DShot RPM Filtering

Recent Announcements

- Bluejay is a new, free, well-supported BlHeli-S firmware that supports DShot telemetry, with a range of options. It is most easily flashed using the https://esc-configurator.com online ESC configurator, which can flash both BlHeli-S and Bluejay to BLHeli-S ESCs, and AM32 to BlHeli-32 ESCs.
- JESC was the first ESC firmware to support RPM filtering on BLHeli_S escs. It was written by the JoeLucid, who wrote the DShot telemetry code that underpins all bidirectional DShot functionality, such as Dynamic Idle and RPM Filtering. JESC is free on L ESCs, but payment is required for H ESCs. 48khz and 96khz PWM versions are available.
- BlHeli-32 fully supports Bidirectional DShot in versions 32.7.0 and higher
- AM32 can also be flashed (but not readily un-flashed) to BlHeli-32 type ESCs, and also fully supports Bidirectional DShot.

Introduction

Bidirectional DShot Telemetry allows the ESC to report RPM data back to the flight controller over the same single wire that we use to control the ESC.

When the Flight Controller knows the RPM of each motor, it can set notch filters at exactly the right frequency to remove that noise. We can also use that RPM information to dynamically control idle RPM, and prevent it dropping too low. This is the Dynamic Idle feature, which gives better protection against ESC desync than conventional idle.

RPM based filtering gives better control over motor RPM related noise than was possible previously. Motors are cooler, bent prop tolerance is better, and full throttle sounds cleaner and often is faster.

With RPM filtering, the dynamic notch filter can be narrower, reducing delay, and since it doesn't need to track motor rpm related noise, is now better at eliminating frame resonances. Configurator automatically makes these adjustments when RPM filtering is enabled or disabled.

On clean quads, lowpass filter delay can usually be improved by moving cutoff frequencies higher or disabling some gyro filtering once RPM filtering is enabled. This should be done carefully, after reading the tuning guide.

Underlying technology:

Bidirectional DShot, a new feature in Betaflight 4.x which lets the flight controller receive accurate RPM telemetry over each motor's ESC signal line. No additional wiring or additional telemetry back-channel is needed. Each DShot frame from the FC gets acknowledged by a frame from the ESC containing the current eRPM. The FC uses the motor pole count to convert ERPM to RPM.

RPM filtering is a bank of 36 notch filters on gyro and (optionally) on Dterm which use the RPM telemetry data to remove motor noise with surgical precision. By default it runs 12 notch filters each on pitch, roll, and yaw, covering the first 3 harmonics of each motor's noise signature.

Extended DShot telemetry standardises the format used for RPM telemetry into an extensible telemetry protocol which can be used for ESC voltage, temperature and other sensors where a dedicated ESC UART connection is not available. The RPM telemetry is typically sent most frequently, with other fields sent at less frequent intervals. EDT v1.0 adds voltage, current and temperature telemetry to the basic RPM data. EDT v2.0 adds status data to signal demag, desync and stall events as well as a max demag metric.

For RPM Filtering to work, the ESC must support the Bidirectional DShot protocol and Bidirectional DShot must be enabled in the CLI.

These two features are supported by BetaFlight 4.1 and higher on all flight controllers, and most modern BLHeli_32 and BLHeli-S ESCs. Betaflight 4.0 is no longer supported.

Here's a demo in flight. Quad has minimal filtering other than the rpm filter, handles very well and shows close to no prop wash: https://youtu.be/jwFYaGHp91c, https://youtu.be/SoG245vmaLo

Arming Prevention Check

If the RPM Filter is enabled but one or more of the ESC's are not supplying valid telemetry data, arming will be prevented, and the `RPMFILTER` message will be shown on the OSD (see Arming Sequence & Safety). This prevents arming with incomplete or non-working configurations that may result in flyaways or hot motors. See below to ensure the ESC's are properly configured to support this feature.

