

Betaflight / Deutsch

Dies hier ist eine Sammlung an Dokumenten die im Laufe meiner kurzen FPV-Fliegerei entstanden sind und auch weiter entstehen.

Hintergrund für diese Ansammlung der Dokumente ist, dass ich mehr Verständnis für mich selbst schaffen möchte und dadurch ein besseres Verständnis über die Zusammenhänge und dem Flugverhalten von FPV-Coptern zu erhalten.

Weiterhin möchte ich kein endloses Wiki in deutsch erstellen sondern einzelne Themenbereiche herauslösen - auch wenn es dadurch Redundanzen gibt. Für mich hat es den Vorteil, dass ich bestimmte Dokumente ausdrucken kann und mit zum Fliegen nehmen kann, wenn ich einen Copter optimieren möchte.

Tja und zum Schluß - ich kann mir einfach nicht alles im Kopf behalten, daher lese ich gerne nach und dann helfen mir kleinere Dokumente mehr als große.

Die einzelnen Seiten stehen auch zum Download als PDF zur Verfügung.

Alternative: Downloaded dieses Repository komplett und startet über Euren Browser `index.md`

Vielleicht hilft es dem ein oder Anderen

Du kannst alle Seiten als PDF downloaden über Komplette PDF-Datei ([site/pdf/combined.pdf](https://github.com/GrumpyPete/Betaflight-Deutsch/blob/master/site/pdf/combined.pdf))

Viel Spass und propwash freies Fliegen

LunaX - August 2020

Handhabung dieses Github-Repositories

Dies Repository basiert auf MKDocs (<https://mkdocs.org/>). Alle geschriebenen Seiten wurden als einfache Markdown-Dateien erstellt.

Komplette PDF-Datei ([../site/pdf/combined.pdf](https://github.com/GrumpyPete/Betaflight-Deutsch/blob/master/site/pdf/combined.pdf))

```
mkdocs.yml (Konfigurationsdatei für MKDocs)
docs/
  xxxx.md Dateien (einzelne Markdown-Dateien zu den Themen)

sites/
  pdf/combined.pdf (hier liegt die Gesamt-PDF-Datei)
  <folder> (diverse Subfolder, enthalten HTML-Seiten zum Offline-Lesen)
```

- Betaflight / Deutsch
 - Handhabung dieses Github-Repositories

BF 4.2 - Neuerungen

Allgemeine Anpassungen

- BF-Configurator 10.7.0 ist notwendig
- Blackbox-Explorer 3.5.0 für BF 4.2
- Neue Tuning-Möglichkeiten (Slider)
- **Bei einigen Targets downgeloaded werden müssen, daher ist 10.7.0 wichtig.**
- **Keine Konfigurationen aus älteren Releases importieren** Einige Default-Werte haben sich geändert
- Es wird kein Motorprotokoll automatisch ausgewählt. Dies muss nach einem Update manuell durchgeführt werden.
- Wenn bidirektionales DShot ausgewählt wird, die Geschwindigkeit des Protokolls wird halbiert um die Bidirektionalität aufrecht zu erhalten. Dies gilt **nicht für DShot600**. Bei `DShot300` muss die PIDLoop auf 4k gesetzt werden. Bei `DShot150` auf 2k. Demnach bei `DShot600` auf 8k
- Bevor gearmt werden kann muss der Accelerometer kalibriert werden.
- Die Strommessung ist nun am Throttle-Signal gekoppelt, das ermöglicht besser Vorhersagen zum Stromverbrauch
-

Major

- Überarbeitung der Gyro-Loop. Diese läuft nun in der Geschwindigkeit der nativen Beschwindigkeit des Gyros
- zwei neue Rates auswählbar `ACTUAL` und `QUICK`
- VBat Kompensation, wenn die Batteriespannung nachläßt
- Neuer NFE Race Modus

Minor

- OSD Logo nun auch beim Armen sichtbar
- Unterstützung für verbesserte OSD-/CMS-Geräte hinzugefügt, wodurch es möglich wurde, das Highlighting von Text oder Symbolen zu unterstützen
- Unterstützung für FrSkyOSD OSD-Geräte hinzugefügt
- Unterstützung für das Redpine RC-Protokoll auf Geräten mit einem über SPI angeschlossenen CC2500-Chip (FrSky SPI) hinzugefügt

