BINDPLUG_PORT  GPIOB   Defines the port for the hardware bind plug
BINDPLUG_PIN   Pin_5   Defines the hardware bind plug pin
```

Hardware

The hardware bind plug will be enabled via defining HARDWARE*BIND*PLUG during building of the firmware. BINDPLUG*PORT and BINDPLUG*PIN also need to be defined (please see above). This is done automatically if the AlienWii32 firmware is built. The hardware bind plug is expected between the defined bind pin and ground.

Function

The bind code will actually work for NAZE, NAZE32PRO, CJMCU, EUSTM32F103RC, SPARKY targets (USART2) and CC3D target (USART3, flex port). The spektrum*sat*bind CLI parameter is defining the number of bind impulses (1-10) send to the satellite receiver. Setting spektrum*sat*bind to zero will disable the bind mode in any case. The bind mode will only be activated after an power on or hard reset. Please refer to the table below for the different possible values.

If the hardware bind plug is configured the bind mode will only be activated if the plug is set during the firmware start-up. The value of the spektrum*sat*bind parameter will be permanently preserved. The bind plug should be always removed for normal flying.

If no hardware bind plug is used the spektrum *sat*bind parameter will trigger the bind process during the next hardware reset and will be automatically reset to "0" after this.

Please refer to the satellite receiver documentation for more details of the specific receiver in bind mode. Usually the bind mode will be indicated with some flashing LEDs.

Table with spektrum*sat*bind parameter value

Value	Receiver mode	Notes
3	DSM2 1024bit/22ms	
5	DSM2 2048bit/11ms	default AlienWii32
7	DSMX 1024bit/22ms	
9	DSMX 2048bit/11ms	

More detailed information regarding the satellite binding process can be found here: http://wiki.openpilot.org/display/Doc/Spektrum+Satellite

Supported Hardware

NAZE, NAZE32PRO, CJMCU, SPARKY, EUSTM32F103RC, CC3D targets and ALIENWIIF1, ALIENWIIF3 targets with hardware bind plug

Connecting a Spektrum-compatible satellite to a Flip32+ flight controller

The Flip32/Flip32+ is wired in a rather strange way, i.e. the dedicated connector for the satellite module uses the same UART pins as the USB adapter. This means that you can't use that connector as it maps to UART1 which you really shouldn't assign to SERIAL_RX as that will break USB functionality. (Looks this problem is fixed in later versions of the Flip32/Flip32+)

In order to connect the satellite to a Flip32+, you have to wire the serial data pin to RC_CH4. This is the fourth pin from the top in the left column of the 3x6 header on the right side of the board. GND and +3.3V may either be obtained from the dedicated SAT connector or from any ground pin and pin 1 of the BOOT connector which also provides 3.3V.

Tested satellite transmitter combinations

Satellite	Remote	Remark
Orange R100	Spektrum DX6i	Bind value 3
Lemon RX DSM2/DSMX	Spektrum DX8	Bind value 5
Lemon RX DSMX	Walkera Devo10	Bind value 9, Deviation firmware 4.01 up to 12 channels
Lemon RX DSM2	Walkera Devo7	Bind value 9, Deviation firmware

Failsafe

There are two types of failsafe:

1. Receiver based failsafe
2. Flight controller based failsafe

Receiver based failsafe is where you, from your transmitter and receiver, configure channels to output desired signals if your receiver detects signal loss and goes to **rx-failsafe-state**. The idea is that you set throttle and other controls so the aircraft descends in a controlled manner. See your receiver's documentation for this method.

Flight controller based failsafe is where the flight controller attempts to detect signal loss and/or the **rx-failsafe-state** of your receiver and upon detection goes to **fc-failsafe-state**. The idea is that the flight controller starts using substitutes for all controls, which are set by you, using the CLI command `rxfail` (see `rxfail` document) or the cleanflight-configurator GUI.

It is possible to use both types at the same time, which may be desirable. Flight controller failsafe can even help if your receiver signal wires come loose, get damaged or your receiver malfunctions in a way the receiver itself cannot detect.

Alternatively you may configure a transmitter switch to activate failsafe mode. This is useful for fieldtesting the failsafe system and as a ***`PANIC`*** switch when you lose orientation.

Flight controller failsafe system

The **failsafe-auto-landing** system is not activated until 5 seconds after the flight controller boots up. This is to prevent **failsafe-auto-landing** from activating, as in the case of TX/RX gear with long bind procedures, before the RX sends out valid data.

The **failsafe-detection** system attempts to detect when your receiver loses signal *continuously* but the **failsafe-auto-landing** starts *only when your craft is* ***armed***. It then attempts to prevent your aircraft from flying away uncontrollably by enabling an auto-level mode and setting the throttle that should allow the craft to come to a safer landing.

The failsafe is activated when the craft is armed and either:

- The control (stick) channels do not have valid signals AND the failsafe guard time specified by `failsafe_delay` has elapsed.
- A transmitter switch that is configured to control the failsafe mode is switched ON (and 'failsafe*kill*switch' is set to 0).

Failsafe intervention will be aborted when it was due to:

- a lost RC signal and the RC signal has recovered.
- a transmitter failsafe switch was set to ON position and the switch is set to OFF position (and 'failsafe*kill*switch' is set to 0).

Note that: * At the end of a failsafe intervention, the flight controller will be disarmed and re-arming will be locked. From that moment on it is no longer possible to abort or re-arm and the flight controller has to be reset. * When 'failsafe*kill*switch' is set to 1 and the rc switch configured for failsafe is set to ON, the craft is instantly disarmed (but re-arming is not locked). Similar effect can be achieved by setting 'failsafe*throttle' to 1000 and 'failsafe* off*delay' to 0 (but arming is locked). * Prior to starting a failsafe intervention it is checked if the throttle position was below 'min*throttle' level for the last 'failsafe*throttle*low_delay' seconds. If it was the craft is assumed to be on the ground and is only disarmed. It may be re-armed without a power cycle.

Some notes about **SAFETY**: * The failsafe system will be activated regardless of current throttle position. So when the failsafe intervention is aborted (RC signal restored/failsafe switch set to OFF) the current stick position will direct the craft ! * The craft may already be on the ground with motors stopped and that motors and props could spin again - the software does not currently detect if the craft is on the ground. Take care when using `MOTOR_STOP` feature. **Props will spin up without warning**, when armed with `MOTOR_STOP` feature ON (props are not spinning) ***and*** failsafe is activated !

Configuration

When configuring the flight controller failsafe, use the following steps:

1. Configure your receiver to do one of the following:

- Upon signal loss, send no signal/pulses over the channels
- Send an invalid signal over the channels (for example, send values lower than 'rx*min*usec')

and

- Ensure your receiver does not send out channel data that would cause a disarm by switch or sticks to be registered by the FC. This is especially important for those using a switch to arm.

See your receiver's documentation for direction on how to accomplish one of these.

- Configure one of the transmitter switches to activate the failsafe mode.

1. Set 'failsafe*off*delay' to an appropriate value based on how high you fly

2. Set 'failsafe_throttle' to a value that allows the aircraft to descend at approximately one meter per second (default is 1000 which should be throttle off).

These are the basic steps for flight controller failsafe configuration; see Failsafe Settings below for additional settings that may be changed.

Failsafe Settings

Failsafe delays are configured in 0.1 second steps.

1 step = 0.1sec

1 second = 10 steps

`failsafe_delay`

Guard time for failsafe activation after signal lost. This is the amount of time the flight controller waits to see if it begins receiving a valid signal again before activating failsafe.

`failsafe_off_delay`

Delay after failsafe activates before motors finally turn off. This is the amount of time 'failsafe_throttle' is active. If you fly at higher altitudes you may need more time to descend safely.

`failsafe_throttle`

Throttle level used for landing. Specify a value that causes the aircraft to descend at about 1M/sec. Default is set to 1000 which should correspond to throttle off.

`failsafe_kill_switch`

Configure the rc switched failsafe action: the same action as when the rc link is lost (set to 0) or disarms instantly (set to 1). Also see above.

`failsafe_throttle_low_delay`

Time throttle level must have been below 'min*throttle' to _only disarm* instead of *full failsafe procedure*.

Use standard RX usec values. See Rx documentation.

`rx_min_usec`

The lowest channel value considered valid. e.g. PWM/PPM pulse length

`rx_max_usec`

The highest channel value considered valid. e.g. PWM/PPM pulse length

The `rx_min_usec` and `rx_max_usec` settings helps detect when your RX stops sending any data, enters failsafe mode or when the RX looses signal.

With a Graupner GR-24 configured for PWM output with failsafe on channels 1-4 set to OFF in the receiver settings then this setting, at its default value, will allow failsafe to be activated.

Testing

Bench test the failsafe system before flying - *remove props while doing so*.

1. Arm the craft.
2. Turn off transmitter or unplug RX.
3. Observe motors spin at configured throttle setting for configured duration.
4. Observe motors turn off after configured duration.
5. Ensure that when you turn on your TX again or reconnect the RX that you cannot re-arm once the motors have stopped.
6. Power cycle the FC.
7. Arm the craft.
8. Turn off transmitter or unplug RX.
9. Observe motors spin at configured throttle setting for configured duration.
10. Turn on TX or reconnect RX.
11. Ensure that your switch positions don't now cause the craft to disarm (otherwise it would fall out of the sky on regained signal).
12. Observe that normal flight behavior is resumed.
13. Disarm.

Field test the failsafe system.

1. Perform bench testing first!
2. On a calm day go to an unpopulated area away from buildings or test indoors in a safe controlled environment - e.g. inside a big net.
3. Arm the craft.
4. Hover over something soft (long grass, ferns, heather, foam, etc.).
5. Descend the craft and observe throttle position and record throttle value from your TX channel monitor. Ideally 1500 should be hover. So your value should be less than 1500.
6. Stop, disarm.
7. Set failsafe throttle to the recorded value.
8. Arm, hover over something soft again.
9. Turn off TX (!)
10. Observe craft descends and motors continue to spin for the configured duration.
11. Observe FC disarms after the configured duration.
12. Remove flight battery.

If craft descends too quickly then increase failsafe throttle setting.

Ensure that the duration is long enough for your craft to land at the altitudes you normally fly at.

Using a configured transmitter switch to activate failsafe mode, instead of switching off your TX, is good primary testing method in addition to the above procedure.

Battery Monitoring

Cleanflight has a battery monitoring feature. The voltage of the main battery can be measured by the system and used to trigger a low-battery warning buzzer, on-board status LED flashing and LED strip patterns.

Low battery warnings can:

- Help ensure you have time to safely land the aircraft
- Help maintain the life and safety of your LiPo/LiFe batteries, which should not be discharged below manufacturer recommendations

Minimum and maximum cell voltages can be set, and these voltages are used to auto-detect the number of cells in the battery when it is first connected.

Per-cell monitoring is not supported, as we only use one ADC to read the battery voltage.

Supported targets

All targets support battery voltage monitoring unless status.

Connections

When dealing with batteries **ALWAYS CHECK POLARITY!**

Measure expected voltages **first** and then connect to the flight controller. Powering the flight controller with incorrect voltage or reversed polarity will likely fry your flight controller. Ensure your flight controller has a voltage divider capable of measuring your particular battery voltage.

Naze32

The Naze32 has an on-board battery divider circuit; just connect your main battery to the VBAT connector.

CAUTION: When installing the connection from main battery to the VBAT connector, be sure to first disconnect the main battery from the frame/power distribution board. Check the wiring very carefully before connecting battery again. Incorrect connections can immediately and completely destroy the flight controller and connected peripherals (ESC, GPS, Receiver etc.).

CC3D

The CC3D has no battery divider. To use voltage monitoring, you must create a divider that gives a 3.3v MAXIMUM output when the main battery is fully charged. Connect the divider output to S5_IN/PA0/RC5.

Notes:

- S5_IN/PA0/RC5 is Pin 7 on the 8 pin connector, second to last pin, on the opposite end from the GND/+5/PPM signal input.
- When battery monitoring is enabled on the CC3D, RC5 can no-longer be used for PWM input.

Sparky

See the Sparky board chapter.

Configuration

Enable the `VBAT` feature.

Configure min/max cell voltages using the following CLI setting:

`vbat_scale` - Adjust this to match actual measured battery voltage to reported value.

`vbat_max_cell_voltage` - Maximum voltage per cell, used for auto-detecting battery voltage in 0.1V units, i.e. 43 = 4.3V

`set vbat_warning_cell_voltage` - Warning voltage per cell; this triggers battery-out alarms, in 0.1V units, i.e. 34 = 3.4V

`vbat_min_cell_voltage` - Minimum voltage per cell; this triggers battery-out alarms, in 0.1V units, i.e. 33 = 3.3V

e.g.

```
set vbat_scale = 110
set vbat_max_cell_voltage = 43
set vbat_warning_cell_voltage = 34
set vbat_min_cell_voltage = 33
```

Current Monitoring

Current monitoring (amperage) is supported by connecting a current meter to the appropriate current meter ADC input (see the documentation for your particular board).

When enabled, the following values calculated and used by the telemetry and OLED display subsystems: * Amps * mAh used * Capacity remaining

Configuration

Enable current monitoring using the CLI command:

```
feature CURRENT_METER
```

Configure the current meter type using the `current_meter_type` settings here:

Value	Sensor Type
0	None
1	ADC/hardware sensor
2	Virtual sensor

Configure capacity using the `battery_capacity` setting, in mAh units.

If you're using an OSD that expects the multiwii current meter output value, then set `multiwii_current_meter_output` to `1` (this multiplies amperage sent to MSP by 10).

ADC Sensor

The current meter may need to be configured so the value read at the ADC input matches actual current draw. Just like you need a voltmeter to correctly calibrate your voltage reading you also need an ammeter to calibrate the current sensor.

Use the following settings to adjust calibration:

`current_meter_scale` `current_meter_offset`

Virtual Sensor

The virtual sensor uses the throttle position to calculate an estimated current value. This is useful when a real sensor is not available. The following settings adjust the virtual sensor calibration:

Setting	Description
`current_meter_scale`	The throttle scaling factor [centiamps, i.e. 1/100th A]
`current_meter_offset`	The current at zero throttle (while disarmed) [centiamps, i.e. 1/100th A]

There are two simple methods to tune these parameters: one uses a battery charger and another depends on actual current measurements.

Tuning Using Actual Current Measurements

If you know your craft's current draw while disarmed (Imin) and at maximum throttle while armed (Imax), calculate the scaling factors as follows:

```
current_meter_scale = (Imax - Imin) * 100000 / (Tmax + (Tmax * Tmax / 50))
current_meter_offset = Imin * 100
```

Note: Tmax is maximum throttle offset (i.e. for `max_throttle` = 1850, Tmax = 1850 - 1000 = 850)

For example, assuming a maximum current of 34.2A, a minimum current of 2.8A, and a Tmax `max_throttle` = 1850:

```
current_meter_scale = (Imax - Imin) * 100000 / (Tmax + (Tmax * Tmax / 50))
                    = (34.2 - 2.8) * 100000 / (850 + (850 * 850 / 50))
                    = 205
current_meter_offset = Imin * 100 = 280
```

Tuning Using Battery Charger Measurement

If you cannot measure current draw directly, you can approximate it indirectly using your battery charger.
However, note it may be difficult to adjust `current_meter_offset` using this method unless you can measure the actual current draw with the craft disarmed.

Note: + This method depends on the accuracy of your battery charger; results may vary. + If you add or replace equipment that changes the in-flight current draw (e.g. video transmitter, camera, gimbal, motors, prop pitch/sizes, ESCs, etc.), you should recalibrate.

The general method is:

1. Fully charge your flight battery
2. Fly your craft, using >50% of your battery pack capacity (estimated)
3. Note Cleanflight's reported mAh draw
4. Re-charge your flight battery, noting the mAh charging data needed to restore the pack to fully charged
5. Adjust `current_meter_scale` to according to the formula given below
6. Repeat and test

Given (a) the reported mAh draw and the (b) mAh charging data, calculate a new `current_meter_scale` value as follows:

```
current_meter_scale = (charging_data_mAh / reported_draw_mAh) * old_current_meter_scale
```

For example, assuming: + A Cleanflight reported current draw of 1260 mAh + Charging data to restore full charge of 1158 mAh + A existing `current_meter_scale` value of 400 (the default)

Then the updated `current_meter_scale` is:

```
current_meter_scale = (charging_data_mAh / reported_draw_mAh) * old_current_meter_scale
                    = (1158 / 1260) * 400
                    = 368
```

GPS

GPS features in Cleanflight are experimental. Please share your findings with the developers.

GPS works best if the GPS receiver is mounted above and away from other sources of interference.

The compass/mag sensor should be well away from sources of magnetic interference, e.g. keep it away from power wires, motors, ESCs.

Two GPS protocols are supported. NMEA text and UBLOX binary.

Configuration

Enable the GPS from the CLI as follows:

1. configure a serial port to use for GPS.
2. set your GPS baud rate
3. enable the `feature GPS`
4. set the `gps_provider`
5. connect your GPS to the serial port configured for GPS.
6. save and reboot.

Note: GPS packet loss has been observed at 115200. Try using 57600 if you experience this.

For the connections step check the Board documentation for pins and port numbers.

GPS Provider

Set the `gps_provider` appropriately.

Value	Meaning
0	NMEA

1	UBLOX

GPS Auto configuration

When using UBLOX it is a good idea to use GPS auto configuration so your FC gets the GPS messages it needs.

Enable GPS auto configuration as follows `set gps_auto_config=1`.

If you are not using GPS auto configuration then ensure your GPS receiver sends out the correct messages at the right frequency. See below for manual UBlox settings.

SBAS

When using a UBLOX GPS the SBAS mode can be configured using `gps_sbas_mode`.

The default is AUTO.

Value	Meaning	Region
0	AUTO	Global
1	EGNOS	Europe
2	WAAS	North America
3	MSAS	Asia
4	GAGAN	India

If you use a regional specific setting you may achieve a faster GPS lock than using AUTO.

This setting only works when `gps_auto_config=1`

GPS Receiver Configuration

UBlox GPS units can either be configured using the FC or manually.

UBlox GPS manual configuration

Use UBox U-Center and connect your GPS to your computer. The CLI `gpspassthrough` command may be of use if you do not have a spare USART to USB adapter.

Note that many boards will not provide +5V from USB to the GPS module, such as the SPRacingF3; if you are using `gpspassthrough` you may need to connect a BEC to the controller if your board permits it, or use a standalone UART adapter. Check your board documentation to see if your GPS port is powered from USB.

Display the Packet Console (so you can see what messages your receiver is sending to your computer).

Display the Configation View.

Navigate to CFG (Configuration)

Select `Revert to default configuration`. Click `Send`.

At this point you might need to disconnect and reconnect at the default baudrate - probably 9600 baud.

Navigate to PRT (Ports)

Set `Target` to `1 - Uart 1` Set `Protocol In` to `0+1+2` Set `Protocol Out` to `0+1` Set `Buadrate` to `57600` `115200` Press `Send`

This will immediatly "break" communication to the GPS. Since you haven't saved the new baudrate setting to the non-volatile memory you need to change the baudrate you communicate to the GPS without resetting the GPS. So `Disconnect`, Change baud rate to match, then `Connect`.

Click on `PRT` in the Configuration view again and inspect the packet console to make sure messages are being sent and acknowledged.

Next, to ensure the FC doesn't waste time processing unneeded messages, click on `MSG` and enable the following on UART1 alone with a rate of 1. When changing message target and rates remember to click `Send` after changing each message.:

```
NAV-POSLLH
NAV-DOP
NAV-SOL
NAV-VELNED
NAV-TIMEUTC
```

Enable the following on UART1 with a rate of 5, to reduce bandwidth and load on the FC.