ESC Configuration

Your ESC must support DShot, and be running a suitable firmware to support RPM filtering. At the minimum RPM telemetry is required, so the ESC must support bidirectional DShot. As EDT extends bidirectional DShot any ESC supporting EDT also support RPM telemetry.

Betaflight Configuration

Motor Poles

When running 8k8k, choose DShot600. The ESCs report eRPM. This must be converted to RPM using the number of magnets of the motors. The magnets to count are those on the bell of the motor. Do not count the stators where the windings are located. Typical 5" motors have 14 magnets, so that is the default setting. Smaller motors have fewer magnets, often 12. Count them or look up the motor specs. If you don't have 14 magnets, change the number of magnets using the Betaflight App on the Configuration tab.

DShot150, DShot300 or DShot600?

For 4k PID loops, eg 8k4k or 4k4k, or 3.2k PID loops, use DShot 300 for greatest reliability; DShot 600 may be used but provides no advantage and is more susceptible to external noise.

For 8k8k setups, you should use DShot600. With 8k PID loops, DShot300 will only update the motors every second PID loop.

With the older L ESCs (efm8bb1), DShot150 and a 2k PID loop time (8k2k) are strongly recommended as the MCU speed is too slow to reliably accept data at faster rates.

Config Snippet

With 4.1 and above it's no longer necessary to install a snippet. Instead just use the Betaflight App and enable bidirectional DShot on the Configuration tab.

Config Verification

Your FC is now set up for bidirectional DShot - let's verify that it works. To do so power cycle FC and ESC. Connect the LIPO first to the ESC, then the USB cable. Now open the Motors tab in the Betaflight App. There should be no red line indicating significant errors on any motor. When you spin the motors you should see

the reported rpm. The reported error percentage should not exceed 1%. All motors should report an RPM of 0 unless spun.

Important: If you connect your FC via USB cable without connecting your LIPO battery, then at the Motors tab in the Betaflight App you will notice an invalid indication "Error 100%" (E: 100%). Connect the LIPO and wait ESC to initialize, the indication will drop down to 0% (E: 0%). Disconnecting the battery will keep showing 0% errors afterwards.

Tuning

The RPM filter will do the heavy lifting of removing nearly all motor noise without adding much latency.

Typically no 'tuning' is required.

Although RPM filtering will remove nearly all motor noise, general 'junk' noise from bearings, wind and turbulence will not be removed by the RPM filters. Lowpass filtering will still be required to control those noise elements.

Frame or prop resonances, which manifest as large fixed frequency noise lines, won't be removed by the RPM filters either; they are best managed by keeping the Dynamic Notch active.

The Dynamic Notch needs to be reconfigured when RPM filtering is active, since it now no longer needs to eliminate motor noise. In 4.1, set the Dynamic Notch Filter Range to Medium, the Dynamic Notch Width Percent to 0 and the Dynamic Notch Q to 250. In 4.2 and higher, set the Dynamic Notch Count to 1 and Q to 500. In 4.3 Configurator will make these changes automatically.

For existing builds that already fly well, don't change any filter settings for your first flights after enabling RPM filtering.

If your motors are running cool and sounding good, you may well be able to reduce filter delay by reducing or removing some of the other filters. You may be able to:

- move gyro lowpass filtering to higher values.
- disable gyro lowpass 1 entirely (by setting its cutoff value to zero, or switching it off).
- increasing the Q factor on the RPM filter, making the RPM notches narrower and reducing delay. Above 750 is not recommended.
- reduce the number of Dynamic Notch filters (width = 0 for 4.1, count = 1 for 4.2 and higher).
- increase the Q factor of the Dynamic Notch filter (avoid going about 750).

- convert existing biquad filters, if any, to PT1s,

Each of these actions lets more noise through the filter bank to the PIDs and then to the motors.

Filters should only be changed one at a time. After each change, always do a test hover and and a short test flight. Listen to the motors carefully, and land quickly to confirm that your motors aren't getting too hot. Then repeat with a more solid flight, and again check motor temps on landing.