BF 4.1 - Neuerungen

Allgemeine Anpassungen

- **Keine Konfigurationen aus älteren Releases importieren** Einige Default-Werte haben sich geändert
- BF-Configurator 10.6.0 ist notwendig
- Blackbox-Explorer 3.4.0 für BF 4.1
- DShot für bidirektionalen Einsatz mit RPM-Filtern wurde verbessert. BLHeli_32 ab Version 32.7 ist erforderlich
- VTX-Tabellen sind nun nach einem flashen neu zu laden. Hiermit können nun die VTX deutlich besser konfiguriert werden
- Eine Anpassungen an den OSD-Fonts

Major

- **NEU** Feedforward 2.0 (<https://github.com/betaflight/betaflight/wiki/Feed-Forward-2.0>)
- Überarbeitetes bidirektionales DShot
- Dynamisches Leerlauf-Management nutzt nun RPM-Telemetrie
- voll konfigurierbare VTX über VTX-Tabellen

Minor

- Unterstützt nun Spektrum SRXL2 Protokoll
- Board spezifische Defaultwerte
- Unterstützung von willkürlichen Gyro & Magnetometer Ausrichtungen

BF 4.0 - Neuerungen

Allgemeine Anpassungen

- **Keine Konfigurationen aus älteren Releases importieren** Einige Default-Werte haben sich geändert
- BF-Configurator 10.5.1 ist notwendig
- Blackbox-Explorer 3.3.1
- BF 4.0 enthält

Major

- Echtzeit RPM feedback und Notch-Filter basieren auf RPM
- DTerm Management mit `d_min`
- Throttle basierendes dynamic gyro und DTerm Filterung
- Start Kontrolle (launch control)
- Umschaltbare OSD-Profile
- Vereinheitlichte Zielsysteme

Minor

- Kaskadierende dynamische Notch-Filter
- Schub Linearisierung
- Integrierte YAW Kontrolle
- Umschaltbare LED_Strip Profile
- Gimbal-Overlay im OSD
- Profil-Switching basierend die Anzahl der Lipo-Zellen
- Pro Profil Limitierung des maximalen Motor Outputs *

FILTER-Beschreibungen

- FILTER-Beschreibungen
- Allgemeines
 - Noise / Vibrationen
 - Harmonics
- Gyro-Data-Filtering
- Filter-Arten
 - Lowpass-Filter
 - Notch-Filter
 - RPM-Filter
 - Static-Filter
 - Dynamic-Filter
- Betaflight-Filter
 - BF - Dynamic-Lowpassfilter
 - BF - Dynamic-Notchfilter
 - BF - Static-Lowpassfilter
 - PT1
 - BIQUAD
 - BF - Gyro-RPM-Filter
 - BF - Gyro-LowPass Filter

{{TOC}}

Allgemeines

Noise / Vibrationen

Ist eine Bezeichnung für Rauschen. Also Messwerte die irreguläre sind und die Performance des Copters negativ beeinflussen. Noise (Rauschen) kann ausgelöst werden durch: * äußere Einflüsse (z.B. Wind, ...) * Vibrationen am Chassis (z.B. lose schrauben, dünne schwingende Arme, ...) * Vibrationen durch Motoren (z.B. Lagerschäden, Unwucht, ...) * Vibrationen durch Props (z.B. Unwucht, Prop-Wash, ...) * Kombinationen aus allem •.

Noise wird durch den Gyro und anschließend durch den PID- Controller verarbeitet und kann zu Fehlerhaften Verhalten führen.

Eingesetzte Filter (LPF, NOTCH, ...) versuchen dieses Rauschen zu eliminieren.

Eine Deiner Hauptaufgabe beim Tunen Deines Copters ist, dass du diese Vibrationen in den Griff bekommst ohne deutliche Delay zu bekommen.

Hinweis:

Der DTerm-Anteil des PID-Controllers verstärkt Vibrationen deutlich. Daher ist eine Abstimmung der Filter mit dem DTerm gut abzustimmen.