```
NAV-SVINFO
```

All other message types should be disabled.

Next change the global update rate, click `Rate (Rates)` in the Configuration view.

Set `Measurement period` to `100` ms. Set `Navigation rate` to `1`. Click `Send`.

This will cause the GPS receive to send the require messages out 10 times a second. If your GPS receiver cannot be set to use `100`ms try `200`ms (5hz) - this is less precise.

Next change the mode, click `NAV5 (Navigation 5)` in the Configuration View.

Set to `Dynamic Model` to `Pedestrian` and click `Send`.

Next change the SBAS settings. Click `SBAS (SBAS Settings)` in the Configuration View.

Set `Subsystem` to `Enabled`. Set `PRN Codes` to `Auto-Scan`. Click `Send`.

Finally, we need to save the configuration.

Click `CFG (Configuration` in the Configuration View.

Select `Save current configuration` and click `Send`.

UBlox Navigation model

Cleanflight will use `Pedestrian` when gps auto config is used.

From the UBlox documentation:

- Pedestrian - Applications with low acceleration and speed, e.g. how a pedestrian would move. Low acceleration assumed. MAX Altitude [m]: 9000, MAX Velocity [m/s]: 30, MAX Vertical, Velocity [m/s]: 20, Sanity check type: Altitude and Velocity, Max Position Deviation: Small.
- Portable - Applications with low acceleration, e.g. portable devices. Suitable for most situations. MAX Altitude [m]: 12000, MAX Velocity [m/s]: 310, MAX Vertical Velocity [m/s]: 50, Sanity check type: Altitude and Velocity, Max Position Deviation: Medium.
- Airborne < 1G - Used for applications with a higher dynamic range and vertical acceleration than a passenger car. No 2D position fixes supported. MAX Altitude [m]: 50000, MAX Velocity [m/s]: 100, MAX Vertical Velocity [m/s]: 100, Sanity check type: Altitude, Max Position Deviation: Large

Hardware

There are many GPS receivers available on the market. Below are some examples of user-tested hardware.

Ublox

U-Blox

NEO-M8

Module	Comments
U-blox Neo-M8N w/Compass	
Reyax RY825AI	NEO-M8N, 18Hz UART USB interface GPS Glonass BeiDou QZSS antenna module flash. eBay

NEO-7

Module	Comments
U-blox Neo-7M w/Compass	Hobby King

NEO-6

Module	Comments
Ublox NEO-6M GPS with Compass	eBay

Serial NMEA

MediaTek

Module	Comments
MTK 3329	Tested on hardware serial at 115200 baud (default) and on softserial at 19200 baud. The baudrate and refresh rate can be adjusted using the MiniGPS software (recommended if you lower the baudrate). The software will estimate the percentage of UART bandwidth used for your chosen baudrate and update rate.

RSSI

RSSI is a measurement of signal strength and is very handy so you know when your aircraft isw going out of range or if it is suffering RF interference.

Some receivers have RSSI outputs. 3 types are supported.

1. RSSI via PPM channel
2. RSSI via Parallel PWM channel
3. RSSI via ADC with PPM RC that has an RSSI output - aka RSSI ADC

RSSI via PPM

Configure your receiver to output RSSI on a spare channel, then select the channel used via the CLI.

e.g. if you used channel 9 then you would set:

```
set rssi_channel = 9
```

Note: Some systems such as EZUHF invert the RSSI (0 = Full signal / 100 = Lost signal). To correct this problem you can invert the channel input so you will get a correct reading by using command:

```
set rssi_ppm_invert = 1
```

Default is set to "0" for normal operation (100 = Full signal / 0 = Lost signal).

RSSI via Parallel PWM channel

Connect the RSSI signal to any PWM input channel then set the RSSI channel as you would for RSSI via PPM

RSSI ADC

Connect the RSSI signal to the RC2/CH2 input. The signal must be between 0v and 3.3v. Use inline resistors to lower voltage if required; inline smoothing capacitors may also help. A simple PPM->RSSI conditioner can easily be made. See the PPM-RSSI conditioning.pdf for details.

Under CLI : - enable using the RSSI*ADC feature : `feature RSSI*ADC` - set the RSSI_SCALE parameter (between 1 and 255) to adjust RSSI level according to your configuration.

FrSky D4R-II and X8R supported.

The feature can not be used when RX*PARALLEL*PWM is enabled.

Telemetry

Telemetry allows you to know what is happening on your aircraft while you are flying it. Among other things you can receive battery voltages and GPS positions on your transmitter.

Telemetry can be either always on, or enabled when armed. If a serial port for telemetry is shared with other functionality then then telemetry will only be enabled when armed on that port.

Telemetry is enabled using the 'TELEMETRY` feature.

```
feature TELEMETRY
```

Multiple telemetry providers are currently supported, FrSky, Graupner HoTT V4, SmartPort (S.Port) and MultiWii Serial Protocol (MSP)

All telemetry systems use serial ports, configure serial ports to use the telemetry system required.

FrSky telemetry

FrSky telemetry is transmit only and just requires a single connection from the TX pin of a serial port to the RX pin on an FrSky telemetry receiver.

FrSky telemetry signals are inverted. To connect a cleanflight capable board to an FrSKy receiver you have some options.

1. A hardware inverter - Built in to some flight controllers.
2. Use software serial and enable frsky_inversion.
3. Use a flight controller that has software configurable hardware inversion (e.g. STM32F30x).

For 1, just connect your inverter to a usart or software serial port.

For 2 and 3 use the CLI command as follows:

```
set telemetry_inversion = 1
```

Precision setting for VFAS

Cleanflight can send VFAS (FrSky Ampere Sensor Voltage) in two ways:

```
set frsky_vfas_precision  = 0
```

This is default setting which supports VFAS resolution of 0.2 volts and is supported on all FrSky hardware.

```
set frsky_vfas_precision  = 1
```

This is new setting which supports VFAS resolution of 0.1 volts and is only supported by OpenTX radios (this method uses custom ID 0x39).

Notes

RPM shows throttle output when armed. RPM shows when disarmed. TEMP2 shows Satellite Signal Quality when GPS is enabled.

RPM requires that the 'blades' setting is set to 12 on your receiver/display - tested with Taranis/OpenTX.

HoTT telemetry

Only Electric Air Modules and GPS Modules are emulated.

Use the latest Graupner firmware for your transmitter and receiver.

Older HoTT transmitters required the EAM and GPS modules to be enabled in the telemetry menu of the transmitter. (e.g. on MX-20)

Serial ports use two wires but HoTT uses a single wire so some electronics are required so that the signals don't get mixed up. The TX and RX pins of a serial port should be connected using a diode and a single wire to the `T` port on a HoTT receiver.

Connect as follows:

- HoTT TX/RX `T` -> Serial RX (connect directly)
- HoTT TX/RX `T` -> Diode `-(  |)-` > Serial TX (connect via diode)

The diode should be arranged to allow the data signals to flow the right way

```
-(  |)- == Diode, | indicates cathode marker.
```

1N4148 diodes have been tested and work with the GR-24.

As noticed by Skrebber the GR-12 (and probably GR-16/24, too) are based on a PIC 24FJ64GA-002, which has 5V tolerant digital pins.

Note: The SoftSerial ports may not be 5V tolerant on your board. Verify if you require a 5v/3.3v level shifters.

MultiWii Serial Protocol (MSP)

MSP Telemetry simply transmits MSP packets in sequence to any MSP device attached to the telemetry port. It rotates though a fixes sequence of command responses.

It is transmit only, it can work at any supported baud rate.

SmartPort (S.Port)

Smartport is a telemetry system used by newer FrSky transmitters and receivers such as the Taranis/XJR and X8R, X6R and X4R(SB).

More information about the implementation can be found here: https://github.com/frank26080115/cleanflight/wiki/Using-Smart-Port

In time this documentation will be updated with further details.

Smartport devices can be connected directly to STM32F3 boards such as the SPRacingF3 and Sparky, with a single straight through cable without the need for any hardware modifications on the FC or the receiver.

For Smartport on F3 based boards, enable the telemetry inversion setting.

```
set telemetry_inversion = 1
```

LED Strip

Cleanflight supports the use of addressable LED strips. Addressable LED strips allow each LED in the strip to be programmed with a unique and independant color. This is far more advanced than the normal RGB strips which require that all the LEDs in the strip show the same color.

Addressable LED strips can be used to show information from the flight controller system, the current implementation supports the following:

- Up to 32 LEDs.
- Indicators showing pitch/roll stick positions.
- Heading/Orientation lights.
- Flight mode specific color schemes.
- Low battery warning.
- AUX operated on/off switch

The function and orientation configuration is fixed for now but later it should be able to be set via the UI or CLI..

In the future, if someone codes it, they could be used to show GPS navigation status, thrust levels, RSSI, etc. Lots of scope for ideas and improvements.

Likewise, support for more than 32 LEDs is possible, it just requires additional development.

Supported hardware

Only strips of 32 WS2811/WS2812 LEDs are supported currently. If the strip is longer than 32 LEDs it does not matter, but only the first 32 are used.

WS2812 LEDs require an 800khz signal and precise timings and thus requires the use of a dedicated hardware timer.

Note: Not all WS2812 ICs use the same timings, some batches use different timings.

It could be possible to be able to specify the timings required via CLI if users request it.

Tested Hardware

- Adafruit NeoPixel Jewel 7 (preliminary testing)
  - Measured current consumption in all white mode ~ 350 mA.
  - Fits well under motors on mini 250 quads.
- Adafruit NeoPixel Stick (works well)
  - Measured current consumption in all white mode ~ 350 mA.

Connections

WS2812 LED strips generally require a single data line, 5V and GND.

WS2812 LEDs on full brightness can consume quite a bit of current. It is recommended to verify the current draw and ensure your supply can cope with the load. On a multirotor that uses multiple BEC ESC's you can try use a different BEC to the one the FC uses. e.g. ESC1/BEC1 -> FC, ESC2/BEC2 -> LED strip. It's also possible to power one half of the strip from one BEC and the other half from another BEC. Just ensure that the GROUND is the same for all BEC outputs and LEDs.

Target	Pin	LED Strip	Signal
Naze/Olimexino	RC5	Data In	PA6
CC3D	RCO5	Data In	PB4
ChebuzzF3/F3Discovery	PB8	Data In	PB8
Sparky	PWM5	Data In	PA6

Since RC5 is also used for SoftSerial on the Naze/Olimexino it means that you cannot use SoftSerial and led strips at the same time. Additionally, since RC5 is also used for Parallel PWM RC input on both the Naze, Chebuzz and STM32F3Discovery targets, led strips can not be used at the same time at Parallel PWM.

If you have LEDs that are intermittent, flicker or show the wrong colors then drop the VIN to less than 4.7v, e.g. by using an inline diode on the VIN to the LED strip. The problem occurs because of the difference in voltage between the data signal and the power signal. The WS2811 LED's require the data signal (Din) to be between 0.3 * Vin (Max) and 0.7 * VIN (Min) to register valid logic low/high signals. The LED pin on the CPU will always be between 0v to ~3.3v, so the Vin should be 4.7v (3.3v / 0.7 = 4.71v). Some LEDs are more tolerant of this than others.

The datasheet can be found here: http://www.adafruit.com/datasheets/WS2812.pdf

Configuration

The led strip feature can be configured via the GUI

GUI: Enable the Led Strip feature via the GUI under setup.

Configure the leds from the Led Strip tab in the cleanflight GUI. First setup how the led's are laid out so that you can visualize it later as you configure and so the flight controller knows how many led's there are available.

There is a step by step guide on how to use the GUI to configure the Led Strip feature using the GUI http://blog.oscarliang.net/setup-rgb-led-cleanflight/ which was published early 2015 by Oscar Liang which may or may not be up-to-date by the time you read this.

CLI: Enable the `LED_STRIP` feature via the cli:

```
feature LED_STRIP
```

If you enable LED_STRIP feature and the feature is turned off again after a reboot then check your config does not conflict with other features, as above.

Configure the LEDs using the `led` command.

The `led` command takes either zero or two arguments - an zero-based led number and a sequence which indicates pair of coordinates, direction flags and mode flags and a color.

If used with zero arguments it prints out the led configuration which can be copied for future reference.

Each led is configured using the following template: `x,y:ddd:mmm:cc`

`x` and `y` are grid coordinates of a 0 based 16x16 grid, north west is 0,0, south east is 15,15 `ddd` specifies the directions, since an led can face in any direction it can have multiple directions. Directions are:

`N` - North `E` - East `S` - South `W` - West `U` - Up `D` - Down

For instance, an LED that faces South-east at a 45 degree downwards angle could be configured as `SED`.

Note: It is perfectly possible to configure an LED to have all directions `NESWUD` but probably doesn't make sense.

`mmm` specifies the modes that should be applied an LED. Modes are:

- `W` - `W`warnings.
- `F` - `F`light mode & Orientation
- `I` - `I`ndicator.
- `A` - `A`rmed state.
- `T` - `T`hrust state.
- `R` - `R`ing thrust state.
- `C` - `C`olor.

`cc` specifies the color number (0 based index).

Example:

```
led 0 0,15:SD:IAW:0
led 1 15,0:ND:IAW:0
led 2 0,0:ND:IAW:0
led 3 0,15:SD:IAW:0
led 4 7,7::C:1
led 5 8,8::C:2
```

To erase an led, and to mark the end of the chain, use `0,0::` as the second argument, like this:

```
led 4 0,0:::
```

It is best to erase all LEDs that you do not have connected.

Modes

Warning

This mode simply uses the LEDs to flash when warnings occur.

Warning	LED Pattern	Notes
Arm-lock enabled	flash between green and off	occurs calibration or when unarmed and the aircraft is tilted too much
Low Battery	flash red and off	battery monitoring must be enabled. May trigger temporarily under high-throttle due to voltage drop
Failsafe	flash between light blue and yellow	Failsafe must be enabled

Flash patterns appear in order, so that it's clear which warnings are enabled.

Flight Mode & Orientation

This mode shows the flight mode and orientation.

When flight modes are active then the LEDs are updated to show different colors depending on the mode, placement on the grid and direction.

LEDs are set in a specific order: * LEDs that marked as facing up or down. * LEDs that marked as facing west or east AND are on the west or east side of the grid. * LEDs that marked as facing north or south AND are on the north or south side of the grid.

That is, south facing LEDs have priority.

The mapping between modes led placement and colors is currently fixed and cannot be changed.

Indicator

This mode flashes LEDs that correspond to roll and pitch stick positions. i.e. they indicate the direction the craft is going to turn.

Mode	Direction	LED Color
Orientation	North	WHITE
Orientation	East	DARK VIOLET
Orientation	South	RED
Orientation	West	DEEP PINK
Orientation	Up	BLUE
Orientation	Down	ORANGE

Head Free	North	LIME GREEN
Head Free	East	DARK VIOLET
Head Free	South	ORANGE
Head Free	West	DEEP PINK
Head Free	Up	BLUE
Head Free	Down	ORANGE
Horizon	North	BLUE
Horizon	East	DARK VIOLET
Horizon	South	YELLOW
Horizon	West	DEEP PINK
Horizon	Up	BLUE
Horizon	Down	ORANGE
Angle	North	CYAN
Angle	East	DARK VIOLET
Angle	South	YELLOW
Angle	West	DEEP PINK
Angle	Up	BLUE
Angle	Down	ORANGE
Mag	North	MINT GREEN
Mag	East	DARK VIOLET
Mag	South	ORANGE
Mag	West	DEEP PINK
Mag	Up	BLUE
Mag	Down	ORANGE
Baro	North	LIGHT BLUE
Baro	East	DARK VIOLET
Baro	South	RED
Baro	West	DEEP PINK
Baro	Up	BLUE
Baro	Down	ORANGE

Armed state

This mode toggles LEDs between green and blue when disarmed and armed, respectively.

Note: Armed State cannot be used with Flight Mode.

Thrust state

This mode fades the LED current LED color to the previous/next color in the HSB color space depending on throttle stick position. When the throttle is in the middle position the color is unaffected, thus it can be mixed with orientation colors to indicate orientation and throttle at the same time. Thrust should normally be combined with Color or Mode/Orientation.

Thrust ring state

This mode is allows you to use a 12, 16 or 24 leds ring (e.g. NeoPixel ring) for an afterburner effect. When armed the leds use the following sequences: 2 On, 4 Off, 2 On, 4 Off, and so on. The light pattern rotates clockwise as throttle increases.

A better effect is acheived when LEDs configured for thrust ring have no other functions.

LED direction and X/Y positions are irrelevant for thrust ring LED state. The order of the LEDs that have the state determines how the LED behaves.

Each LED of the ring can be a different color. The color can be selected between the 16 colors availables.

For example, led 0 is set as a `R`ing thrust state led in color 13 as follow.

```
led 0 2,2::R:13
```

LED strips and rings can be combined.

Solid Color

The mode allows you to set an LED to be permanently on and set to a specific color.

x,y position and directions are ignored when using this mode.

Other modes will override or combine with the color mode.

For example, to set led 0 to always use color 10 you would issue this command.