Black box logging, before and after, and using PID Toolbox to look at the spectral graphs is very useful. Logging can help you decide which filters can be lifted, or aren't needed, and to check that you haven't suddenly gained too much extra noise after making a change.

It's best to save a `diff all` listing of your starting filter settings before changing anything, so you can go back.

Changing filtering in 4.1 after adding RPM filtering (not relevant to 4.2 or 4.3)

Lifting all filter values by an equal amount is probably the safest and most reliable way of reducing filter delay.

The following snippet shifts all 4.1 lowpass filters up by about 50%, reducing total delays to 2.3ms at idle and 0.9ms on full throttle:

```
# 4.1 lowpass filter set shifted up 1.5x

set gyro_lowpass_type = PT1
set gyro_lowpass_hz = 300
set dyn_lpf_gyro_min_hz = 300
set dyn_lpf_gyro_max_hz = 900
set gyro_lowpass2_type = PT1
set gyro_lowpass2_hz = 350

set dterm_lowpass_type = PT1
set dterm_lowpass_hz = 150
set dyn_lpf_dterm_min_hz = 100
set dyn_lpf_dterm_max_hz = 250
set dterm_lowpass2_type = PT1
set dterm_lowpass2_hz = 200
```

If you started with 4.1 default filters, this is modest but significant change. You should notice improved propwash handling. If, on first arming, the quad makes a grinding noise or makes strange noises and jumps up on arming or during hover, typically you've gone up to high on the filters, or you have too much D. Don't fly it like that.

The next step would be 2x defaults. Only the very cleanest quads will be ok when filters are twice as high as normal. However filter delay is now half normal, and that may help propwash significantly. Here are 2x 4.1 filter settings; their total latency is 1.7ms at idle and 0.65ms on full throttle:

```
# 4.1 lowpass filter set shifted up 2x

set gyro_lowpass_hz = 400
set dyn_lpf_gyro_max_hz = 1000
set dyn_lpf_gyro_min_hz = 400
set gyro_lowpass2_type = PT1
set gyro_lowpass2_hz = 500

set dterm_lowpass_type = PT1
set dterm_lowpass_hz = 200
set dyn_lpf_dterm_max_hz = 340
set dyn_lpf_dterm_min_hz = 140
set dterm_lowpass2_type = PT1
set dterm_lowpass2_hz = 300
```

Disabling low-pass filters completely

To disable lowpass filters completely, without cooking your motors, you are going to need a mechanically sound build that doesn't have de-laminated arms, loose bolts, a bad gyro chip or noisy motor bearings.

Disabling filters have fairly substantial effects and should only be done by people familiar with filtering. Generally it is safer to move filters up or down as a group, as described above.

The following will disable the first gyro lowpass completely:

```
set gyro_lowpass_hz = 0
set dyn_lpf_gyro_max_hz = 0
```

We do not recommend disabling both gyro lowpass 1 and gyro lowpass 2 unless you are certain that the build is really clean.

This will disable the second gyro lowpass completely:

```
set gyro_lowpass2_hz = 0
```

Disabling D filters completely is not recommended. A minimum of two PT1 filters (as per the defaults) or one second order lowpass (biquad or PT2) are required by all quads. Reducing D filtering is very hazardous. It can be very easy to smoke your motors. If you do want to experiment with reduced D filtering, do so very cautiously.

Configuring the Dynamic Notch with RPM filtering

In 4.1 and 4.2 there is a need to manually re-configure the Dynamic Notch to the narrower single notch configuration, saving filter delay time. In 4.3 the appropriate changes are actioned automatically when you enable or disable RPM filtering, greatly simplifying RPM filter setup.

The following notes apply to 4.1 only:

In 4.1, the Dynamic Notch range is, by default, set to AUTO mode, and uses the value of dyn_lpf_gyro_max_hz to select the frequency range over which the dynamic notch will operate.