Um Vibrationen zu analysieren musst du eine Blackbox-Analyse durchführen oder Tools wie

`Blackbox-Explorer`, `PIDToolbox` oder `Plasmatree` verwenden.

Um einen ersten Eindruck von Vibrationen zu erhalten empfehle ich eine Spektral-Analyse (<https://github.com/mrRobot62/PIDtoolbox/wiki/SpectralAnalyzer>). Hier sieht man sehr deutlich in welchem Frequenzband Vibrationen an Deinem Copter auftreten.

Filter helfen, diese Vibrationen möglichst aus den Signalen für den Flight-Controller herauszufiltern um anschließend den Motoren möglichst saubere Signale zu übermitteln.

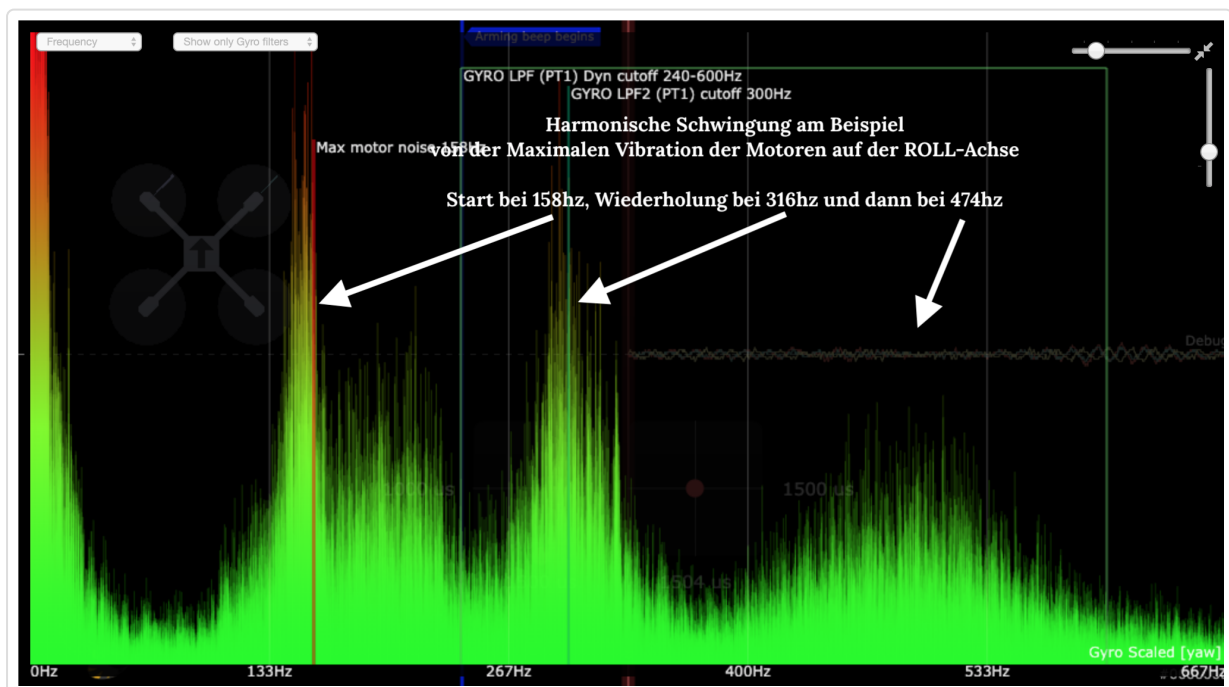
Tip

Bevor du damit startetest die PID-Werte anzupassen, stelle Deine Filter optimal auf Deinen Copter ein. Erst dann fängst du mit den PID-Werten an.

Versuche durch Deine Filter ein maximales Gyro & DTerm Delay von <5ms zu erreichen. Du kannst das sehr einfach mit `PIDToolbox` erkennen.

Harmonics

Harmonics oder Harmonische sind wiederkehrende Amplituden (Vibrationen) in den gleichen Frequenzabständen.