```
led 0 0,0::C:10
```

Colors

Colors can be configured using the cli `color` command.

The `color` command takes either zero or two arguments - an zero-based color number and a sequence which indicates pair of hue, saturation and value (HSV).

See http://en.wikipedia.org/wiki/HSL_and_HSV

If used with zero arguments it prints out the color configuration which can be copied for future reference.

The default color configuration is as follows:

Index	Color
0	black
1	white
2	red
3	orange
4	yellow
5	lime green
6	green
7	mint green
8	cyan
9	light blue
10	blue
11	dark violet
12	magenta
13	deep pink
14	black
15	black

```
color 0 0,0,0
color 1 0,255,255
color 2 0,0,255
color 3 30,0,255
color 4 60,0,255
color 5 90,0,255
color 6 120,0,255
color 7 150,0,255
color 8 180,0,255
color 9 210,0,255
color 10 240,0,255
color 11 270,0,255
color 12 300,0,255
color 13 330,0,255
color 14 0,0,0
color 15 0,0,0
```

Positioning

Cut the strip into sections as per diagrams below. When the strips are cut ensure you reconnect each output to each input with cable where the break is made. e.g. connect 5V out to 5V in, GND to GND and Data Out to Data In.

Orientation is when viewed with the front of the aircraft facing away from you and viewed from above.

Example 12 LED config

The default configuration is as follows

```
led 0 15,15:ES:IA:0
led 1 15,8:E:WF:0
led 2 15,7:E:WF:0
led 3 15,0:NE:IA:0
led 4 8,0:N:F:0
led 5 7,0:N:F:0
led 6 0,0:NW:IA:0
led 7 0,7:W:WF:0
led 8 0,8:W:WF:0
led 9 0,15:SW:IA:0
led 10 7,15:S:WF:0
led 11 8,15:S:WF:0
led 12 7,7:U:WF:0
led 13 8,7:U:WF:0
led 14 7,8:D:WF:0
led 15 8,8:D:WF:0
led 16 8,9::R:3
led 17 9,10::R:3
led 18 10,11::R:3
led 19 10,12::R:3
led 20 9,13::R:3
led 21 8,14::R:3
led 22 7,14::R:3
led 23 6,13::R:3
led 24 5,12::R:3
led 25 5,11::R:3
led 26 6,10::R:3
led 27 7,9::R:3
led 28 0,0:::0
led 29 0,0:::0
led 30 0,0:::0
led 31 0,0:::0
```

Which translates into the following positions:

```
     6             3
      \           /
       \   5-4   /
        \ FRONT /
    7,8 | 12-15 | 1,2
        /  BACK \
       /  10,11  \
      /           \
     9             0
       RING 16-27
```

LEDs 0,3,6 and 9 should be placed underneath the quad, facing downwards. LEDs 1-2, 4-5, 7-8 and 10-11 should be positioned so the face east/north/west/south, respectively. LEDs 12-13 should be placed facing down, in the middle LEDs 14-15 should be placed facing up, in the middle LEDs 16-17 should be placed in a ring and positioned at the rear facing south.

This is the default so that if you don't want to place LEDs top and bottom in the middle just connect the first 12 LEDs.

Example 16 LED config

```
led 0 15,15:SD:IA:0
led 1 8,8:E:FW:0
led 2 8,7:E:FW:0
led 3 15,0:ND:IA:0
led 4 7,7:N:FW:0
led 5 8,7:N:FW:0
led 6 0,0:ND:IA:0
led 7 7,7:W:FW:0
led 8 7,8:W:FW:0
led 9 0,15:SD:IA:0
led 10 7,8:S:FW:0
led 11 8,8:S:FW:0
led 12 7,7:D:FW:0
led 13 8,7:D:FW:0
led 14 7,7:U:FW:0
led 15 8,7:U:FW:0
```

Which translates into the following positions:

```
     6             3
      \           /
       \   5-4   /
      7 \ FRONT / 2
        | 12-15 |
      8 /  BACK \ 1
       /  10-11  \
      /           \
     9             0
```

LEDs 0,3,6 and 9 should be placed underneath the quad, facing downwards. LEDs 1-2, 4-5, 7-8 and 10-11 should be positioned so the face east/north/west/south, respectively. LEDs 12-13 should be placed facing down, in the middle LEDs 14-15 should be placed facing up, in the middle

Exmple 28 LED config

```
#right rear cluster
led 0 9,9:S:FWT:0
led 1 10,10:S:FWT:0
led 2 11,11:S:IA:0
led 3 11,11:E:IA:0
led 4 10,10:E:AT:0
led 5 9,9:E:AT:0
# right front cluster
led 6 10,5:S:F:0
led 7 11,4:S:F:0
led 8 12,3:S:IA:0
led 9 12,2:N:IA:0
led 10 11,1:N:F:0
led 11 10,0:N:F:0
# center front cluster
led 12 7,0:N:FW:0
led 13 6,0:N:FW:0
led 14 5,0:N:FW:0
led 15 4,0:N:FW:0
# left front cluster
led 16 2,0:N:F:0
led 17 1,1:N:F:0
led 18 0,2:N:IA:0
led 19 0,3:W:IA:0
led 20 1,4:S:F:0
led 21 2,5:S:F:0
# left rear cluster
led 22 2,9:W:AT:0
led 23 1,10:W:AT:0
led 24 0,11:W:IA:0
led 25 0,11:S:IA:0
led 26 1,10:S:FWT:0
led 27 2,9:S:FWT:0
```

```
       16-18  9-11
19-21 \           / 6-8
       \  12-15  /
        \ FRONT /
        /  BACK \
       /         \
22-24 /           \ 3-5
       25-27   0-2
```

All LEDs should face outwards from the chassis in this configuration.

Note: This configuration is specifically designed for the Alien Spider AQ50D PRO 250mm frame.

Troubleshooting

On initial power up the LEDs on the strip will be set to WHITE. This means you can attach a current meter to verify the current draw if your measurement equipment is fast enough. Most 5050 LEDs will draw 0.3 Watts a piece. This also means that you can make sure that each R,G and B LED in each LED module on the strip is also functioning.

After a short delay the LEDs will show the unarmed color sequence and or low-battery warning sequence.

Also check that the feature `LED_STRIP` was correctly enabled and that it does not conflict with other features, as above.

Display

Cleanflight supports displays to provide information to you about your aircraft and cleanflight state.

When the aircraft is armed the display does not update so flight is not affected. When disarmed the display cycles between various pages.

There is currently no way to change the information on the pages, the list of pages or the time between pages - Code submissions via pull-requests are welcomed!

Supported Hardware

At this time no other displays are supported other than the SSD1306 / UG-2864HSWEG01.

Configuration

From the CLI enable the `DISPLAY` feature

```
feature DISPLAY
```

SSD1306 OLED displays

The SSD1306 display is a 128x64 OLED display that is visible in full sunlight, small and consumes very little current. This makes it ideal for aircraft use.

There are various models of SSD1306 boards out there, they are not all equal and some require addtional modifications before they work. Choose wisely!

Links to displays:

- banggood.com 0.96 Inch 4Pin White IIC I2C OLED Display Module 12864 LED For Arduino
- banggood.com 0.96 Inch 4Pin IIC I2C Blue OLED Display Module For Arduino
- wide.hk I2C 0.96" OLED display module
- witespyquad.gostorego.com ReadyToFlyQuads 1" OLED Display
- multiwiicopter.com PARIS 1" OLED 128x64 PID tuning screen AIR

The banggood.com display is the cheapest at the time fo writing and will correctly send I2C ACK signals.

Crius CO-16

This display is best avoided but will work if you modify it.

Step 1

As supplied the I2C ack signal is not sent because the manufacturer did not bridge D1 and D2 together. To fix this solder the two pins together as they enter the screen. Failure to do this will result is a screen that doesn't display anything.

Step 2

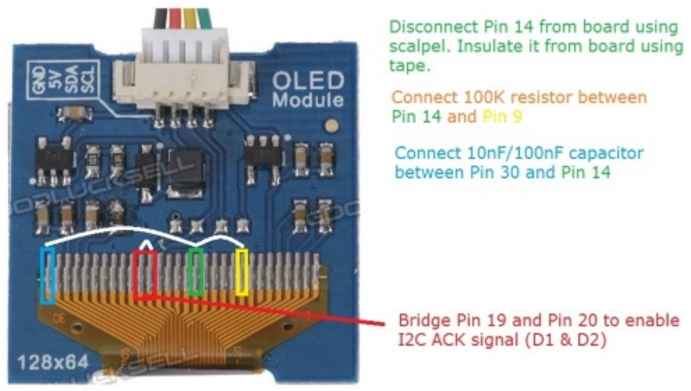
Pin 14 must be disconnected from the main board using a scalpel. Then connect a 10nF or 100nF capacitor between pins 30 and the lifted pin 14.

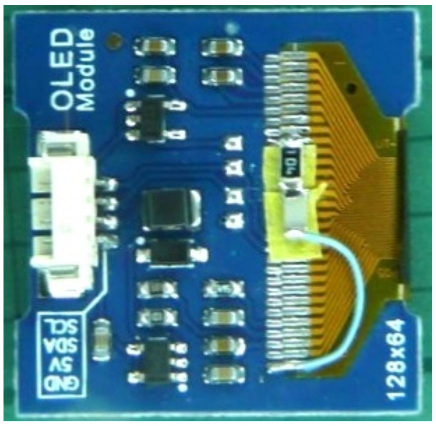
Step 3

Connect a 100K resistor between Pin 9 and the lifted Pin 14.

Failure to perform steps 2 and 3 will result in a display that only works on power up some of the time any may display random dots or other display corruption.

More can be read about this procedure here: http://www.multiwii.com/forum/viewtopic.php?f=6&t=2705&start=10


Disconnect Pin 14 from board using scalpel. Insulate it from board using tape.
Connect 100K resistor between Pin 14 and Pin 9
Connect 10nF/100nF capacitor between Pin 30 and Pin 14
Bridge Pin 19 and Pin 20 to enable I2C ACK signal (D1 & D2)
OLED Module
GND 5V SDA SCL
128x64



Connections

Connect +5v, Ground, I2C SDA and I2C SCL from the flight controller to the display.

On Naze32 rev 5 boards the SDA and SCL pads are underneath the board.

Buzzer

Cleanflight supports a buzzer which is used for the following purposes:

- Low and critical battery alarms (when battery monitoring enabled)
- Arm/disarm tones (and warning beeps while armed)
- Notification of calibration complete status
- TX-AUX operated beeping - useful for locating your aircraft after a crash
- Failsafe status
- Flight mode change
- Rate profile change (via TX-AUX switch)

If the arm/disarm is via the control stick, holding the stick in the disarm position will sound a repeating tone. This can be used as a lost-model locator.

Three beeps immediately after powering the board means that the gyroscope calibration has completed successfully. Cleanflight calibrates the gyro automatically upon every power-up. It is important that the copter stay still on the ground until the three beeps sound, so that gyro calibration isn't thrown off. If you move the copter significantly during calibration, Cleanflight will detect this, and will automatically re-start the calibration once the copter is still again. This will delay the "three beeps" tone. If you move the copter just a little bit, the gyro calibration may be incorrect, and the copter may not fly correctly. In this case, the gyro calibration can be performed manually via stick command, or you may simply power cycle the board.

There is a special arming tone used if a GPS fix has been attained, and there's a "ready" tone sounded after a GPS fix has been attained (only happens once). The tone sounded via the TX-AUX-switch will count out the number of satellites (if GPS fix).

The CLI command `play_sound` is useful for demonstrating the buzzer tones. Repeatedly entering the command will play the various tones in turn. Entering the command with a numeric-index parameter (see below) will play the associated tone.

Buzzer is enabled by default on platforms that have buzzer connections.

Tone sequences

Buzzer tone sequences (square wave generation) are made so that : 1st, 3rd, 5th, .. are the delays how long the beeper is on and 2nd, 4th, 6th, .. are the delays how long beeper is off. Delays are in milliseconds/10 (i.e., 5 => 50ms).

Sequences available in Cleanflight v1.9 and above are :

```
0    GYRO_CALIBRATED       20, 10, 20, 10, 20, 10   Gyro is calibrated
1    RX_LOST_LANDING       10, 10, 10, 10, 10, 40, 40, 10, 40, 10, 40, 40, 10, 10, 10, 10, 10, 70    SOS morse code
2    RX_LOST               50, 50       TX off or signal lost (repeats until TX is okay)
3    DISARMING             15, 5, 15, 5     Disarming the board
4    ARMING                30, 5, 5, 5      Arming the board
5    ARMING_GPS_FIX        5, 5, 15, 5, 5, 5, 15, 30    Arming and GPS has fix
6    BAT_CRIT_LOW          50, 2        Battery is critically low (repeats)
7    BAT_LOW               25, 50       Battery is getting low (repeats)
8    NULL                  multi beeps      GPS status (sat count)
9    RX_SET                10, 10       RX is set (when aux channel is set for beep or beep sequence how many satellites has found if GPS enabled)
10   ACC_CALIBRATION       5, 5, 5, 5       ACC inflight calibration completed
11   ACC_CALIBRATION_FAIL  20, 15, 35, 5    ACC inflight calibration failed
12   READY_BEEP            4, 5, 4, 5, 8, 5, 15, 5, 8, 5, 4, 5, 4, 5    GPS locked and copter ready
13   NULL                  multi beeps      Variable # of beeps (confirmation, GPS sat count, etc)
14   DISARM_REPEAT         0, 100, 10       Stick held in disarm position (after pause)
15   ARMED                 0, 245, 10, 5    Board is armed (after pause ; repeats until board is disarmed or throttle is increased)
```

Types of buzzer supported

The buzzers are enabled/disabled by simply enabling or disabling a GPIO output pin on the board. This means the buzzer must be able to generate its own tone simply by having power applied to it.

Buzzers that need an analog or PWM signal do not work and will make clicking noises or no sound at all.

Examples of a known-working buzzers.

- Hcm1205x Miniature Buzzer 5v
- 5V Electromagnetic Active Buzzer Continuous Beep
- Radio Shack Model: 273-074 PC-BOARD 12VDC (3-16v) 70DB PIEZO BUZZER
- MultiComp MCKPX-G1205A-3700 TRANSDUCER, THRU-HOLE, 4V, 30MA
- 3-24V Piezo Electronic Tone Buzzer Alarm 95DB

Connections

Naze32

Connect a supported buzzer directly to the BUZZ pins. Observe polarity. Also if you are working with flight controller outside of a craft, on a bench for example, you need to supply 5 volts and ground to one of the ESC connections or the buzzer will not function.

CC3D

Buzzer support on the CC3D requires that a buzzer circuit be created to which the input is PA15. PA15 is unused and not connected according to the CC3D Revision A schematic. Connecting to PA15 requires careful soldering.

See the CC3D - buzzer circuit.pdf for details.

Sonar

A sonar sensor can be used to measure altitude for use with BARO and SONAR altitude hold modes.

The sonar sensor is used instead of the pressure sensor (barometer) at low altitudes. The sonar sensor is only used when the aircraft inclination angle (attitude) is small.

Hardware

Currently the only supported sensor is the HCSR04 sensor.

Connections

Naze/Flip32+

Mode	Trigger	Echo	Inline 1k resistors
Parallel PWM/ADC current sensor	PB8 / Motor 5	PB9 / Motor 6	NO (5v tolerant)
PPM/Serial RX	PB0 / RC7	PB1 / RC8	YES (3.3v input)

Constraints

Current meter cannot be used in conjunction with Parallel PWM and Sonar.

Olimexino

Trigger	Echo	Inline 1k resistors
PB0 / RC7	PB1 / RC8	YES (3.3v input)

Constraints

Current meter cannot be used in conjunction with Sonar.

CC3D

Trigger	Echo	Inline 1k resistors
PB5	PB0	YES (3.3v input)

Constraints

Sonar cannot be used in conjuction with SoftSerial or Parallel PWM.

Profiles

A profile is a set of configuration settings.

Currently three profiles are supported. The default profile is profile `0`.

Changing profiles

Profiles can be selected using a GUI or the following stick combinations:

Profile	Throttle	Yaw	Pitch	Roll
0	Down	Left	Middle	Left
1	Down	Left	Up	Middle
2	Down	Left	Middle	Right

The CLI `profile` command can also be used:

```
profile <index>
```

Rate Profiles

Cleanflight supports rate profiles in addition to regular profiles.

Rate profiles contain settings that adjust how your craft behaves based on control input.

Three rate profiles are supported.

Rate profiles can be selected while flying using the inflight adjustments feature.

Each normal profile has a setting called 'default*rate*profile`. When a profile is activated the corresponding rate profile is also activated.

Profile 0 has a default rate profile of 0. Profile 1 has a default rate profile of 1. Profile 2 has a default rate profile of 2.

The defaults are set this way so that it's simple to configure a profile and a rate profile at the same.

The current rate profile can be shown or set using the CLI `rateprofile` command:

```
rateprofile <index>
```

The values contained within a rate profile can be seen by using the CLI `dump rates` command.

e.g

```
# dump rates

# rateprofile
rateprofile 0

set rc_rate = 90
set rc_expo = 65
set thr_mid = 50
set thr_expo = 0
set roll_pitch_rate = 0
set yaw_rate = 0
set tpa_rate = 0
set tpa_breakpoint = 1500
```

Modes

Cleanflight has various modes that can be toggled on or off. Modes can be enabled/disabled by stick positions, auxillary receiver channels and other events such as failsafe detection.

MSP ID	CLI ID	Short Name	Function
0	0	ARM	Enables motors and flight stabilisation
1	1	ANGLE	Legacy auto-level flight mode
2	2	HORIZON	Auto-level flight mode
3	3	BARO	Altitude hold mode (Requires barometer sensor)
5	4	MAG	Heading lock
6	5	HEADFREE	Head Free - When enabled yaw has no effect on pitch/roll inputs
7	6	HEADADJ	Heading Adjust - Sets a new yaw origin for HEADFREE mode
8	7	CAMSTAB	Camera Stabilisation
9	8	CAMTRIG	
10	9	GPSHOME	Autonomous flight to HOME position
11	10	GPSHOLD	Maintain the same longitude/lattitude
12	11	PASSTHRU	Pass roll, yaw, and pitch directly from rx to servos in airplane mix
13	12	BEEPERON	Enable beeping - useful for locating a crashed aircraft
14	13	LEDMAX	
15	14	LEDLOW	
16	15	LLIGHTS	
17	16	CALIB	
18	17	GOV	
19	18	OSD	Enable/Disable On-Screen-Display (OSD)
20	19	TELEMETRY	Enable telemetry via switch
22	21	SONAR	Altitude hold mode (sonar sensor only)
26	25	BLACKBOX	Enable BlackBox logging
27	26	GTUNE	G-Tune - auto tuning of Pitch/Roll/Yaw P values

Mode details

Headfree

In this mode, the "head" of the multicopter is always pointing to the same direction as when the feature was activated. This means that when the multicopter rotates around the Z axis (yaw), the controls will always respond according the same "head" direction.

With this mode it is easier to control the multicopter, even fly it with the physical head towards you since the controls always respond the same. This is a friendly mode to new users of multicopters and can prevent losing the control when you don't know the head direction.

GPS Return To Home

WORK-IN-PROGRESS. This mode is not reliable yet, please share your experiences with the developers.

In this mode the aircraft attempts to return to the GPS position recorded when the aircraft was armed.

This mode should be enabled in conjunction with Angle or Horizon modes and an Altitude hold mode.

Requires a 3D GPS fix and minimum of 5 satellites in view.

GPS Position Hold

WORK-IN-PROGRESS. This mode is not reliable yet, please share your experiences with the developers.

In this mode the aircraft attempts to stay at the same GPS position, as recorded when the mode is enabled.

Disabling and re-enabling the mode will reset the GPS hold position.

This mode should be enabled in conjunction with Angle or Horizon modes and an Altitude hold mode.

Requires a 3D GPS fix and minimum of 5 satellites in view.

Auxillary Configuration

Spare auxillary receiver channels can be used to enable/disable modes. Some modes can only be enabled this way.

Configure your transmitter so that switches or dials (potentiometers) send channel data on channels 5 and upwards (the first 4 channels are usually occupied by the throttle, aileron, rudder, and elevator channels).

e.g. You can configure a 3 position switch to send 1000 when the switch is low, 1500 when the switch is in the middle and 2000 when the switch is high.

Configure your tx/rx channel limits to use values between 1000 and 2000. The range used by mode ranges is fixed to 900 to 2100.

When a channel is within a specifed range the corresponding mode is enabled.

Use the GUI configuration tool to allow easy configuration when channel.

CLI

There is a CLI command, `aux` that allows auxillary configuration. It takes 5 arguments as follows:

- AUD range slot number (0 - 39)
- mode id (see mode list above)
- AUX channel index (AUX1 = 0, AUX2 = 1,... etc)
- low position, from 900 to 2100. Should be a multiple of 25.
- high position, from 900 to 2100. Should be a multiple of 25.

If the low and high position are the same then the values are ignored.

e.g.

Configure AUX range slot 0 to enable ARM when AUX1 is withing 1700 and 2100.