Because frame resonances usually happen in the LOW or MEDIUM dynamic notch range, it is best, when rpm filtering is used, to manually configure the Dynamic Notch on one of those ranges. If you know you have no frame resonances below 150hz, choosing MEDIUM will reduce delay and keep the dynamic notch higher.

If you want the dynamic notch to stay above a certain minimum value, because you know you have no resonances below that value, set the dynamic notch minimum frequency accordingly (just below the resonance that you are targeting).

For example, this snippet would let the dynamic notch run between 90 to about 330hz with a narrow single notch:

```
set dyn_notch_range = LOW
set dyn_notch_width_percent = 0
set dyn_notch_q = 200
set dyn_notch_min_hz = 90
```

This snippet would let the dynamic notch range from 180 to about 550hz with a narrow single notch:

```
set dyn_notch_range = MEDIUM
set dyn_notch_width_percent = 0
set dyn_notch_q = 200
set dyn_notch_min_hz = 180
```

If the resonance line is very narrow, you can likely get away with the dynamic notch Q of 250. Wider or more diffuse resonance bands may need a Q of 120-150.

It is always best to compare spectrums with dynamic filter on or off to check exactly what it is doing.

It is possible to turn the dynamic notch filter off altogether on stiff resonance-free builds, but this should be done with caution, and is best tested by before and after

logging confirms that it is not contributing meaningfully to the end result.

Advanced Topics

Timer Based Bidirectional DShot

If your FC supports it you can use our timer based bidirectional DShot implementation to lower the cpu load of bidirectional DShot a bit. Prior to 4.1 RC1 this was the only available implementation. Its downside is that it requires remapping of timers and DMA channels on some boards and does not work everywhere.

Don't be discouraged if your target isn't listed. Many targets will work. Use this Default Snippet, try it out and report back.

Starting from 4.1 RC1 you need to disable the default DShot bitbang implementation using the command `set dshot_bitbang=off`. Don't forget to set this back to `auto` if you want to switch back to DShot bitbang.

Snippets for supported targets

Target	Install Snippet	Notes	Supported Motors
AG3XF4	Snippet		M1 - M4 (tested Mister_M)
AIKONF4	Snippet		M1 - M4 (tested fujin)
AIRBOTF7	Snippet		M1-M4 (tested by Asizon)
ALIENFLIGHTNGF7	Snippet	M3 doesn't work, use one of M5-9 instead. LED doesn't work with M1	M1-M2, M4-M9
ALIENWHOOP	Snippet		M1-M4
ANYFCF7	Snippet		M1 M2 M3 M4 M5 M6 M9

Target	Install Snippet	Notes	Supported Motors
ANYFCM7	Snippet		M1 M2 M3 M4 M5 M7 M9 M10
BETAFLIGHTF4	Snippet		M1 - M4 ok (tested Balint)
CLRACINGF4	Snippet		M1-M4 ok
CLRACINGF7	Snippet	Motor 4 doesn't work. Use the LED pad instead	M1 M2 M3 M5
CRAZYBEEF4DX	Snippet		M1-M4 ok (tested Noctaro)
CRAZYBEEF4FR	Snippet		M1-M4 ok (tested joelucid)
DALRCF4	Snippet		M1-M6 (tested QuadMcFly)
DALRCF722DUAL	Snippet		M1-M6. But either M5 or M6
DYSF4PRO	Snippet		M1-M4 (tested BRadFPV)
ELINF405	Snippet		M1-M4 (tested elin-neo)
ELINF722	Snippet		M1-M4 (tested elin-neo)
EXF722DUAL	Snippet		M1-M8
FLYWOOF7DUAL	Snippet		M1-M6
FORTINIF4	Snippet		M1-M4(tested QuadMcFly)