Gyro-Data-Filtering

```
graph LR
GY(Gyro)
GY -. -> DMGYR[[DebugMode-GyroRaw]]:::dbg
GY -. -> DMNF[[DebugMode-Notch]]:::dbg
GY -. -> DMFFT[[DebugMode-FFT]]:::dbg
DMFFT & DMNF & DMGYR -. -> DNF(DynamicNotch-Filter)
GY-->DNF
DNF -. -> DMFFT2[[DebugMode-FFT]]:::dbg
DNF --> SNF(Static Notch Filter)
DMFFT2 -. -> SNF(Static Notch Filter)

classDef dbg fill:#ddd
```

```
graph LR
SNF(Static Notch Filter)
SNF -. -> DMGY[[DebugMode-Gyro]]:::dbg
DMGY -. -> SLPF(Static LowPass Filter)
SNF --> SLPF(Static LowPass Filter)
SLPF --> PID(PID-Loop)

PID -. -> DMDT[[DebugMode - DTerm-Filter]]:::dbg
PID --> DTLPF(DTerm LowPass Filter)
DMDT -. -> DTLPF(DTerm LowPass Filter)

DTLPF --> DTNF(DTerm Notch Filter)
DTNF --> M(Motors)

classDef dbg fill:#ddd
```

Filter-Arten

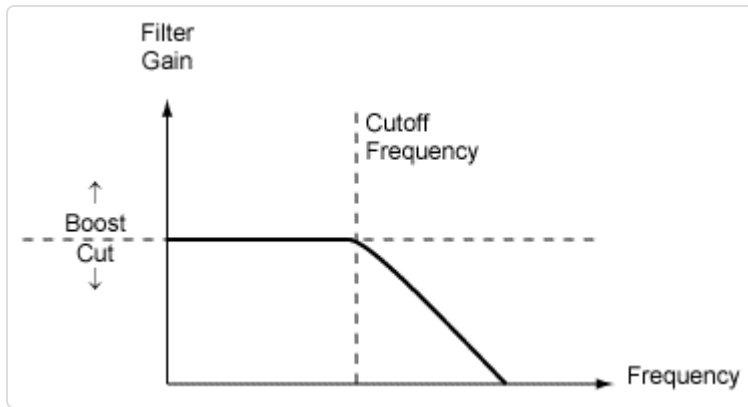
Lowpass-Filter

Niedrige Frequenzen werden durch den Filter durchgelassen, hohe Frequenzen werden gedämpft.

Hohe Frequenzen sind in der Regel im System nur Rauschen bzw. Vibrationen und werden für die Flugdaten nicht benötigt, daher versucht man sie herauszufiltern.

Mit dem LOWPASS-Filter wird eine Grenzfrequenz (**cut-off**) angelegt und der FC reduziert die Signale die oberhalb dieser Grenzfrequenz liegen.

Die Dämpfungskurve ist eine Steigung. d.h. je höher die Signalfrequenz desto stärker die Dämpfung



Für den Einsatzbereich des Quadcopters ist der Frequenzbereich von 0-80Hz relevant, alles darüber sollte möglichst effizient weg gefiltert werden.

Ein fundamentaler Aspekt ist, je niedriger der Schwellwert für den Lowpass Filter ist - umso mehr muss gefiltert werden. Das wirkt sich auf dem Gesamtperformance aus. Daher gibt es mehrere andere Filter die andere Algorithmen verwenden und das System effizienter gestalten.

Notch-Filter

Notch-Filter eignen sich hervorragend zur Unterdrückung von Rauschen in einem sehr spezifischen Frequenzband.

Notch-Filter sind in der Regel effektiver zur Reduzierung von Motorrauschen als LPF- Filter aber ggf. müssen manuelle Abstimmungen durchgeführt werden um die Bandbreite und die Mittenfrequenz zu bestimmen (Notch=Kerbe)

Weiterhin können Notch-Filter zur Reduzierung von Propwash genutzt werden.

todo bild einfügen ()

Es gibt zwei Notch-Filter für den GYRO. Einer oder beide können bei Bedarf abgestellt werden.

Default-Values sind auf 200Hz-400Hz. Diese Grenzfrequenzen arbeiten sehr gut und funktionieren bei den meisten Coptern.