```
aux 0 0 0 1700 2100
```

You can display the AUX configuration by using the `aux` command with no arguments.

In-flight Adjustments

With Cleanflight it's possible to make adjustments to various settings by using AUX channels from your transmitter while the aircraft is flying.

Warning

Changing settings during flight can make your aircraft unstable and crash if you are not careful.

Recommendations

- Always make adjustments while flying in a large open area.
- Make small adjustments and fly carefully to test your adjustment.
- Give yourself enough flying space and time to adjust to how your changes affect the behaviour of the aircraft.
- Remember to set adjustment channel switches/pots to the center position before powering on your TX and your aircraft.
- If possible configure switch warnings on your transitter for dedicated adjustment switches.
- A momentary 3 position switch is the best choice of switch for this - i.e. one that re-centers itself when you let go of it.

Overview

Up to 4 RX channels can be used to make different adjustments at the same time.

The adjustment a channel makes can be controlled by another channel.

The following adjustments can be made, in flight, as well as on the ground.

- RC Rate
- RC Expo
- Throttle Expo
- Roll & Pitch Rate
- Yaw Rate
- Pitch+Roll P I and D

- Yaw P I and D

Example scenarios: Up to 4 3-position switches or pots can be used to adjust 4 different settings at the same time. A single 2/3/4/5/6/x position switch can be used to make one 3 position switch adjust one setting at a time.

Any combination of switches and pots can be used. So you could have 6 POS switch.

Settings are not saved automatically, connect a GUI, refresh and save or save using stick position when disarmed. Powering off without saving will discard the adjustments.

Settings can be saved when disarmed using stick positions: Throttle Low, Yaw Left, Pitch Low, Roll Right.

Adjustment switches

The switch can be a ON-OFF-ON, POT or momentary ON-OFF-ON switch. The latter is recommended.

When the switch is returned to the center position the value will not be increased/decreased.

Each time you can press the switch high/low and then return it to the middle the value will change at least once, you do not have to wait before pressing the switch again if you want to increase/decrease at a faster rate. While the adjustment switch held is high/low, the adjustment function applies and increases/decreases the value being adjusted twice a second and the flight controller will beep shorter/longer, respectively. The system works similar to how a keyboard repeat delay works.

Hint: With OpenTX transmitters you can combine two momentary OFF-ON switches to control a single channel. You could make it so that a momentary switch on the left of your transmitter decreases the value and a momentary switch on the right increases the value. Experiment with your mixer!

Configuration

The CLI command `adjrange` is used to configure adjustment ranges.

12 adjustment ranges can be defined. 4 adjustments can be made at the same time, each simultaneous adjustment requires an adjustment slot.

Show the current ranges using:

```
adjrange
```

Configure a range using:

```
adjrange <index> <slot> <range channel> <range start> <range end> <adjustment function>
<adjustment channel>
```

Argument	Value	Meaning
Index	0 - 11	Select the adjustment range to configure
Slot	0 - 3	Select the adjustment slot to use
Range Channel	0 based index, AUX1 = 0, AUX2 = 1	The AUX channel to use to select an adjustment for a switch/pot
Range Start	900 - 2100. Steps of 25, e.g. 900, 925, 950...	Start of range
Range End	900 - 2100	End of range
Adjustment function	0 - 11	See Adjustment function table
Adjustment channel	0 based index, AUX1 = 0, AUX2 = 1	The channel that is controlled by a 3 Position switch/Pot

Range Start/End values should match the values sent by your receiver.

Normally Range Channel and Slot values are grouped together over multiple adjustment ranges.

The Range Channel and the Adjustment Channel can be the same channel. This is useful when you want a single 3 Position switch to be dedicated to a single adjustment function regardless of other switch positions.

The adjustment function is applied to the adjustment channel when range channel is between the range values. The adjustment is made when the adjustment channel is in the high or low position. high = mid*rc + 200, low = mid* rc - 200. by default this is 1700 and 1300 respectively.

When the Range Channel does not fall into Start/End range the assigned slot will retain it's state and will continue to apply the adjustment. For this reason ensure that you define enough ranges to cover the range channel's usable range.

Adjustment function

Value	Adjustment	Notes
0	None	
1	RC RATE	
2	RC_EXPO	
3	THROTTLE_EXPO	
4	PITCH*ROLL*RATE	
5	YAW_RATE	
6	PITCH*ROLL*P	
7	PITCH*ROLL*I	
8	PITCH*ROLL*D	
9	YAW_P	
10	YAW_I	
11	YAW_D	
12	RATE_PROFILE	Switch between 3 rate profiles using a 3 position switch.
13	PITCH_RATE	
14	ROLL_RATE	

15	PITCH_P	
16	PITCH_I	
17	PITCH_D	
18	ROLL_P	
19	ROLL_I	
20	ROLL_D	

Examples

Example 1 - 3 Position switch used to adjust pitch/roll rate

```
adjrange 0 0 3 900 2100 4 3
```

explained:

- configure adjrange 0 to use adjustment slot 1 (0) so that when aux4 (3) in the range 900-2100 then use adjustment 4 (pitch/roll rate) when aux 4 (3) is in the appropriate position.

Example 2 - 2 Position switch used to enable adjustment of RC rate via a 3 position switch

```
adjrange 1 1 0 900 1700 0 2
adjrange 2 1 0 1700 2100 1 2
```

explained:

- configure adjrange 1 to use adjustment slot 2 (1) so that when aux1 (0) in the range 900-1700 then do nothing (0) when aux 3 (2) is in any position.
- configure adjrange 2 to use adjustment slot 2 (1) so that when aux1 (0) in the range 1700-2100 then use adjustment rc rate (1) when aux 3 (2) is in the appropriate position.

Without the entire range of aux1 being defined there is nothing that would stop aux 3 adjusting the pitch/roll rate once aux 1 wasn't in the higher range.

Example 3 - 6 Position switch used to select PID tuning adjustments via a 3 position switch

```
adjrange 3 2 1 900 1150 6 3
adjrange 4 2 1 1150 1300 7 3
adjrange 5 2 1 1300 1500 8 3
adjrange 6 2 1 1500 1700 9 3
adjrange 7 2 1 1700 1850 10 3
adjrange 8 2 1 1850 2100 11 3
```

explained:

- configure adjrange 3 to use adjustment slot 3 (2) so that when aux2 (1) in the range 900-1150 then use adjustment Pitch/Roll P (6) when aux 4 (3) is in the appropriate position.
- configure adjrange 4 to use adjustment slot 3 (2) so that when aux2 (1) in the range 1150-1300 then use adjustment Pitch/Roll I (7) when aux 4 (3) is in the appropriate position.
- configure adjrange 5 to use adjustment slot 3 (2) so that when aux2 (1) in the range 1300-1500 then use adjustment Pitch/Roll D (8) when aux 4 (3) is in the appropriate position.
- configure adjrange 6 to use adjustment slot 3 (2) so that when aux2 (1) in the range 1500-1700 then use adjustment Yaw P (9) when aux 4 (3) is in the appropriate position.
- configure adjrange 7 to use adjustment slot 3 (2) so that when aux2 (1) in the range 1700-1850 then use adjustment Yaw I (10) when aux 4 (3) is in the appropriate position.
- configure adjrange 8 to use adjustment slot 3 (2) so that when aux2 (1) in the range 1850-2100 then use adjustment Yaw D (11) when aux 4 (3) is in the appropriate position.

Example 4 - Use a single 3 position switch to change between 3 different rate profiles

adjrange 11 3 3 900 2100 12 3

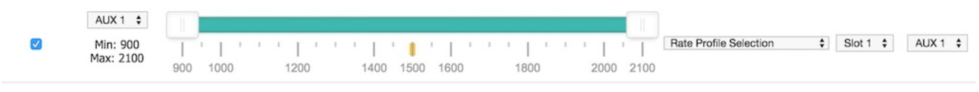
explained:

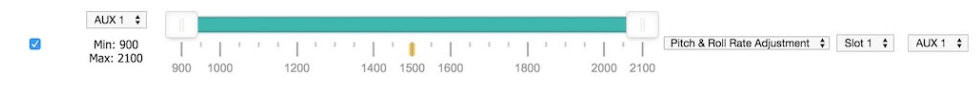
- configure adjrange 11 to use adjustment slot 4 (3) so that when aux4 (3) in the range 900-2100 then use adjustment Rate Profile (12) when aux 4 (3) is in the appropriate position.

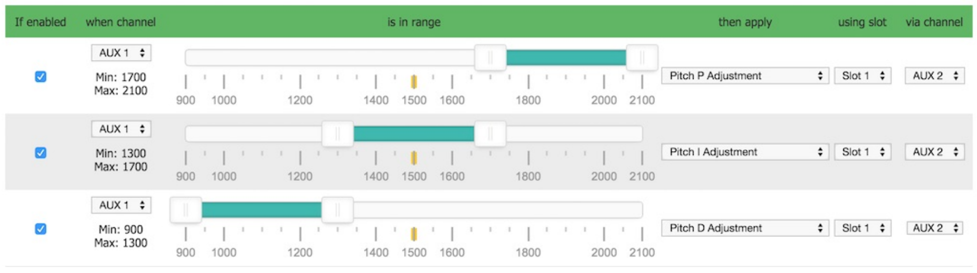
When the switch is low, rate profile 0 is selcted. When the switch is medium, rate profile 1 is selcted. When the switch is high, rate profile 2 is selcted.

Configurator examples

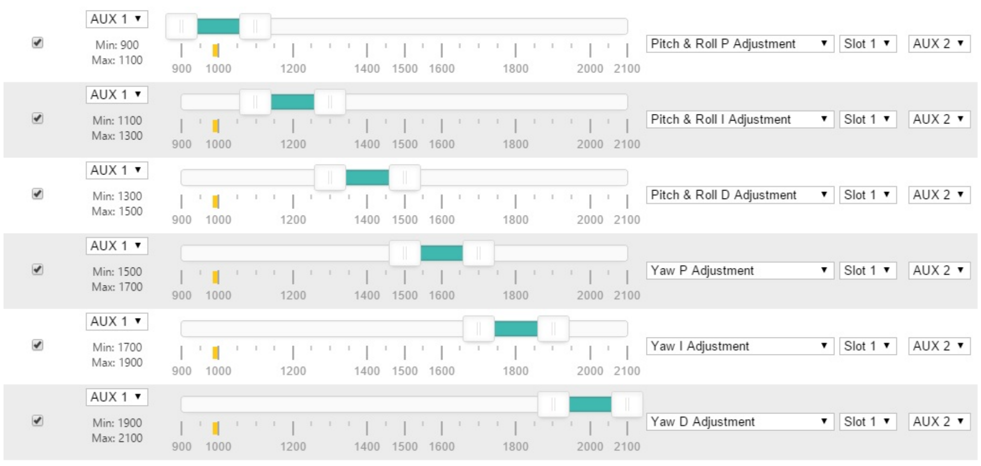
The following 5 images show valid configurations. In all cales the enture usable range for the Range Channel is used.


AUX 1, Min: 900, Max: 2100, Rate Profile Selection, Slot 1, AUX 1

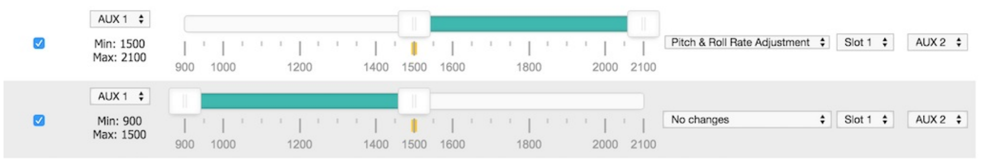

AUX 1, Min: 900, Max: 2100, Pitch & Roll Rate Adjustment, Slot 1, AUX 1



If enabled	when channel	is in range	then apply	using slot	via channel
✓	AUX 1	Min: 1700 Max: 2100	Pitch P Adjustment	Slot 1	AUX 2
✓	AUX 1	Min: 1300 Max: 1700	Pitch I Adjustment	Slot 1	AUX 2
✓	AUX 1	Min: 900 Max: 1300	Pitch D Adjustment	Slot 1	AUX 2

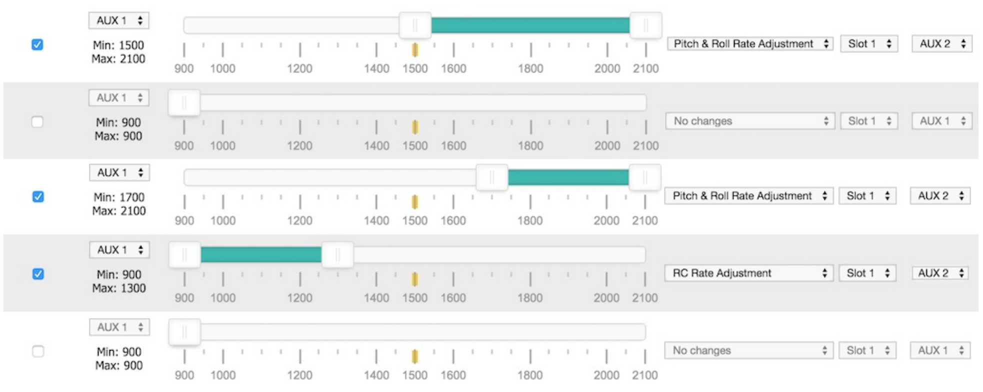


✓	AUX 1	Min: 900 Max: 1100	Pitch & Roll P Adjustment	Slot 1	AUX 2
✓	AUX 1	Min: 1100 Max: 1300	Pitch & Roll I Adjustment	Slot 1	AUX 2
✓	AUX 1	Min: 1300 Max: 1500	Pitch & Roll D Adjustment	Slot 1	AUX 2
✓	AUX 1	Min: 1500 Max: 1700	Yaw P Adjustment	Slot 1	AUX 2
✓	AUX 1	Min: 1700 Max: 1900	Yaw I Adjustment	Slot 1	AUX 2
✓	AUX 1	Min: 1900 Max: 2100	Yaw D Adjustment	Slot 1	AUX 2



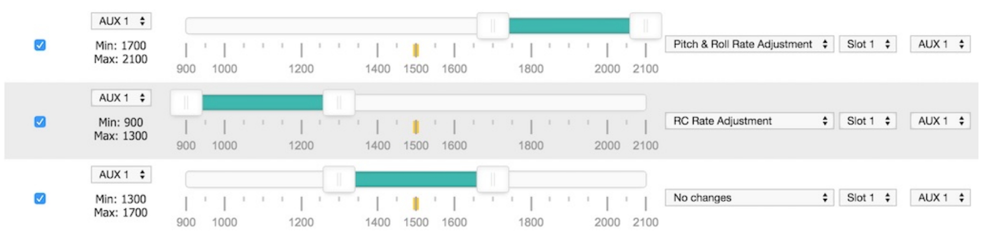
✓	AUX 1	Min: 1500 Max: 2100	Pitch & Roll Rate Adjustment	Slot 1	AUX 2
✓	AUX 1	Min: 900 Max: 1500	No changes	Slot 1	AUX 2

The following examples shows **incorrect** configurations - the entire usable range for the Range Channel is not used in both cases.



✓	AUX 1	Min: 1500 Max: 2100	Pitch & Roll Rate Adjustment	Slot 1	AUX 2
	AUX 1	Min: 900 Max: 900	No changes	Slot 1	AUX 1
✓	AUX 1	Min: 1700 Max: 2100	Pitch & Roll Rate Adjustment	Slot 1	AUX 2
✓	AUX 1	Min: 900 Max: 1300	RC Rate Adjustment	Slot 1	AUX 2
	AUX 1	Min: 900 Max: 900	No changes	Slot 1	AUX 1

In the following example, the incorrect configuraton (above) has been corrected by adding a range that makes 'No changes'.



✓	AUX 1	Min: 1700 Max: 2100	Pitch & Roll Rate Adjustment	Slot 1	AUX 1
✓	AUX 1	Min: 900 Max: 1300	RC Rate Adjustment	Slot 1	AUX 1
✓	AUX 1	Min: 1300 Max: 1700	No changes	Slot 1	AUX 1

Controls

Arming

When armed, the aircraft is ready to fly and the motors will spin when throttle is applied. The motors will spin at a slow speed when armed (this feature may be disabled by setting MOTOR_STOP, but for safety reasons, that is not recommended).

By default, arming and disarming is done using stick positions. (NOTE: this feature is disabled when using a switch to arm.)

Stick Positions

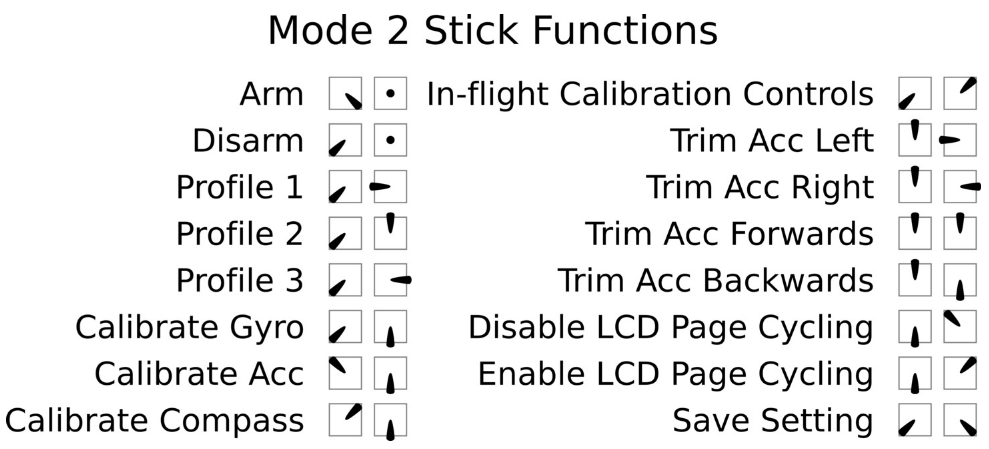
The three stick positions are:

Position	Approx. Channel Input
LOW	1000

CENTER	1500
HIGH	2000

The stick positions are combined to activate different functions:

Function	Throttle	Yaw	Pitch	Roll
ARM	LOW	HIGH	CENTER	CENTER
DISARM	LOW	LOW	CENTER	CENTER
Profile 1	LOW	LOW	CENTER	LOW
Profile 2	LOW	LOW	HIGH	CENTER
Profile 3	LOW	LOW	CENTER	HIGH
Calibrate Gyro	LOW	LOW	LOW	CENTER
Calibrate Acc	HIGH	LOW	LOW	CENTER
Calibrate Mag/Compass	HIGH	HIGH	LOW	CENTER
Inflight calibration controls	LOW	LOW	HIGH	HIGH
Trim Acc Left	HIGH	CENTER	CENTER	LOW
Trim Acc Right	HIGH	CENTER	CENTER	HIGH
Trim Acc Forwards	HIGH	CENTER	HIGH	CENTER
Trim Acc Backwards	HIGH	CENTER	LOW	CENTER
Disable LCD Page Cycling	LOW	CENTER	HIGH	LOW
Enable LCD Page Cycling	LOW	CENTER	HIGH	HIGH
Save setting	LOW	LOW	LOW	HIGH



Download a graphic cheat sheet with Tx stick commands (the latest version can always be found here).

Yaw control

While arming/disarming with sticks, your yaw stick will be moving to extreme values. In order to prevent your craft from trying to yaw during arming/disarming while on the ground, your yaw input will not cause the craft to yaw when the throttle is LOW (i.e. below the `min_check` setting).

For tricopters, you may want to retain the ability to yaw while on the ground, so that you can verify that your tail servo is working correctly before takeoff. You can do this by setting `tri_unarmed_servo` to `1` on the CLI (this is the default). If you are having issues with your tail rotor contacting the ground during arm/disarm, you can set this to `0` instead. Check this table to decide which setting will suit you:

Is yaw control of the tricopter allowed?				
	Disarmed		Armed	
	Throttle low	Throttle normal	Throttle low	Throttle normal
tri_unarmed_servo = 0	No	No	No	Yes
	No	No	No	Yes
tri_unarmed_servo = 1	Yes	Yes	Yes	Yes
	Yes	Yes	Yes	Yes

G-Tune instructions.

The algorithm has been originally developed by Mohammad Hefny (mohammad.hefny@gmail.com):

- http://technicaladventure.blogspot.com/2014/06/zero-pids-tuner-for-multirotors.html
- http://diydrones.com/profiles/blogs/zero-pid-tunes-for-multirotors-part-2
- http://www.multiwii.com/forum/viewtopic.php?f=8&t=5190

The G-Tune functionality for Cleanflight is ported from the Harakiri firmware.

Safety preamble: *Use at your own risk*

The implementation you have here is quiet different and just for adjusting the P values of ROLL/PITCH/YAW in Acro mode. When flying in Acro mode (yaw tune in other modes possible as well - see below) you can activate G-Tune with an AUX box (switch) while the copter is armed.

It will start tuning the wanted / possible axes (see below) in a predefined range (see below).

After activation you will probably notice nothing! That means G-Tune will not start shaking your copter, you will have to do it (or simply fly and let it work).

The G-Tune is based on the gyro error so it is only active when you give no RC input (that would be an additional error). So if you just roll only pitch and yaw are tuned. If you stop rolling G-Tune will wait ca. 450ms to let the axis settle and then start tuning that axis again. All axes are treated independently.

The easiest way to tune all axes at once is to do some air-jumps with the copter in Acro (RC centered and G-Tune activated... of course..).

You can set a too high P for the axes as default in the GUI, when the copter starts shaking the wobbles will be detected and P tuned down (be careful with the strength setting though - see below).

Yaw tune is disabled in any copter with less than 4 motors (like tricopters).

G-Tune in Horizon or Level mode will just affect Yaw axis (if more than 3 motors...)

You will see the results in the GUI - the tuning results will only be saved if you enable G-Tune mode while the copter is disarmed and G-Tune was used before when armed. You also can save the configuration in an alternative way (like hitting save button in the GUI, casting an eepromwrite with trimming, acc calibration etc.)

TPA and G-Tune: It is not tested and will most likely not result into something good. However G-Tune might be able to replace TPA for you.

Parameters and their function:

```
gtune_loP_rll        = 10  [0..200] Lower limit of ROLL P during G-Tune.  Note "10" means "1.0" in the GUI.
gtune_loP_ptch       = 10  [0..200] Lower limit of PITCH P during G-Tune. Note "10" means "1.0" in the GUI.
gtune_loP_yw         = 10  [0..200] Lower limit of YAW P during G-Tune.   Note "10" means "1.0" in the GUI.
gtune_hiP_rll        = 100 [0..200] Higher limit of ROLL P during G-Tune. 0 Disables tuning for that axis.  Note "100" means "10.0" in the GUI.
gtune_hiP_ptch       = 100 [0..200] Higher limit of PITCH P during G-Tune. 0 Disables tuning for that axis. Note "100" means "10.0" in the GUI.
gtune_hiP_yw         = 100 [0..200] Higher limit of YAW P during G-Tune. 0 Disables tuning for that axis.   Note "100" means "10.0" in the GUI.
gtune_pwr            = 0   [0..10] Strength of adjustment
gtune_settle_time    = 450 [200..1000] Settle time in ms
gtune_average_cycles = 16  [8..128] Number of looptime cycles used for gyro average calcullation
```

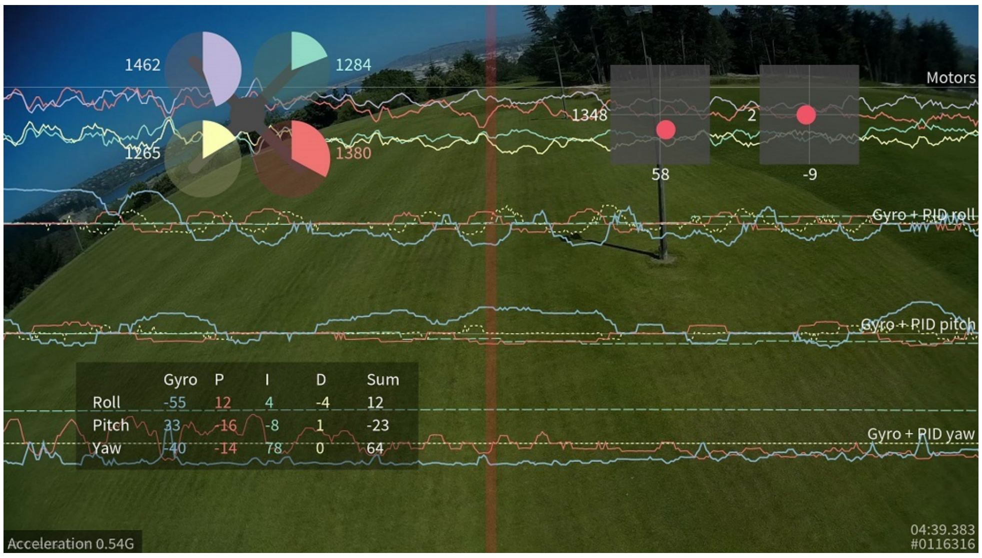
So you have lower and higher limits for each P for every axis. The preset range (GUI: 1.0 - 10.0) is quiet broad to represent most setups.

If you want tighter or more loose ranges change them here. gtune*loP*XXX can be configured lower than "10" that means a P of "1.0" in the GUI. So you can have "Zero P" but you may get sluggish initial control.

If you want to exclude one axis from the tuning you must set gtune*hiP*XXX to zero. Let's say you want to disable yaw tuning write in CLI `set gtune_hiP_yw = 0`. Note: The MultiWii Wiki advises you to trim the yaw axis on your transmitter. If you have done so (yaw not neutral on your RC) yaw tuning will be disabled.

You can adjust the strength of tuning by using `set gtune_pwr = N`. My small copter works fine with 0 and doesn't like a value of "3". My big copter likes "gtune*pwr = 5". It shifts the tuning to higher values and if too high can diminish the wobble blocking! So start with 0 (default). If you feel your resulting P is always too low for you, increase gtune*pwr. You will see it getting a little shaky if value is too high.

Blackbox flight data recorder



	Gyro	P	I	D	Sum
Roll	-55	12	4	-4	12
Pitch	33	-16	-8	1	-23
Yaw	40	-14	78	0	64

Introduction

This feature transmits your flight data information on every control loop iteration over a serial port to an external logging device to be recorded, or to a dataflash chip which is present on some flight controllers.

After your flight, you can view the resulting logs using the interactive log viewer:

https://github.com/cleanflight/blackbox-log-viewer

You can also use the `blackbox_decode` tool to turn the logs into CSV files for analysis, or render your flight log as a video using the `blackbox_render` tool. Those tools can be found in this repository:

https://github.com/cleanflight/blackbox-tools

Logged data

The blackbox records flight data on every iteration of the flight control loop. It records the current time in microseconds, P, I and

D corrections for each axis, your RC command stick positions (after applying expo curves), gyroscope data, accelerometer data (after your configured low-pass filtering), barometer and sonar readings, 3-axis magnetometer readings, raw VBAT and current measurements, RSSI, and the command being sent to each motor speed controller. This is all stored without any approximation or loss of precision, so even quite subtle problems should be detectable from the fight data log.

GPS data is logged whenever new GPS data is available. Although the CSV decoder will decode this data, the video renderer does not yet show any of the GPS information (this will be added later).

Supported configurations

The maximum data rate that can be recorded to the flight log is fairly restricted, so anything that increases the load can cause the flight log to drop frames and contain errors.

The Blackbox is typically used on tricopters and quadcopters. Although it will work on hexacopters and octocopters, because these craft have more motors to record, they must transmit more data to the flight log. This can increase the number of dropped frames. Although the browser-based log viewer supports hexacopters and octocopters, the command-line `blackbox_render` tool currently only supports tri- and quadcopters.

Cleanflight's `looptime` setting decides how frequently an update is saved to the flight log. The default looptime on Cleanflight is 3500. If you're using a looptime smaller than about 2400, you may experience some dropped frames due to the high required data rate. In that case you will need to reduce the sampling rate in the Blackbox settings, or increase your logger's baudrate to 250000. See the later section on configuring the Blackbox feature for details.

Setting up logging

First, you must enable the Blackbox feature. In the Cleanflight Configurator enter the Configuration tab, tick the "BLACKBOX" feature at the bottom of the page, and click "Save and reboot"

Now you must decide which device to store your flight logs on. You can either transmit the log data over a serial port to an external logging device like the OpenLog serial data logger to be recorded to a microSDHC card, or if you have a compatible flight controller you can store the logs on the onboard dataflash storage instead.

OpenLog serial data logger

The OpenLog is a small logging device which attaches to your flight controller using a serial port and logs your flights to a MicroSD card.

The OpenLog ships from SparkFun with standard "OpenLog 3" firmware installed. Although this original OpenLog firmware will work with the Blackbox, in order to reduce the number of dropped frames it should be reflashed with the higher performance OpenLog Blackbox firmware. The special Blackbox variant of the OpenLog firmware also ensures that the OpenLog is using Cleanflight compatible settings, and defaults to 115200 baud.

You can find the Blackbox version of the OpenLog firmware here, along with instructions for installing it onto your OpenLog.

microSDHC

Your choice of microSDHC card is very important to the performance of the system. The OpenLog relies on being able to make many small writes to the card with minimal delay, which not every card is good at. A faster SD-card speed rating is not a guarantee of better performance.

microSDHC cards known to have poor performance

- Generic 4GB Class 4 microSDHC card - the rate of missing frames is about 1%, and is concentrated around the most interesting parts of the log!
- Sandisk Ultra 32GB (unlike the smaller 16GB version, this version has poor write latency)

microSDHC cards known to have good performance

- Transcend 16GB Class 10 UHS-I microSDHC (typical error rate < 0.1%)
- Sandisk Extreme 16GB Class 10 UHS-I microSDHC (typical error rate < 0.1%)
- Sandisk Ultra 16GB (it performs only half as well as the Extreme in theory, but still very good)

You should format any card you use with the SD Association's special formatting tool , as it will give the OpenLog the best chance of writing at high speed. You must format it with either FAT, or with FAT32 (recommended).

Choosing a serial port for the OpenLog

First, tell the Blackbox to log using a serial port (rather than to an onboard dataflash chip). Go to the Configurator's CLI tab, enter `set blackbox_device=SERIAL` to switch logging to serial, and save.

You need to let Cleanflight know which of your serial ports you connect your OpenLog to (i.e. the Blackbox port), which you can do on the Configurator's Ports tab.

You should use a hardware serial port (such as UART1 on the Naze32, the two-pin Tx/Rx header in the center of the board). SoftSerial ports can be used for the Blackbox. However, because they are limited to 19200 baud, your logging rate will need to be severely reduced to compensate. Therefore the use of SoftSerial is not recommended.

When using a hardware serial port, Blackbox should be set to at least 115200 baud on that port. When using fast looptimes (<2500), a baud rate of 250000 should be used instead in order to reduce dropped frames.

The serial port used for Blackbox cannot be shared with any other function (e.g. GPS, telemetry) except the MSP protocol. If MSP is used on the same port as Blackbox, then MSP will be active when the board is disarmed, and Blackbox will be active when the board is armed. This will mean that you can't use the Configurator or any other function that requires MSP, such as an OSD or a Bluetooth wireless configuration app, while the board is armed.

Connect the "TX" pin of the serial port you've chosen to the OpenLog's "RXI" pin. Don't connect the serial port's RX pin to the OpenLog, as this will cause the OpenLog to interfere with any shared functions on the serial port while disarmed.

Naze32 serial port choices

On the Naze32, the TX/RX pins on top of the board are connected to UART1, and are shared with the USB connector. Therefore, MSP must be enabled on UART1 in order to use the Configurator over USB. If Blackbox is connected to the pins on top of the Naze32, the Configurator will stop working once the board is armed. This configuration is usually a good choice if you don't already have an OSD installed which is using those pins while armed, and aren't using the FrSky telemetry pins.

Pin RC3 on the side of the board is UART2's Tx pin. If Blackbox is configured on UART2, MSP can still be used on UART1 when the board is armed, which means that the Configurator will continue to work simultaneously with Blackbox logging. However, the RC3 pin is only available for use by UART2 if the receiver mode is *not* `PARALLEL_PWM`. In other words, a PPM or Serial receiver must be used. If a PWM receiver is used, the RC3 and RC4 pins are used for channel input from the receiver. Sharing UART1 between Blackbox and MSP is the only way to use Blackbox on a Naze32 with a PWM receiver.

The OpenLog tolerates a power supply of between 3.3V and 12V. If you are powering your Naze32 with a standard 5V BEC, then you can use a spare motor header's +5V and GND pins to power the OpenLog with.

Other flight controller hardware

Boards other than the Naze32 may have more accessible hardware serial devices, in which case refer to their documentation to decide how to wire up the logger. The key criteria are:

- Should be a hardware serial port rather than SoftSerial.
- Cannot be shared with any other function (GPS, telemetry) except MSP.
- If MSP is used on the same UART, MSP will stop working when the board is armed.

OpenLog configuration

Power up the OpenLog with a microSD card inside, wait 10 seconds or so, then power it down and plug the microSD card into your computer. You should find a "CONFIG.TXT" file on the card, open it up in a text editor. You should see the baud rate that the OpenLog has been configured for (usually 115200 or 9600 from the factory). Set the baud rate to match the rate you entered for the Blackbox in the Configurator's Port tab (typically 115200 or 250000).

Save the file and put the card back into your OpenLog, it will use those settings from now on.

If your OpenLog didn't write a CONFIG.TXT file, create a CONFIG.TXT file with these contents and store it in the root of the MicroSD card:

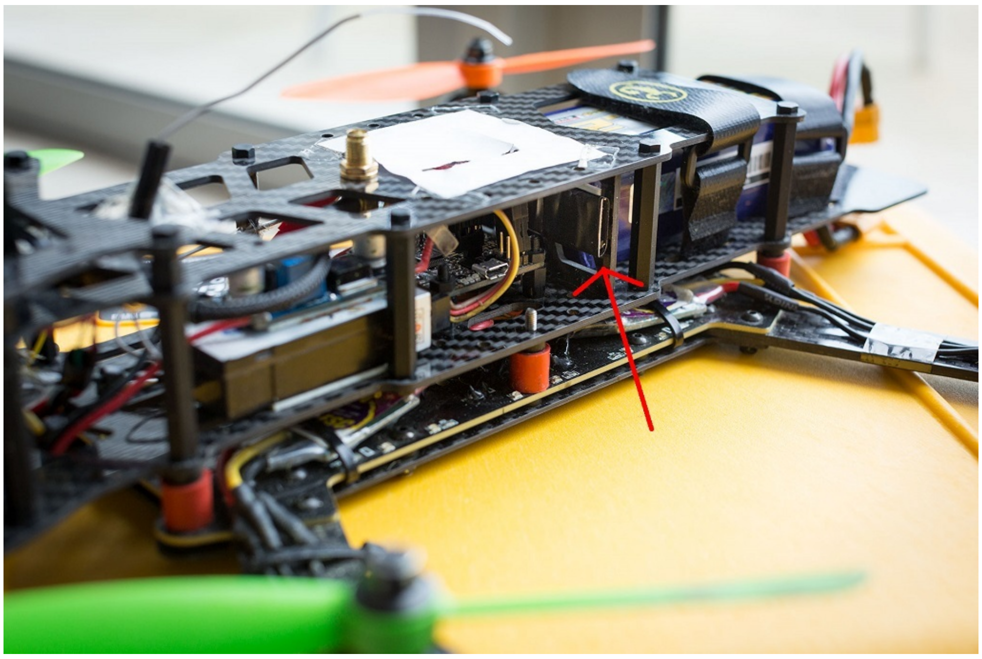
```
115200
baud
```

If you are using the original OpenLog firmware, use this configuration instead:

```
115200,26,0,0,1,0,1
baud,escape,esc#,mode,verb,echo,ignoreRX
```

OpenLog protection

The OpenLog can be wrapped in black electrical tape or heat-shrink in order to insulate it from conductive frames (like carbon fiber), but this makes its status LEDs impossible to see. I recommend wrapping it with some clear heatshrink tubing instead.



Onboard dataflash storage

Some flight controllers have an onboard SPI NOR dataflash chip which can be used to store flight logs instead of using an OpenLog.

The full version of the Naze32 and the CC3D have an onboard "m25p16" 2 megabyte dataflash storage chip. This is a small chip with 8 fat legs, which can be found at the base of the Naze32's direction arrow. This chip is not present on the "Acro" version of the Naze32.

The SPRacingF3 has a larger 8 megabyte dataflash chip onboard which allows for longer recording times.

These chips are also supported:

- Micron/ST M25P16 - 16 Mbit / 2 MByte
- Micron N25Q064 - 64 Mbit / 8 MByte
- Winbond W25Q64 - 64 Mbit / 8 MByte
- Micron N25Q0128 - 128 Mbit / 16 MByte
- Winbond W25Q128 - 128 Mbit / 16 MByte

Enable recording to dataflash

On the Configurator's CLI tab, you must enter `set blackbox_device=SPIFLASH` to switch to logging to an onboard dataflash chip, then save.

Configuring the Blackbox

The Blackbox currently provides two settings (`blackbox_rate_num` and `blackbox_rate_denom`) that allow you to control the rate at which data is logged. These two together form a fraction (`blackbox_rate_num / blackbox_rate_denom`) which decides what portion of the flight controller's control loop iterations should be logged. The default is 1/1 which logs every iteration.

If you're using a slower MicroSD card, you may need to reduce your logging rate to reduce the number of corrupted logged frames that `blackbox_decode` complains about. A rate of 1/2 is likely to work for most craft.

You can change the logging rate settings by entering the CLI tab in the [Cleanflight Configurator]() and using the `set` command, like so:

```
set blackbox_rate_num = 1
set blackbox_rate_denom = 2
```

The data rate for my quadcopter using a looptime of 2400 and a rate of 1/1 is about 10.25kB/s. This allows about 18 days of flight logs to fit on my OpenLog's 16GB MicroSD card, which ought to be enough for anybody :).

If you are logging using SoftSerial, you will almost certainly need to reduce your logging rate to 1/32. Even at that logging rate, looptimes faster than about 1000 cannot be successfully logged.

If you're logging to an onboard dataflash chip instead of an OpenLog, be aware that the 2MB of storage space it offers is pretty small. At the default 1/1 logging rate, and a 2400 looptime, this is only enough for about 3 minutes of flight. This could be long enough for you to investigate some flying problem with your craft, but you may want to reduce the logging rate in order to extend your recording time.

To maximize your recording time, you could drop the rate all the way down to 1/32 (the smallest possible rate) which would result in a logging rate of about 10-20Hz and about 650 bytes/second of data. At that logging rate, a 2MB dataflash chip can store around 50 minutes of flight data, though the level of detail is severely reduced and you could not diagnose flight problems like vibration or PID setting issues.

Usage

The Blackbox starts recording data as soon as you arm your craft, and stops when you disarm.

If your craft has a buzzer attached, you can use Cleanflight's arming beep to synchronize your Blackbox log with your flight video. Cleanflight's arming beep is a "long, short" pattern. The beginning of the first long beep will be shown as a blue line in the flight data log, which you can sync against your recorded audio track.

You should wait a few seconds after disarming your craft to allow the Blackbox to finish saving its data.

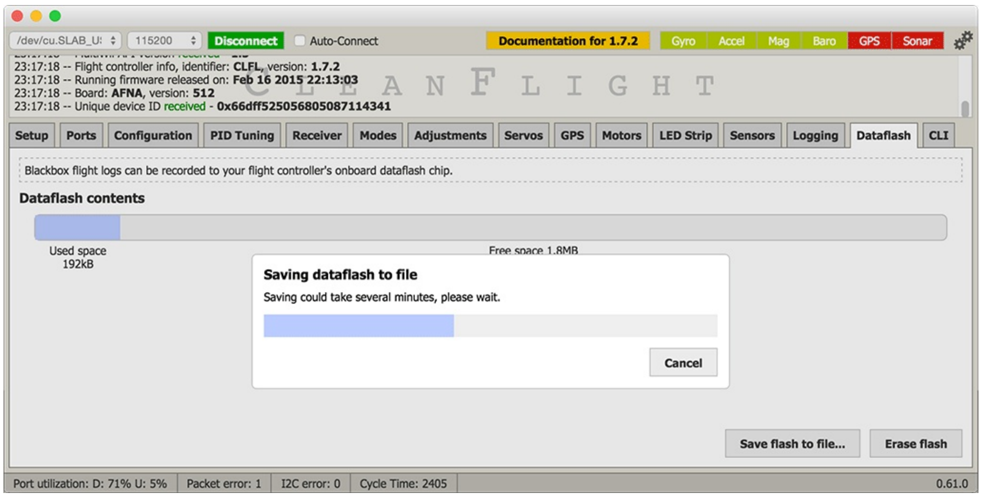
Usage - OpenLog

Each time the OpenLog is power-cycled, it begins a fresh new log file. If you arm and disarm several times without cycling the power (recording several flights), those logs will be combined together into one file. The command line tools will ask you to pick which one of these flights you want to display/decode.

Don't insert or remove the SD card while the OpenLog is powered up.

Usage - Dataflash chip

After your flights, you can use the [Cleanflight Configurator]() to download the contents of the dataflash to your computer. Go to the "dataflash" tab and click the "save flash to file..." button. Saving the log can take 2 or 3 minutes.



```
23:17:18 -- Flight controller info, identifier: CLFL, version: 1.7.2
23:17:18 -- Running firmware released on: Feb 16 2015 22:13:03
23:17:18 -- Board: AFNA, version: 512
23:17:18 -- Unique device ID received - 0x66dff525056805087114341
```

Blackbox flight logs can be recorded to your flight controller's onboard dataflash chip.

Dataflash contents

Used space 192kB

Free space 1.8MB

Saving dataflash to file

Saving could take several minutes, please wait.

Port utilization: D: 71% U: 5% | Packet error: 1 | I2C error: 0 | Cycle Time: 2405 | 0.61.0

After downloading the log, be sure to erase the chip to make it ready for reuse by clicking the "erase flash" button.

If you try to start recording a new flight when the dataflash is already full, Blackbox logging will be disabled and nothing will be recorded.

Usage - Logging switch

If you're recording to an onboard flash chip, you probably want to disable Blackbox recording when not required in order to save storage space. To do this, you can add a Blackbox flight mode to one of your AUX channels on the Configurator's modes tab. Once you've added a mode, Blackbox will only log flight data when the mode is active.

A log header will always be recorded at arming time, even if logging is paused. You can freely pause and resume logging while in flight.

Viewing recorded logs

After your flights, you'll have a series of flight log files with a .TXT extension.

You can view these .TXT flight log files interactively using your web browser with the Cleanflight Blackbox Explorer:

https://github.com/cleanflight/blackbox-log-viewer

This allows you to scroll around a graphed version of your log and examine your log in detail. You can also export a video of your log to share it with others!

You can decode your logs with the `blackbox_decode` tool to create CSV (comma-separated values) files for analysis, or render them into a series of PNG frames with `blackbox_render` tool, which you could then convert into a video using another software package.

You'll find those tools along with instructions for using them in this repository:

https://github.com/cleanflight/blackbox-tools

Migrating from baseflight

Procedure

First ensure your main flight battery is disconnected or your props are off!

Before flashing with cleanflight, dump your configs for each profile via the CLI and save to a text file.

```
profile 0
dump
profile 1
dump
profile 2
dump
```

Then after flashing cleanflight paste the output from each dump command into the cli, switching profiles as you go.

You'll note that some commands are not recognised by cleanflight when you do this. For the commands that are not recognised look up the new configuration options and choose appropriate values for the settings. See below for a list of differences.

Once you've done this for the first profile, save the config. Then verify your config is OK, e.g. features serial ports, etc. When you've verified the first profile is OK repeat for the other profiles.

It's also advisable to take screenshots of your AUX settings from baseflight configurator and then after re-applying the settings verify your aux config is correct - aux settings are not backwards compatible.

CLI command differences from baseflight

In general all CLI commands use underscore characters to separate words for consistency. In baseflight the format of CLI commands is somewhat haphazard.

gps_baudrate

reason: new serial port configuration.

See `serial` command.

gps_type

reason: renamed to `gps_provider` for consistency

serialrx_type

reason: renamed to `serialrx_provider` for consistency

rssi*aux*channel

reason: renamed to `rssi_channel` for improved functionality

Cleanflight supports using any RX channel for rssi. Baseflight only supports AUX1 to 4.

In Cleanflight a value of 0 disables the feature, a higher value indicates the channel number to read RSSI information from.

Example: to use RSSI on AUX1 in Cleanflight use `set rssi_channel = 5`, since 5 is the first AUX channel (this is equivalent to `set rssi_aux_channel = 1` in Baseflight).

failsafe*detect*threshold

reason: improved functionality

See `rx_min_usec` and `rx_max_usec` in Failsafe documentation.

emfavoidance

reason: renamed to `emf_avoidance` for consistency

yawrate

reason: renamed to `yaw_rate` for consistency

yawdeadband

reason: renamed to `yaw_deadband` for consistency

midrc

reason: renamed to `mid_rc` for consistency

mincheck

reason: renamed to `min_check` for consistency

maxcheck

reason: renamed to `max_check` for consistency

minthrottle

reason: renamed to `min_throttle` for consistency

maxthrottle

reason: renamed to `max_throttle` for consistency

mincommand

reason: renamed to `min_command` for consistency

deadband3d_low

reason: renamed to `3d_deadband_low` for consistency

deadband3d_high

reason: renamed to `3d_deadband_high` for consistency

deadband3d_throttle

reason: renamed to `3d_deadband_throttle` for consistency

neutral3d

reason: renamed to `3d_neutral` for consistency

alt*hold*throttle_neutral

reason: renamed to `alt_hold_deadband` for consistency

gimbal_flags

reason: seperation of features.

see `gimbal_mode` and `CHANNEL_FORWARDING` feature

Flight controller hardware

The current focus is geared towards flight controller hardware that use the STM32F303 and legacy STM32F103 series processors. The core logic is separated from the hardware drivers, porting to other processors is possible.

The core set of supported flyable boards are:

- AlienWii32
- CC3D
- CJMCU
- Flip32+
- Naze32
- Sparky
- SPRacingF3

Cleanflight also runs on the following developer boards:

- STM32F3Discovery
- Port103R
- EUSTM32F103RB

There is also limited support for the following boards which may be removed due to lack of users or commercial availability.

- Olimexino
- Naze32Pro
- STM32F3Discovery with Chebuzz F3 shield.

NOTE: Users are advised against purhasing boards that have CPUs with less than 256KB of EEPROM space - available features may be limited. NOTE: Hardware developers should not design new boards that have CPUs with less than 256KB EEPROM space.

Each board has it's pros and cons, before purchasing hardware the main thing to check is if the board offers enough serial ports and input/output pins for the hardware you want to use with it and that you can use them at the same time. On some boards some features are mutually exclusive.

Please see the board-specific chapters in the manual for wiring details.

There are off-shoots (forks) of the project that support the STM32F4 processors as found on the Revo and Quanton boards.

Where applicable the chapters also provide links to other hardware that is known to work with Cleanflight, such as receivers, buzzers, etc.

Board - AlienWii32 (ALIENWIIF1 and ALIENWIIF3 target)

The AlienWii32 is actually in prototype stage and few samples exist. There are some different variants and field testing with some users is ongoing. The information below is preliminary and will be updated as needed.

Here are the hardware specifications:

- STM32F103CBT6 MCU (ALIENWIIF1)
- STM32F303CCT6 MCU (ALIENWIIF3)
- MPU6050 accelerometer/gyro sensor unit
- 4-8 x 4.2A brushed ESCs, integrated, to run the strongest micro motors
- extra-wide traces on the PCB, for maximum power throughput
- USB port, integrated
- (*) serial connection for external DSM2/DSMX sat receiver (e.g. Spektrum SAT, OrangeRx R100, Lemon RX or Deltang Rx31)
- ground and 3.3V for the receiver
- hardware bind plug for easy binding
- motor connections are at the corners for a clean look with reduced wiring
- dimensions: 29x33mm
- direct operation from an single cell lipoly battery
- 3.3V LDO power regulator (older prototypes)
- 3.3V buck-boost power converter (newer prototypes and production versions)
- battery monitoring with an LED for buzzer functionality (actualy for an ALIENWIIF3 variant)

(*) Spektrum Compatible DSM2 satellites are supported out of the box. DSMX sat will work with DSM2 protocol with default settings (DSM2, 11bit, 11ms is preset). This is chosen for maximum compatibility. For optimal connection it is recommended to adjust settings to match the capabilities of your transmitter and satellite receiver. If possible it is recommended to use the

DSMX protocol since it is known as more reliable. Also to make use of additional channels you should adjust the following two parameters with the Cleanflight Configurator.

```
set serialrx_provider = 1   (0 for 1024bit, 1 for 2048bit)
set spektrum_sat_bind = 5
```

For more detail of the different bind modes please refer the Spektrum Bind document

Deltang receivers in serial mode will work like any other Spektrum satellite receiver (10bit, 22ms) only the bind process will be different.

The pin layout for the ALIENWIIF1 is very similar to NAZE32 or the related clones (MW32, Flip32, etc.). The hardware bind pin is connected to pin 41 (PB5). The pin layout for the ALIENWIIF3 is similar to Sparky. The hardware bind pin is connected to pin 25 (PB12). The AlienWii32 firmware will be built as target ALIENWIIF1 or ALIENWIIF3. The firmware image will come with alternative default settings which will give the user a plug and play experience. There is no computer needed to get this into the air with an small Quadcopter. An preconfigured custom mixer for an Octocopter is part of the default settings to allow clean straight wiring with the AlienWii32. The mixer can be activated with "mixer custom" in the CLI. To use the AlienWii32 in an Hexa- or Octocopter or to do some more tuning additional configuration changes can be done as usual in the CLI or the Cleanflight configurator.

Flashing the firmware

The AlienWii32 F1 board can be flashed like the Naze board or the related clones. All the different methods will work in the same way.

The AlienWii32 F3 board needs to be flashed via the USB port in DFU mode. Flashing via the Cleanflight GUI is not possible yet. The DFU mode can be activated via setting the BOOT0 jumper during power on of the board. The second method is to connect with an terminal program (i.e. Putty) to the board and enter the character "R" immediately after connecting. Details about the flashing process can be found in the related section of the Sparky documentation. The BOOT0 jumper should be removed and the board needs to be repowerd after firmware flashing. Please be aware, during reboot of the AlienWii F3 board, the GUI will disconnect and an manual reconnect is required.# Board - CC3D

The OpenPilot Copter Control 3D aka CC3D is a board more tuned to Acrobatic flying or GPS based auto-piloting. It only has one sensor, the MPU6000 SPI based Accelerometer/Gyro. It also features a 16Mbit SPI based EEPROM chip. It has 6 ports labeled as inputs (one pin each) and 6 ports labeled as motor/servo outputs (3 pins each).

If issues are found with this board please report via the github issue tracker.

The board has a USB port directly connected to the processor. Other boards like the Naze and Flip32 have an on-board USB to uart adapter which connect to the processor's serial port instead.

The board cannot currently be used for hexacopters/octocopters.

Tricopter & Airplane support is untested, please report success or failure if you try it.

Pinouts

The 8 pin RC*Input connector has the following pinouts when used in RX* PPM/RX_SERIAL mode

Pin	Function	Notes
1	Ground	
2	+5V	
3	PPM Input	Enable `feature RX_PPM`
4	SoftSerial1 TX / Sonar trigger	
5	SoftSerial1 RX / Sonar Echo	
6	Current	Enable `feature CURRENT_METER`. Connect to the output of a current sensor, 0v-3.3v input
7	Battery Voltage sensor	Enable `feature VBAT`. Connect to main battery using a voltage divider, 0v-3.3v input
8	RSSI	Enable `feature RSSI_ADC`. Connect to the output of a PWM-RSSI conditioner, 0v-3.3v input

The 6 pin RC*Output connector has the following pinouts when used in RX* PPM/RX_SERIAL mode

Pin	Function	Notes
1	MOTOR 1	
2	MOTOR 2	
3	MOTOR 3	
4	MOTOR 4	
5	LED Strip	
6	Unused	

The 8 pin RC*Input connector has the following pinouts when used in RX* PARALLEL_PWM mode

Pin	Function	Notes
1	Ground	
2	+5V	
3	Unused	
4	CH1	
5	CH2	
6	CH3	
7	CH4/Battery Voltage sensor	CH4 if battery voltage sensor is disabled
8	CH5/CH4	CH4 if battery voltage monitor is enabled

The 6 pin RC*Output connector has the following pinouts when used in RX* PARALLEL_PWM mode

Pin	Function	Notes
1	MOTOR 1	
2	MOTOR 2	
3	MOTOR 3	
4	MOTOR 4	
5	Unused	
6	Unused	

Serial Ports

Value	Identifier	Board Markings	Notes
1	VCP	USB PORT	
2	USART1	MAIN PORT	Connected to an MCU controllable inverter
3	USART3	FLEX PORT	
4	SoftSerial	RC connector	Pins 4 and 5 (Tx and Rx respectively)

The SoftSerial port is not available when RX*PARALLEL*PWM is used. The transmission data rate is limited to 19200 baud.

To connect the GUI to the flight controller you just need a USB cable to use the Virtual Com Port (VCP) or you can use UART1 (Main Port).

CLI access is only available via the VCP by default.

Main Port

The main port has MSP support enabled on it by default.

The main port is connected to an inverter which is automatically enabled as required. For example, if the main port is used for SBus Serial RX then an external inverter is not required.

Flex Port

The flex port will be enabled in I2C mode unless USART3 is used. You can connect external I2C sensors and displays to this port.

You cannot use USART3 and I2C at the same time.

Flex port pinout

Pin	Signal	Notes
1	GND	
2	VCC unregulated	
3	I2C SCL / UART3 TX	3.3v level
4	I2C SDA / UART3 RX	3.3v level (5v tolerant

Flashing

There are two primary ways to get Cleanflight onto a CC3D board.

- Single binary image mode - best mode if you don't want to use OpenPilot.
- OpenPilot Bootloader compatible image mode - best mode if you want to switch between OpenPilot and Cleanflight.

Single binary image mode.

The entire flash ram on the target processor is flashed with a single image.

The image can be flashed by using a USB to UART adapter connected to the main port when the CC3D is put into the STM32 bootloader mode, achieved by powering on the CC3D with the SBL/3.3v pads bridged.

OpenPilot Bootloader compatible image mode.

The initial section of flash ram on the target process is flashed with a bootloader which can then run the code in the remaining area of flash ram.

The OpenPilot bootloader code also allows the remaining section of flash to be reconfigured and re-flashed by the OpenPilot Ground Station (GCS) via USB without requiring a USB to uart adapter.

The following features are not available: * Display * Sonar

Restoring OpenPilot bootloader

If you have a JLink debugger, you can use JLinkExe to flash the open pilot bootloader.

1. Run JLinkExe `/Applications/SEGGER/JLink/JLinkExe`
2. `device STM32F103CB`
3. `r`
4. `h`
5. `loadbin opbl.bin, 0x08000000`
6. `q`

7. Re-plug CC3D.

Here's an example session:

```
$ /Applications/SEGGER/JLink/JLinkExe
SEGGER J-Link Commander V4.90c ('?' for help)
Compiled Aug 29 2014 09:52:38
DLL version V4.90c, compiled Aug 29 2014 09:52:33
Firmware: J-Link ARM-OB STM32 compiled Aug 22 2012 19:52:04
Hardware: V7.00
S/N: -1
Feature(s): RDI,FlashDL,FlashBP,JFlash,GDBFull
VTarget = 3.300V
Info: Could not measure total IR len. TDO is constant high.
Info: Could not measure total IR len. TDO is constant high.
No devices found on JTAG chain. Trying to find device on SWD.
Info: Found SWD-DP with ID 0x1BA01477
Info: Found Cortex-M3 r1p1, Little endian.
Info: FPUnit: 6 code (BP) slots and 2 literal slots
Info: TPIU fitted.
Cortex-M3 identified.
Target interface speed: 100 kHz
J-Link>device STM32F103CB
Info: Device "STM32F103CB" selected (128 KB flash, 20 KB RAM).
Reconnecting to target...
Info: Found SWD-DP with ID 0x1BA01477
Info: Found SWD-DP with ID 0x1BA01477
Info: Found Cortex-M3 r1p1, Little endian.
Info: FPUnit: 6 code (BP) slots and 2 literal slots
Info: TPIU fitted.
J-Link>r
Reset delay: 0 ms
Reset type NORMAL: Resets core & peripherals via SYSRESETREQ & VECTRESET bit.
J-Link>h
PC = 0800010C, CycleCnt = 00000000
R0 = 0000000C, R1 = 0000003F, R2 = 00000000, R3 = 00000008
R4 = 00003000, R5 = 023ACEFC, R6 = 200000F0, R7 = 20000304
R8 = 023B92BC, R9 = 00000000, R10= ED691105, R11= F626177C
R12= 000F0000
SP(R13)= 20000934, MSP= 20000934, PSP= 20000934, R14(LR) = FFFFFFFF
XPSR = 01000000: APSR = nzcvq, EPSR = 01000000, IPSR = 000 (NoException)
CFBP = 00000000, CONTROL = 00, FAULTMASK = 00, BASEPRI = 00, PRIMASK = 00
J-Link>loadbin opbl.bin, 0x08000000
Downloading file [opbl.bin]...
WARNING: CPU is running at low speed (8004 kHz).
Info: J-Link: Flash download: Flash download into internal flash skipped. Flash contents alread
y match
Info: J-Link: Flash download: Total time needed: 0.898s (Prepare: 0.709s, Compare: 0.128s, Eras
e: 0.000s, Program: 0.000s, Verify: 0.000s, Restore: 0.059s)
O.K.
J-Link>q
$
```

Board - ChebuzzF3

The ChebuzzF3 is a daugter board which connects to the bottom of an STM32F3Discovery board and provides pin headers and ports for various FC connections.

All connections were traced using a multimeter and then verified against the TauLabs source code using the revision linked.

https://github.com/TauLabs/TauLabs/blob/816760dec2a20db7fb9ec1a505add240e696c31f/flight/targets/flyingf3/board-info/board_hw_defs.c

Connections

Board orientation.

These notes assume that when the board is placed with the header pins facing up, the bottom right of the board is next to the 8 sets of INPUT pin headers. Inner means between the two rows of header sockets, outer means between the left/right board edges and the header sockets.

SPI2 / External SPI

sclk GPIOB 13 miso GPIOB 14 mosi GPIOB 15

There are 4 pins, labelled CS1-4 next to a label that reads Ext SPI. The 3rd pin is connected to the flash chip on the bottom right inner of the board. The other pins on the flash chip are wired up to PB3/4/5

SPI3 / SPI

sclk GPIOB 3 miso GPIOB 4 mosi GPIOB 5

ssel 1 GPIOB 10 / Ext SPI CS1 ssel 2 GPIOB 11 / Ext SPI CS2 ssel 3 GPIOB 12 / Ext SPI CS3 - wired up to Slave Select of M25P16 15MBitFlash chip ssel 4 GPIOB 13 / Ext SPI CS4 - not usable since it is used for SPI2 sclk

RC Input

INPUT PA8 / CH1 - TIM1*CH1 PB8 / CH2 - TIM16* CH1 PB9 / CH3 - TIM17*CH1 PC6 / CH4 - TIM8* CH1 PC7 / CH5 - TIM8*CH2 PC8 / CH6 - TIM8*CH3 PF9 / CH7 - TIM15*CH1 PF10 / CH8 - TIM15* CH2

PWM Outputs

OUTPUT PD12 / CH1 - TIM4*CH1 PD13 / CH2 - TIM4* CH2 PD14 / CH3 - TIM4*CH3 PD15 / CH4 - TIM4* CH4 PA1 / CH5 - TIM2*CH2 PA2 / CH6 - TIM2* CH3 PA3 / CH7 - TIM2*CH4 PB0 / CH8 - TIM3* CH3 PB1 / CH9 - TIM3*CH4 PA4 / CH10 - TIM3* CH2

Other ports

There is space for a MS5611 pressure sensor at the top left inner of the board.

There is an I2C socket on the left outer of the board which connects to a PCA9306 I2C level shifter directly opposite (inner). The PCA9306 is not populated on some boards and thus the I2C socket is unusable.

There is a CAN socket on the top right outer of the board which connects to a MAX3015 CAN Tranceiver. The MAX3015 is not populated on some boards and thus the CAN socket is unusable.

There are some solder pads labelled Ext 1-4 at the top right inner of the board.

GPIOE 6 / PE6 / Ext 1 GPIOD 3 / PD3 / Ext 2 GPIOD 4 / PD4 / Ext 3 GPIOB 3 / PB3 / Ext 4

There are some solder pads labelled ADC0-3 & Diff Press at the top left inner of the board They are connected to the ADC socket at the top left outer of the board

PC3 / Diff Press - ADC12*IN9 (Differential Pressure) PC2 / ADC2 - ADC12* IN8 PC1 / ADC1 - ADC12*IN7 PC0 / ADC0 - ADC12*IN6

There is space for a MPXV5004/MPVZ5004 differential pressure sensor, if populated it's analog pin connects to PC3.

There are sockets for 5 UARTs labelled USART1-5.

There is a socket labelled RX_IN.

GPIOD 2 / PD2 / RX_IN

Board - CJMCU

The CJMCU is a tiny (80mm) board running a STM32F103, which contains a 3-Axis Compass (HMC5883L) and an Accelerometer/Gyro (MPU6050).

This board does not have an onboard USB-Serial converter, so an external adapter is needed.

Hardware revisions

Revision	Notes
1	No boot jumper pads by LED1. Uses blue and red LEDs
2	Boot jumper pads presoldered with pins and a jumper by LED1. Uses green and red LEDs.

Version 2 boards are supported from firmware v1.4.0 onwards, do NOT flash earlier versions to version 2 boards.

Pins

RX Connections

Pin Label	Description
PA0	RC Channel 1
PA1	RC Channel 2
PA2	RC Channel 3 / USART2 TX
PA3	RC Channel 4 / USART2 RX
VCC	Power (See note)
GND	Ground

NOTE: The VCC RX Pin is not regulated and will supply what ever voltage is provided to the board, this will mean it'll provide 5v if a 5v serial connection is used. Be careful if you are using a voltage sensitive RX. A regulated 3.3v supply can be found on the top pin of column 1, just below the RX GND pin.

Serial Connections

USART1 (along with power) is on the following pins.

Pin Label	Description
TX1	UART1 TX
RX1	UART2 RX
GND	Ground
3V3	Power +3.3v
5V	Power +5v

USART2 is the following pins.

Pin Label	Description
PA2	USART2 TX
PA3	USART2 RX

Power Connections

Pin Label	Description
Power +	Power - 1 Cell 3.7v Max
Power -	Ground

Motor Connections

In standard QUADX configuration, the motors are mapped:

Cleanflight	CJMCU
Motor 1	Motor3
Motor 2	Motor2
Motor 3	Motor4
Motor 4	Motor1

It is therefore simplest to wire the motors: * Motor 1 -> Clockwise * Motor 2 -> Anti-Clockwise * Motor 3 -> Clockwise * Motor 4 -> Anti-Clockwise

If you are using the Hubsan x4/Ladybird motors, clockwise are Blue (GND) / Red (VCC) wires, anticlockwise are Black (GND) / White (VCC). i.e. there is one wire on each motor out of the standard RED/BLACK VCC/GND polarity colors that can be used to identify polarity.

If you have wired as above, Motor1/Motor2 on the board will be forward.

Connecting a Serial-USB Adapter

You will need a USB -> Serial UART adapter. Connect:

Adapter	CJMCU
Either 3.3v OR 5v	The correct 3.3v OR 5v pin
RX	TX
TX	RX

When first connected this should power up the board, and will be in bootloader mode. If this does not happen, check the charge switch is set to POW. After the flashing process has been completed, this will allow access via the cleanflight configurator to change settings or flash a new firmware.

WARNING: If the motors are connected and the board boots into the bootloader, they will start to spin after around 20 seconds, it is recommended not to connect the motors until the board is flashed.

Flashing

To flash the board: * Open Cleanflight Configurator * Choose the latest CJMCU firmware from the list. * Select "Load Firmware [Online]" and wait for the firmware to download. * Tick "No Reboot Sequence" and "Full Chip Erase" * Connect the USB->Serial adapter to the board * Select the USB-UART adapter from the top left box * Click "Flash Firmware" * You should see "Programming: SUCCESSFUL" in the log box * Click "Connect" -> This should open the "Initial Setup" tab and you should see sensor data from the quad shown * Unplug the quad and short the 2 "BOOT0" pins. Revision 1 boards require this to be soldered, revision 2 boards can connect the included jumper to the two pre-soldered pins - This prevents the board from going into bootloader mode on next boot, if anything goes wrong, simply disconnect these two pins and the bootloader will start, allowing you to reflash. You cannot overwrite the bootloader.

Charging

The CJMCU has on it a TP4056 Lithium battery charging IC that can charge a 1S battery at 1A using a provided 5v supply attached to the 5v serial pin.

To charge an attached battery: * Set the power switch to OFF * Set the charge switch to CHG * Plug in a 1S battery to the battery pins * Plug in a 5v supply to the 5v serial pins

The charger will finish when either the battery reaches 4.2v, or the battery's voltage is greater than the charger's input voltage.

The two nearby LEDs will show the status of charging:

Status	Green LED	Red LED
Charging	On	Off
Finished	Off	On
5v not connected	Off	Off
Batt not connected	Flashing	On

Helpful Hints

- If you are only using a 4 channel RX, in the auxiliary configuration tab, you can add a "Horizon" mode range around 1500 for one of the the AUX channels which will result in it being always on
- Enabling the feature MOTOR_STOP helps with crashes so it doesn't try to keep spinning on its back
- When the power runs low, the quad will start jumping around a bit, if the flight behaviour seems strange, check your batteries charge

Board - Colibri RACE

The Colibri RACE is a STM32F3 based flight control designed specifically to work with the TBS POWERCUBE multi rotor stack.

Hardware Features:

- STM32F303 based chipset for ultimate performance
- PPM, SBUS, DSM, DSMX input (5V and 3.3V provided over internal BUS). No inverters or hacks needed.

- 6 PWM ESC output channels (autoconnect, internal BUS)

    - RGB LED strip support incl. power management
    - Extension port for GPS / external compass / pressure sensor
    - UART port for peripherals (Blackbox, FrSky telemetry etc.)

- Choose between plug & play sockets or solder pads for R/C and buzzer
- 5V buzzer output
- MPU6500 new generation accelerometer/gyro
- 3x status LED (DCDC pwr/ 3.3V pwr/ status)
- Battery monitoring for 12V, 5V and VBat supply
- Size: 36mmx36mm (30.5mm standard raster)
- Weight: 4.4g

For more details please visit: http://www.team-blacksheep.com/powercube

Serial Ports

Value	Identifier	Board Markings	Notes
1	VCP	USB PORT	Main Port For MSP
2	USART1	FREE PORT	PC4 and PC5(Tx and Rx respectively)
3	USART2	PPM Serial	PA15
4	USART3	GPS PORT	PB10 and PB11(Tx and Rx respectively)

Pinouts

Full pinout details are available in the manual, here:

http://www.team-blacksheep.com/colibri_race

SWD - ICSP

Pin	Function	Notes
1	VCC_IN	3.3 Volt
2	SWDIO	
3	nRESET	
4	SWCLK	
5	Ground	
6	SWO/TDO	

Internal Bus

Pin	Function	Notes
1	PWM1	MOTOR 1
2	PWM2	MOTOR 2
3	PWM3	MOTOR 3
4	PWM4	MOTOR 4
5	PWM5	MOTOR 5 (For Y6 or Hex X)
6	PWM6	MOTOR 6 (For Y6 or Hex X)
7	BST SDA	Use For TBS CorePro Control Device
8	BST SCL	Use For TBS CorePro Control Device
9	PWM7	Can be a normal GPIO (PA1) or PWM
10	PWM8	Can be a normal GPIO (PA2) or PWM
11	12.2V DCDC	If 12v is detected, the Blue LED will turn on
12	5.1V DCDC	Voltage for MCU

Servo

Pin	Function	Notes
1	Ground	
2	VCC_OUT	5.1 Volt output to LCD Strip
3	PWM Servo	PB14 - PWM10

IO_1 - LED Strip

Pin	Function	Notes
1	LED_STRIP	Enable `feature LED_STRIP`
2	VCC_OUT	5.1 Volt output to LCD Strip
3	Ground	

IO_2 - Sensor Interface

Pin	Function	Notes
1	VCC_OUT	4.7 Volt output to the device
2	Ground	
3	UART3 TX	GPS
4	UART3 RX	GPS
5	SDA	mag, pressure, or other i2c device
6	SCL	mag, pressure, or other i2c device

IO_3 - RC input

IO_3 is used for RX_PPM/RX_SERIAL. Under the `PORT` tab, set RX_SERIAL to UART2 when using RX_SERIAL.

Pin	Function	Notes
1	PPM/Serial	Can PPM or Serial input
2	VCC_OUT	3.3 Volt output to the device
3	Ground	
4	VCC_OUT	5.1 Volt output to the device

IO_4 - Buzzer

Pin	Function	Notes
1	BUZZER	Normal high (5.1v)
2	VCC_OUT	5.1 Volt output to the device

IO_5 - Free UART

Pin	Function	Notes
1	UART1 TX	Free UART
2	UART1 RX	Free UART
3	Ground	
4	VCC_OUT	4.7 Volt output to the device

IO_6 - IR TX (extension)

Pin	Function	Notes
1	IR TX	
2	Ground	

Board - MotoLab

The MOTOLAB build target supports the STM32F3-based boards provided by MotoLab.

At present this includes the TornadoFC and MotoF3. The TornadoFC is described here:

http://www.rcgroups.com/forums/showthread.php?t=2473157

The MotoF3 documentation will be provided when the board is available.

Both boards use the STM32F303 microcontroller and have the following features:

- 256K bytes of flash memory
- Floating point math coprocessor
- Three hardware serial port UARTs
- USB using the built-in USB phy that does not interfere with any hadware UART
- Stable voltage regulation
- High-current buzzer/LED output
- Serial LED interface
- Low-pass filtered VBAT input with 1/10 divider ratio
- 8 short-circuit protected PWM outputs, with 5V buffering on the TornadoFC
- On-board 6S-compatible switching regulator (MotoF3)
- Direct mounting option for a Pololu switching regulator for up to 6S lipo operation (TornadoFC)

Flashing

The MotoLab boards use the internal DFU USB interface on the STM32F3 microcontroller which is not compatible with the Cleanflight configurator flashing tool.

Instead, on Windows you can use the Impulse Flashing Utility from ImpulseRC, available here:

http://www.warpquad.com/ImpulseFlash.zip

Download and unzip the program. Start the program, plug in the USB on the target board, and drag and drop the intended binary file onto the program icon. The program will put the STM32F3 into bootloader mode automatically and load the binary file to the flash.

For programming on Linux, use the dfu-util program which is installed by default on Ubuntu-based systems. Connect the boot pins on the board and plug in the USB.

Verify that the system identifies the DFU device with this command:

```
dfu-util -l
```

The output should list a "Found DFU" device, something like this:

```
dfu-util 0.5

(C) 2005-2008 by Weston Schmidt, Harald Welte and OpenMoko Inc.
(C) 2010-2011 Tormod Volden (DfuSe support)
This program is Free Software and has ABSOLUTELY NO WARRANTY

dfu-util does currently only support DFU version 1.0

Found DFU: [0483:df11] devnum=0, cfg=1, intf=0, alt=0, name="@Internal Flash  /0x08000000/128*0
002Kg"
Found DFU: [0483:df11] devnum=0, cfg=1, intf=0, alt=1, name="@Option Bytes  /0x1FFFF800/01*016
e"
```

Use this command to load the binary file to the flash memory on the board:

```
dfu-util --alt 0 -s 0x08000000 -D <binfile>
```

The output should look something like this:

```
dfu-util 0.5

(C) 2005-2008 by Weston Schmidt, Harald Welte and OpenMoko Inc.
(C) 2010-2011 Tormod Volden (DfuSe support)
This program is Free Software and has ABSOLUTELY NO WARRANTY

dfu-util does currently only support DFU version 1.0

Opening DFU USB device... ID 0483:df11
Run-time device DFU version 011a
Found DFU: [0483:df11] devnum=0, cfg=1, intf=0, alt=0, name="@Internal Flash  /0x08000000/128*0
002Kg"
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuDNLOAD-IDLE, status = 0
aborting previous incomplete transfer
```

```
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 011a
Device returned transfer size 2048
No valid DFU suffix signature
Warning: File has no DFU suffix
DfuSe interface name: "Internal Flash  "
```

A binary file is required for the Impulse flashing Utility and dfu-util. The binary file can be built as follows:

```
make TARGET=MOTOLAB clean
make TARGET=MOTOLAB binary
```

To completely erase the flash, create an all-zero file with this command on linux:

```
dd if=/dev/zero of=zero.bin bs=1 count=262144
```

Todo

Pinout documentation

Board - Naze32

The Naze32 target supports all Naze hardware revisions. Revision 4 and 5 are used and frequently flown by the primary maintainer. Previous Naze hardware revisions may have issues, if found please report via the github issue tracker.

Serial Ports

Value	Identifier	RX	TX	Notes
1	USART1	RX / PA10	TX / PA9 / TELEM	TELEM output is always inverted (for FrSky). Internally connected to USB port via CP2102 IC
2	USART2	RC4 / PA3	RC3 / PA2	
4	SOFTSERIAL1	RC5 / PA6	RC6 / PA7	
5	SOFTSERIAL2	RC7 / PB0	RC8 / PB1	

- You cannot use USART1/TX/TELEM pins at the same time.
- You may encounter flashing problems if you have something connected to the RX/TX pins. Try disconnecting RX/TX.

Pinouts

The 10 pin RC I/O connector has the following pinouts when used in RX*PPM/RX*SERIAL mode.

Pin	Identifier	Function	Notes
1		Ground	
2	Circle	+5V	
3	1	RX_PPM	Enable `feature RX_PPM`
4	2	RSSI_ADC	Enable `feature RSSI_ADC`. Connect to the output of a PWM-RSSI conditioner, 0v-3.3v input
5	3	USART2 TX	
6	4	USART2 RX	
7	5	LED_STRIP	Enable `feature LED_STRIP`
8	6	unused	
9	7	Sonar Trigger	
10	8	Sonar Echo / CURRENT	Enable `feature CURRENT_METER` Connect to the output of a current sensor, 0v-3.3v input

When SOFTSERIAL is enabled, LED*STRIP and CURRENT* METER are unavailable, but two SoftSerial ports are made available to use instead.

Pin	Identifier	Function	Notes
7	5	SOFTSERIAL1 RX	Enable `feature SOFTSERIAL`
8	6	SOFTSERIAL1 TX	
9	7	SOFTSERIAL2 RX	
10	8	SOFTSERIAL2 TX	

Recovery

Board

- Short the two pads labelled 'Boot' **taking extra care not to touch the 5V pad**
- Apply power to the board
- Remove the short on the board

Cleanflight configurator

- Select the correct hardware and the desired release of the Clearflight firmware
- Put a check in the "No reboot sequence"

- Flash firmware