Target	Install Snippet	Notes	Supported Motors
FOXEERF722DUAL	Snippet		M1-M6
FURYF4	Snippet		M1-M4, No LED support, Tested RawFPV
FURYF7	Snippet		M1-M4
HAKRCF722	Snippet		M1-M6
KAKUTEF4V2	Snippet		M1-M4 tested
KISSFCV2F7	Snippet		M1-M6
LUXF4OSD	Snippet		M1-M4 tested (Mister_M)
MAMBAF411	Snippet		DMA & timer reviewed by joelucid
MAMBAF722	Snippet		M1-M4 tested (kc10kevin)
MATEKF405	Snippet		M1-M4 tested (Wudz_17)
MATEKF411	Snippet		DMA & timer reviewed by joelucid
MATEKF411RX	Snippet		DMA & timer reviewed by joelucid
MATEKF722	Snippet		M1-M8
MATEKF722SE	Snippet	M5 does not work	M1-M4, M6-M8
MLTEMPF4	Snippet		M1-M4 (tested RC2 monko760)

Target	Install Snippet	Notes	Supported Motors
NERO	Snippet		M1-M8
NOX	Snippet		M1-M4
NUCLEOF7	Snippet	M4 does not work but can be replaced with M6	M1-M3,M6
NUCLEOF722	Snippet	M4 does not work but can be replaced with M6	M1-M3,M6
OMNIBUSF4	Snippet		M1-M4 (tested omerco)
OMNIBUSF4SD	Snippet		M1-M4 (tested joe lucid)
OMNIBUSF4FW	Snippet		M1-M4 tested (skonk)
OMNIBUSF7	Snippet		M1-M4 (tested in RC2 lgguT)
OMNINXTF7	Snippet		M1-M4
PYRODRONEF4	Snippet		M1-M4 (tested fujin)
REVOLTOSD	Snippet		M1-M4 (tested JayBird)
SPEEDYBEEF4	Snippet		Flavoredcrayon
SPRACINGF7DUAL	Snippet		M1-M10
TMOTORF4		M4-M5 does not work but can be replaced with M6	M1-M3, M6

Target	Install Snippet	Notes	Supported Motors
YUPIF7	Snippet		M1-M6

Please add additional verified configurations here.

Unsupported targets

Target	Notes

Below is a description of the settings in the config snippets.

Loop times and DShot protocol

Bidirectional DShot works with DShot 300, 600 and 1200, and also with Proshot 1000. For practical purposes, DShot 600 works well at all PID loop rates.

Remember that for each frame sent there will now be a frame coming back, and between input and output frames there is a period of 30us to switch the line, DMA, and timers. The loop time selection needs to be low enough that given the DShot protocol rate both frames + 50 us fit into one gyro loop iteration.

Both bidirectional DShot and the RPM filter are fairly CPU intensive and it is very important for the loop rates to be exactly on spot so that the filters get tuned to the right frequencies. It is recommended to run at 4k/4k with DShot 600 for your initial test flight. All DShot speeds should work at that loop rate.

On F4, RPM telemetry costs about 3-4uS per motor per line direction change. Something around 24-32uS is required for the line direction switching both directions together. The RPM filter has 36 notch filters that get dynamically tuned at 1000Hz update frequency.

8k4k is good on most F405's; 8k8k is possible, but often needs overclocking. F411's will require over-clocking to run at 8k4k. F7's run 8k8k without problems.

You may need to switch off any extended startup melody, since that may interfere with bidirectional DShot. The standard startup tones are fine.

DMA

The current implementation requires normal DMA to be used, not burst DMA. Turning burst DMA off may, of itself, not work with a given FC. You can try it out in

advance:

```
set dshot_burst = OFF
```

And test whether your quad still flies. If so proceed to the next step:

Enabling new scheduler policy

Since the RPM filter works with very narrow notch filters, it's imperative that the gyro loop time does not vary and is exactly as specified. This used to require low loop rates and overclocking. A scheduler change has now been added, which allows consistent gyro rates even at higher loop rates. Bidirectional DShot requires enabling this feature:

```
set scheduler_optimize_rate = ON
```

Enabling Bidirectional DShot

```
set dshot_bidir = ON
```

See if your motors still spin up. If so try detach the USB cable, connect a battery and reconnect USB. Now go to the CLI and type `status`. You should see DShot telemetry being reported. The reported RPM should be zero for each motor and there should be few errors.