Für eine Feinabstimmung ist es notwendig eine Blackbox- Auswertung durchzuführen (oder Plasma-Tree)

Beispiel:

Es wird festgestellt das wir ein Vibrationsspitze bei ca 260Hz haben (z.B. hervorgerufen durch Propwash), dann kann der Notch-Filter Cut-off bei ca. 200Hz beginnen und bei 300Hz enden

RPM-Filter

RPM-Filter basieren auf mehrere **Notch-Filter** die präzise darauf ausgerichtet sind Motorvibrationen und ihre **Harmonics** zu filtern und möglichst komplett zu eliminieren. Für jeden Motore können bis zu drei Notch-Filter (max 3 Harmonics) generiert werden (Pro Achse). Das heißt insgesamt steht eine Bank von 36 Notch-Filtern zur Verfügung. $3 \text{ Achsen} * 3 \text{ Harmonics} * 4 \text{ Motoren} = 36 \text{ Notch-Filter}$.

Die RPM-Filter basieren auf die Drehzahl der Motoren und dem bidirektionalen DShot-Protokoll in Zusammenspiel mit den Gyrodaten - daher auch `Gyro-RPM-Filter`

BEACHTEN

RPM-Filter benötigen für BLHeli32 ESC die aktuellste Firmware `>= 32.7`

Für BLHeli_S ESCs empfehle ich die FW von (auch wenn sie Geld kosten) JFLIGHT (<https://jflight.net/index.php>)

Static-Filter

Statische Filter sind an eine Frequenz gebunden

Dynamic-Filter

Dynamic Filter reduzieren ebenfalls Rauschen, wenn die entsprechenden Parameter richtig eingestellt sind. Schwingungen, Motorgeräusche, ... können durch die Dynamic-Filter reduziert werden.

Ein Dynamic-Filter ist ein Algorithmus, der die Frequenz des Rauschens erkennen kann und er kann den Notch-Filter verwenden, um ihn automatisch zu reduzieren.

Nachteil von Dynamic-Filters ist die Erhöhung der CPU-Last und die Delays.

Betaflight-Filter

BF - Dynamic-Lowpassfilter

BF - Dynamic-Notchfilter

BF - Static-Lowpassfilter

In Betaflight wird zwischen zwei Static-LPF Filtern unterschieden

- PT1
- BIQUAD

PT1

Dieser Filter hat eine etwas sanftere Kurve und ist ein LPF-Filter 1. Ordnung und hat dadurch eine geringere Latenzzeit. Der Nachteil dieses Filters, er filtert nicht so stark Vibrationen aus dem Signal (bedingt durch seine Kurvenausprägung)

BIQUAD

Dieser Filter hat eine deutliche steilere Kurve und filtert besser als ein PT1. Er ist ein Filter 2. Ordnung. Dadurch ist die Latenzzeit schlechter, das Filterergebniss besser.

BF - Gyro-RPM-Filter

Ein mächtiges neues Feature, welches mit BF 4.0 eingeführt wurde und in den nachfolgenden Releases weiter verbessert wurde. Die RPM Filter wurden weiter oben schon beschrieben, nachfolgend eine Reihe von Detailinformationen.

BF - Gyro-LowPass Filter

Documentation built with MkDocs (<http://www.mkdocs.org/>) using Windmill (<https://github.com/gristlabs/mkdocs-windmill>) theme by Grist Labs.

BF Blackbox-Analysen

Documentation built with MkDocs (<http://www.mkdocs.org/>) using Windmill (<https://github.com/gristlabs/mkdocs-windmill>)
theme by Grist Labs.

PID - Controller

Documentation built with MkDocs (<http://www.mkdocs.org/>) using Windmill (<https://github.com/gristlabs/mkdocs-windmill>)
theme by Grist Labs.

BF 4.2 Tuning Tips

Documentation built with MkDocs (<http://www.mkdocs.org/>) using Windmill (<https://github.com/gristlabs/mkdocs-windmill>)
theme by Grist Labs.