```
/-------------------\
|O                 O|
| []5V              |
| [][]Boot          |
|                   |
|                   |
|                   |
|                   |
|O                 O|
\-------[USB]-------/
```

Board - Olimexino

The Olimexino is a cheap and widely available development board

This board is not recommended for cleanflight development because many of the pins needed are not broken out to header pins. A better choice for development is the Port103R, EUSTM32F103RB (F1) or the STM32F3Discovery (F3).

Connections

RC Input

INPUT

PA0 CH1 - D2 - PWM1 / PPM PA1 CH2 - D3 - PWM2 / PWM RSSI PA2 CH3 - D1 - PWM3 / UART2 TX PA3 CH4 - D0 - PWM4 / UART2 RX PA6 CH5 - D12 - PWM5 / SOFTSERIAL1 RX PA7 CH6 - D11 - PWM6 / SOFTSERIAL1 TX PB0 CH7 - D27 - PWM7 / SOFTSERIAL2 RX PB1 CH8 - D28 - PWM8 / SOFTSERIAL2 TX

PWM Output

OUTPUT PA8 CH1 - PWM9 - D6 PA11 CH2 - PWM10 - USBDM PB6 CH3 - PWM11 - D5 PB7 CH4 - PWM12 - D9 PB8 CH5 - PWM13 - D14 PB9 CH6 - PWM14 - D24

Olimexino Shield V1

Headers for a CP2102 for UART1

Top left

6 way header pinouts (left to right)

1 - N/C 2 - N/C 3 - N/C 4 - D7 - UART1 TX / CP2102 RX 5 - D8 - UART1 RX / CP2102 TX 6 - GND

Headers for PPM, RSSI, SoftSerial1 inputs and Motor outputs

Top centre

Top Row = GROUND Middle Row = 5V Bottom Row = Signals

Signal pinouts (left to right)

1 - D2 - PWM1 - PPM 2 - D3 - PWM2 - RSSI 3 - D11 - PWM6 - INPUT CH6 / SS1 TX 4 - D12 - PWM5 - INPUT CH5 / SS1 RX 5 - D5 - PWM11 - OUTPUT CH3 6 - D9 - PWM12 - OUTPUT CH4 7 - D14 - PWM13 - OUTPUT CH5 8 - D24 - PWM14 - OUTPUT CH6

SoftSerial 1 - Headers for FTDI

Top Right

6 way header pinouts (left to right)

1 - N/C 2 - D11 - SS1 or UART2 TX / FTDI RX 3 - D12 - SS1 or UART2 RX / FTDI TX 4 - N/C 5 - N/C 6 - GND

Top Right

3 way power selector header

1 - VIN 2 - 5V from FTDI 3 - N/C - Jumper Storage

Middle Left

3 way power selector header

1 - VIN 2 - 5V from CP2102 3 - N/C - Jumper Storage

Use either power selector to supply VIN from the serial port adapter sockets, ensure that both power selectors are not enabled at the same time. One or both of the power selector jumpers MUST be in the jumper storage position.

Sonar

Inner Middle Bottom Right

4 Header pins (top to bottom)

1 - VIN 2 - Trigger 3 - Echo 4 - GND

Serial IO & Serial Loopback

Bottom right

The header pins allows combinations of serial loopback and Serial IO. Any amount of connections or jumpers can be used at the same time.

Jumper positions

> < = Horizontal jumper

v = Vertical jumper ^

> <- FTDI RX connected to SS1 TX <- FTDI TX connected to SS1 RX

->< FTDI RX connected to UART2 TX ->< FTDI TX connected to UART2 RX

-v- FTDI LOOPBACK -^-

v-- SS1 LOOPBACK ^--

--v UART2 LOOPBACK --^

6 way header pinouts (top left to bottom right)

123 456

1 - SS 1 TX 2 - FTDI RX 3 - UART2 TX 4 - SS1 RX 5 - FTDI TX 6 - UART2 RX

Bottom Right

HoTT Telemetry port

When the HoTT enable jumper is on pins 2 and 3 then HoTT data can be received/transmitted on either serial port depending on the placement of the Derial IO selection jumpers.

When not in use the HoTT enable jumper can be stored on pins 3 and 4

The HoTT telemetry is connected to the center pins (2 & 5) of the Serial IO header.

4 way header (left to right)

1 - HoTT Telemetry In/Out 2 - HoTT Enable Jumper 3 - HoTT Enable Jumper 4 - N/C - Jumper Storage

Board - Paris Air Hero 32 / Acro Naze 32 Mini

This board uses the same firmware as the Naze32 board.

Sensors

MPU6500 via SPI interface.

Ports

6 x 3pin ESC / Servo outputs 1 x 8pin JST connector (PPM/PWM/UART2) 1 x 4pin JST connector (UART3/I2C)

Pinouts

The 10 pin RC I/O connector has the following pinouts when used in RX*PPM/RX*SERIAL mode.

From right to left when looking at the socket from the edge of the board.

Pin	Function	Notes
1	Ground	
2	+5V	
3	RX_PPM	Enable `feature RX_PPM`
4	RSSI_ADC	Enable `feature RSSI_ADC`. Connect to the output of a PWM-RSSI conditioner, 0v-3.3v input
5	USART2 TX	
6	USART2 RX	Built-in inverter
7	LED_STRIP	Enable `feature LED_STRIP`
8	unused	

When SOFTSERIAL is enabled, LED*STRIP and CURRENT*METER are unavailable, but one SoftSerial port is made available to use instead.

Pin	Function	Notes
7	SOFTSERIAL1 RX	Enable `feature SOFTSERIAL`
8	SOFTSERIAL1 TX	

Serial Ports

Value	Identifier	RX	TX	Notes
1	USART1	RX / PA10	TX / PA9 / TELEM	TELEM output is always inverted (for FrSky). Internally connected to USB port via CP2102 IC
2	USART2	RC4 / PA3	RC3 / PA2	
3	USART3	F3 / PB11	F2 / PB10	Flex port is configured as UART3 when port is configured
4	SOFTSERIAL1	RC5 / PA6	RC6 / PA7	

Board - Sparky

The Sparky is a very low cost and very powerful board.

- 3 hardware serial ports.
- Built-in serial port inverters which allows S.BUS receivers to be used without external inverters.
- USB (can be used at the same time as the serial ports).
- 10 PWM outputs.
- Dedicated PPM/SerialRX input pin.
- MPU9150 I2C Acc/Gyro/Mag
- Baro

Tested with revision 1 & 2 boards.

TODO

- Sonar
- Display (via Flex port)
- SoftSerial - though having 3 hardware serial ports makes it a little redundant.
- Airplane PWM mappings.

Voltage and current monitoring (ADC support)

Voltage monitoring is possible when enabled via PWM9 pin and current can be monitored via PWM8 pin. The voltage divider and current sensor need to be connected externally. The vbatscale cli parameter need to be adjusted to fit the sensor specification. For more details regarding the sensor hardware you can check here: https://github.com/TauLabs/TauLabs/wiki/User-Guide:-Battery-Configuration

Flashing

Via Device Firmware Upload (DFU, USB) - Windows

These instructions are for flashing the Sparky board under Windows using DfuSE. Credits go to Thomas Shue (Full video of the below steps can be found here: https://www.youtube.com/watch?v=I4yHiRVRY94)

Required Software: DfuSE Version 3.0.2 (latest version 3.0.4 causes errors): http://code.google.com/p/multipilot32/downloads/detail?name=DfuSe.rar STM VCP Driver 1.4.0: http://www.st.com/web/en/catalog/tools/PF257938

A binary file is required for DFU, not a .hex file. If one is not included in the release then build one as follows.

```
Unpack DfuSE and the STM VCP Drivers into a folder on your Hardrive
Download the latest Sparky release (cleanflight_SPARKY.hex) from:
https://github.com/cleanflight/cleanflight/releases and store it on your Hardrive

In your DfuSE folder go to BIN and start DfuFileMgr.exe
Select: "I want to GENERATE a DFUfile from S19,HEX or BIN files" press OK
Press: "S19 or Hex.."
Go to the folder where you saved the cleanflight_SPARKY.hex file, select it  and press open
(you might need to change the filetype in the DfuSE explorer window to "hex Files (*.hex)" to b
e able to see the file)
Press: "Generate" and select the .dfu output file and location
If all worked well you should see " Success for 'Image for lternate Setting 00 (ST..)'!"
```

Put the device into DFU mode by powering on the sparky with the bootloader pins temporarily bridged. The only light that should come on is the blue PWR led.

Check the windows device manager to make sure the board is recognized correctly. It should show up as "STM Device in DFU mode" under Universal Serial Bus Controllers

If it shows up as "STMicroelectronics Virtual COM" under Ports (COM & LPT) instead then the board is not in DFU mode. Disconnect the board, short the bootloader pins again while connecting the board.

If the board shows up as "STM 32 Bootloader" device in the device manager, the drivers need to be updated manually. Select the device in the device manager, press "update drivers", select "manual update drivers" and choose the location where you extracted the STM VCP Drivers, select "let me choose which driver to install". You shoud now be able to select either the STM32 Bootloader driver or the STM in DFU mode driver. Select the later and install.

Then flash the binary as below.

```
In your DfuSE folder go to BIN and start DfuSeDemo.exe
Select the Sparky Board (STM in DFU Mode) from the Available DFU and compatible HID Devices dro
p down list
Press "Choose.." at the bootom of the window and select the .dfu file created in the previous s
tep
"File correctly loaded" should appear in the status bar
Press "Upgrade" and confirm with "Yes"
The status bar will show the upload progress and confirm that the upload is complete at the end
```

Disconnect and reconnect the board from USB and continue to configure it via the Cleanflight configurator as per normal

Via Device Firmware Upload (DFU, USB) - Mac OS X / Linux

These instructions are for dfu-util, tested using dfu-util 0.7 for OSX from the OpenTX project.

http://www.open-tx.org/2013/07/15/dfu-util-07-for-mac-taranis-flashing-utility/

A binary file is required for DFU, not a .hex file. If one is not included in the release then build one as follows.

```
make TARGET=SPARKY clean
make TARGET=SPARKY binary
```

Put the device into DFU mode by powering on the sparky with the bootloader pins temporarily bridged. The only light that should come on is the blue PWR led.

Run 'dfu-util -l' to make sure the device is listed, as below.

```
$ dfu-util -l
dfu-util 0.7

Copyright 2005-2008 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2012 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to dfu-util@lists.gnumonks.org
```

```
Found DFU: [0483:df11] devnum=0, cfg=1, intf=0, alt=0, name="@Internal Flash  /0x08000000/128*0
002Kg"
Found DFU: [0483:df11] devnum=0, cfg=1, intf=0, alt=1, name="@Option Bytes  /0x1FFFF800/01*016
e"
```

Then flash the binary as below.

```
dfu-util -D obj/cleanflight_SPARKY.bin --alt 0 -R -s 0x08000000
```

The output should be similar to this:

```
dfu-util 0.7

Copyright 2005-2008 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2012 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to dfu-util@lists.gnumonks.org

Opening DFU capable USB device... ID 0483:df11
Run-time device DFU version 011a
Found DFU: [0483:df11] devnum=0, cfg=1, intf=0, alt=0, name="@Internal Flash  /0x08000000/128*0
002Kg"
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuERROR, status = 10
dfuERROR, clearing status
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 011a
Device returned transfer size 2048
No valid DFU suffix signature
Warning: File has no DFU suffix
DfuSe interface name: "Internal Flash  "
Downloading to address = 0x08000000, size = 76764
......................................
File downloaded successfully
can't detach
Resetting USB to switch back to runtime mode
```

On Linux you might want to take care that the modemmanager isn't trying to use your sparky as modem getting it into bootloader mode while doing so. In doubt you probably want to uninstall it. It could also be good idea to get udev fixed. It looks like teensy did just that -> http://www.pjrc.com/teensy/49-teensy.rules (untested)

To make a full chip erase you can use a file created by

```
dd if=/dev/zero of=zero.bin bs=1 count=262144
```

This can be used by dfu-util.

Via SWD

On the bottom of the board there is an SWD header socket onto switch a JST-SH connector can be soldered. Once you have SWD connected you can use the st-link or j-link tools to flash a binary.

See Sparky schematic for CONN2 pinouts.

TauLabs bootloader

Flashing cleanflight will erase the TauLabs bootloader, this is not a problem and can easily be restored using the st flashloader tool.

Serial Ports

Value	Identifier	RX	TX	Notes
1	USB VCP	RX (USB)	TX (USB)	
2	USART1	RX / PB7	TX / PB6	Conn1 / Flexi Port.
3	USART2	RX / PA3	PWM6 / PA2	On RX is on INPUT header. Best port for Serial RX input
4	USART3	RX / PB11	TX / PB10	RX/TX is on one end of the 6-pin header about the PWM outputs.

USB VCP *can* be used at the same time as other serial ports (unlike Naze32).

All USART ports all support automatic hardware inversion which allows direct connection of serial rx receivers like the FrSky X4RSB - no external inverter needed.

Battery Monitoring Connections

Pin	Signal	Function
PWM9	PA4	Battery Voltage
PWM8	PA7	Current Meter

Voltage Monitoring

The Sparky has no battery divider cricuit, PWM9 has an inline 10k resistor which has to be factored into the resistor calculations. The divider circuit should eventally create a voltage between 0v and 3.3v (MAX) at the MCU input pin.

WARNING: Double check the output of your voltage divider using a voltmeter *before* connecting to the FC.

Example Circuit

For a 3Cell battery divider the following circuit works:

```
Battery (+) ---< R1 >--- PWM9 ---< R2 >--- Battery (-)
```

- R1 = 8k2 (Grey Red Red)
- R2 = 2k0 (Red Black Red)

This gives a 2.2k for an 11.2v battery. The `vbat_scale` for this divider should be set around `52`.

Current Monitoring

Connect a current sensor to PWM8/PA7 that gives a range between 0v and 3.3v out (MAX). # Board - RMDO

The DoDo board is a clone of the SPRacingF3 board in terms of CPU pin mappings. See the SPRacingF3 documentation.

Hardware differences compared to SPRacingF3 are as follows:

- Rev 1 and Rev 2: the CPU is the cheaper version of the F3 with only 128KB FLASH. Rev 3: the CPU is a F3 version with 256KB FLASH.
- The external flash rom is the same size as found on the Naze32 (2MBit)
- The barometer is the cheaper BMP280.
- It does not have any compass sensor.
- Onboard BEC.
- Different physical connectors/pins/pads/ports. # Board - SPRacingF3

The Seriously Pro Racing MOF3 board (SPRacingF3) is the first board designed specifically for Cleanflight.

Full details available on the website, here:

http://seriouslypro.com/spracingf3

Hardware Features

- No compromise I/O. Use all the features all the time; e.g. Connect your OSD + SmartPort + SBus + GPS + LED Strip + Battery Monitoring + Sonar + 8 motors - all at the same time!
- On-board high-capacity black box flight log recorder - optimize your tuning and see the results of your setup without guesswork. (Acro and Deluxe)
- Next-generation STM32 F3 processor with hardware floating point unit for efficient flight calculations and faster ARM-Cortex M4 core.
- Stackable design - perfect for integrating with OSDs and power distribution boards.
- 16 PWM I/O lines for ESCs, Servos and legacy receivers. 8 available on standard pin headers. 8 via side mounted connectors.
- Supports SBus, SumH, SumD, Spektrum1024/2048, XBus, PPM, PWM receivers. No external inverters required (built-in).
- Dedicated output for programmable LEDs - great for orientation, racing and night flying.
- Dedicated I2C port for connection of OLED display without needing flight battery.
- Battery monitoring ports for voltage and current.
- Buzzer port for audible warnings and notifications.
- Solder pads in addition to connectors for Sonar, PPM, RSSI, Current, GPIO, LED Strip, 3.3v,
- Developer friendly debugging port (SWD) and boot mode selection, unbrickable bootloader.
- Symmetrical design for a super tidy wiring.
- Wire up using using pin headers, JST-SH sockets or solder pads. Use either right-angled or straight pin-headers.
- Barometer mounted on the bottom of the board for easy wind isolation.

Serial Ports

Value	Identifier	RX	TX	5v Tolerant	Notes
1	USART1	PA10	PA9	YES	Internally connected to USB port via CP2102 IC. Also available on a USART1 JST connector and on through hole pins.
2	USART2	PA15	PA14	YES	Available on USART2 JST port only.
3	USART3	PB11 / IO2_3	PB10 / IO2_4	NO	Available on IO_2, USART3 JST port and through hole pins.

- You cannot use SWD and USART2 at the same time.
- You may encounter flashing problems if you have something connected to the USART1 RX/TX pins. Power other devices of and/or disconnect them.

Pinouts

Full pinout details are available in the manual, here:

http://seriouslypro.com/spracingf3#manual

IO_1

The 8 pin IO*1 connector has the following pinouts when used in RX* PARALLEL_PWM mode.

Pin	Function	Notes
1	Ground	
2	VCC_IN	Voltage as-supplied by BEC.
3	RC_CH1	
4	RC_CH2	
5	RC_CH5	
6	RC_CH6	
7	LED_STRIP	Enable `feature LED_STRIP`
8	VCC	3.3v output for LOW CURRENT application only

When RX*PPM/RX*SERIAL is used the IO_1 pinout is as follows.

Pin	Function	Notes
1	Ground	
2	VCC_IN	Voltage as-supplied by BEC.
3	RX_PPM	Enable `feature RX_PPM`
4	GPIO	
5	SoftSerial1_RX	
6	SoftSerial1_TX	
7	LED_STRIP	Enable `feature LED_STRIP`
8	VCC	3.3v output for LOW CURRENT application only

IO_2

The 8 pin IO2 *connector has the following pinouts when used in RX* PARALLEL_PWM mode.

Pin	Function	Notes
1	Ground	
2	VCC_IN	Voltage as-supplied by BEC.
3	RC_CH3	
4	RC_CH4	
5	RC*CH7/SONAR*TRIG	
6	RC*CH8/SONAR*ECHO	
7	ADC_1	Current Sensor
8	ADC_2	RSSI

When RX*PPM/RX*SERIAL is used the IO_2 pinout is as follows.

Pin	Function	Notes
1	Ground	
2	VCC_IN	Voltage as-supplied by BEC.
3	RX_SERIAL	UART3 RX
4		UART3_TX
5	SONAR*TRIG/SoftSerial2*RX	Enable `feature SONAR/SOFTSERIAL`
6	SONAR*ECHO/SoftSerial2*TX	Enable `feature SONAR/SOFTSERIAL`
7	ADC_1	Current Sensor
8	ADC_2	RSSI

UART1/2/3

Pin	Function	Notes
1	Ground	
2	VCC_IN	Voltage as-supplied by BEC.
3	TXD	
4	RXD	

I2C

Pin	Function	Notes
1	Ground	
2	5.0v	Voltage as-supplied by BEC OR USB, always on
3	SCL	
4	SDA	

SWD

The port cannot be used at the same time as UART2.

Pin	Function	Notes
1	Ground	
2	NRST	Voltage as-supplied by BEC OR USB, always on
3	SWDIO	
4	SWDCLK	