Motor Poles

The ESCs report eRPM, which needs to be converted to RPM using the number of poles (magnets) of the motors. These are found on the bell of the motor, not the stator magnets where the windings are located. Typical 5" motors have 14 poles, so that is the default setting. Smaller motors have fewer poles, often 12. Count them or look up the motor specs and configure using:

```
set motor_poles = 14
```

Verifying consistent loop time

Important:

After enabling all of the above features double check that your loop rate is consistent. If not select a lower loop rate. Remember that unlike effective filtering, loop time has very minimal effect on flight performance.

To do so enter `tasks` in the CLI and check that the gyro rate matches what you have specified (Note: Battery should be connected to FC to get accurate loop rate

results). For example:

```
# tasks
Task list             rate/hz  max/us  avg/us maxload avgload     total/ms
00 - (          SYSTEM)     10       1       0    0.5%    0.0%         0
01 - (          SYSTEM)   1000       3       1    0.8%    0.6%       522
02 - (        GYRO/PID)   7999      43      34   34.8%   27.6%      2845
03 - (             ACC)   1000      12      10    1.7%    1.5%       107
04 - (        ATTITUDE)    100      17      10    0.6%    0.6%        11
05 - (              RX)     32      34      32    0.6%    0.6%        12
06 - (          SERIAL)    100     851       3    9.0%    0.5%         8
08 - (BATTERY_VOLTAGE)     50       4       2    0.5%    0.5%         1
09 - (BATTERY_CURRENT)     50       1       1    0.5%    0.5%         0
10 - ( BATTERY_ALERTS)      5       3       2    0.5%    0.5%         0
11 - (          BEEPER)    100      2       1    0.5%    0.5%         1
14 - (            BARO)     43      98      66    0.9%    0.7%        34
15 - (        ALTITUDE)     40       7       3    0.5%    0.5%         1
17 - (       TELEMETRY)    250       1       0    0.5%    0.0%        27
19 - (             OSD)     60      21      13    0.6%    0.5%         9
21 - (             CMS)     60       1       1    0.5%    0.5%         0
22 - (         VTXCTRL)      5       1       1    0.5%    0.5%         0
23 - (         CAMCTRL)      5       1       1    0.5%    0.5%         0
25 - (     ADCINTERNAL)      2       3       1    0.5%    0.5%         0
26 - (        PINIOBOX)     19       1       1    0.5%    0.5%         0
RX Check Function                     2       1                         0
Total (excluding SERIAL)                              46.0%   37.1%
```

You need to check the *GYRO/PID* line:

```
02 - (        GYRO/PID)   7999      43      34   34.8%   27.6%      2845
```

In this case we have the Gyro/PID configured in 8k/8k and this line show us that it is executing at a rate of 7999Hz. This must be very, very close to the 8k value (8000Hz). A recommendation will be maintain the error under the 1%.

Debug modes

There are two blackbox debug modes to verify the RPM filter: RPM_FILTER logs the frequency of each motor as reported by the ESC. DSHOT_RPM_TELEMETRY logs the unconverted eRPM.

TPA

This snippet will set a 4.0x build to the 4.1 default of 65% D only TPA starting at 1250

```
set tpa_rate = 65
set tpa_breakpoint = 1250
```

References

Bidirectional DShot PR

Rpm Filter PR

Youtube

Joshua Bardwell

Ivan Efimov

Links

Oscar Liang

VitroidFPV

Community

Discord

IntoFPV

Feeds

RSS

Atom

JSON

Built with Docusaurus

made with ❤️ by **VitroidFPV** and **un!t**