Betaflight BF4.x Tuning-Parameters



Inhaltsverzeichnis

- Betaflight BF4.x Tuning-Parameters
- Inhaltsverzeichnis
- Historie
- Tuning-Parameter
 - Betaflight - Tuning-Tips
- DSHOT RPM Telemetrie-Daten
 - Allgemeines
 - IN
 - OUT
 - Parameter
- Gyro Filter
 - Allgemeines
 - IN
 - OUT
 - Gyro Filterarten
 - Gyro Filter => GYRO-RPM Notch Filter
 - Allgemeines
 - Parameter
 - Gyro Filter => Dynamic-Notch Filter
 - Allgemeines
 - Parameter
 - Gyro Filter => Static Gyro-Notch Filter (1 und 2)
 - Allgemeines
 - Parameter
 - Gyro Filter => Dynamic Gyro LowPass Filter
 - Allgemeines
 - Parameter
 - Gyro Filter => Static Gyro LowPass Filter
 - Allgemeines
 - IN
 - Parameter
- DTerm Filter
 - Allgemeines
 - IN
 - OUT

- DTerm => Dynamic D lowpass
 - Allgemein
 - Parameter
- DTerm => Static D notch
 - AllgemeiHz
 - Parameter
- Feedforward
 - Allgemeines
 - IN
 - OUT
 - Parameter
- VBat
 - Allgemeines
 - IN
 - OUT
 - Parameter
- RC-Command
 - Allgemeines
 - IN
 - OUT
 - Parameter
 - rc_smoothing_auto_smoothness (Default:10)
- Setpoint
 - Allgemeines
 - IN
 - OUT
 - Parameter
- ITerm Parameter
 - Allgemeines

Historie

| Version | Datum | Inhalt |
|---------|-------------|------------------|
| 0.1 | August 2020 | initial |
| 0.2 | August 2020 | neu strukturiert |

Tuning-Parameter

Nachfolgende eine Reihe der wichtigsten Tuning-Variablen.

Viel mehr Details findet man hier: BF4.2-Tuning-Notes (<https://github.com/betaflight/betaflight/wiki/4.2-Tuning-Notes>)

ACHTUNG

Bei einem Update von BF <4.2 bitte **KEIN** Restore von alten Werten die durch `diff all` gespeichert wurden, importieren.
Fangt bei **NULL** an

Betaflight - Tuning-Tips

Weitere Tuning-Tips findest du im BF-Wiki der jeweiligen Versionen: * BF4.2-Tuning-Notes (<https://github.com/betaflight/betaflight/wiki/4.2-Tuning-Notes>) * BF4.1-Tuning-Notes (<https://github.com/betaflight/betaflight/wiki/4.1-Tuning-Notes>) * BF4.0-Tuning-Notes (<https://github.com/betaflight/betaflight/wiki/4.0-Tuning-Notes>) * BF4.0-Tuning-Notes (<https://github.com/betaflight/betaflight/wiki/3.5-tuning-notes>)

DSHOT RPM Telemetrie-Daten

Allgemeines

Ab BF 4.0 werden Telemetriedaten des ESCs ausgelesen und analysiert. Diese Informationen sind elementar wichtig für viele nachgelagerte Filtertechniken und für den PID-Controller.

Voraussetzung ist, dass man für den ESC ein DSHOT-Protokoll ausgewählt hat

IN

ESC - **Beachten:** die aktuelle Firmware des ESCs muss RPM-Telemetrie-Daten verarbeiten können.

OUT

RPM-Daten pro Motor

Parameter

Diese Parameter können nicht direkt beeinflusst werden. Zu beachten sind welches **DSHOT** Protokoll verwendet wird.

Bedenke: bei DSHOT300 und einer 8k PIDLoop erhältst du nur jede zweite PID-Loop Daten zugesendet. Daraus folgt, du solltest das passende **DSHOT-Protokoll** auf Deine PID-Loop auswählen

- **DSHOT150:** empfohlen bei 2k PIDLoop
- **DSHOT300:** empfohlen bei 4k PIDLoop
- **DSHOT600:** empfohlen bei 8k PIDLoop
- **DSHOT1200:** 8k PIDLoop

Gyro Filter

Allgemeines

Der Gyro ist das zentrale Bauelement auf dem FC und stellt die aktuellen **IST** Flugdaten zur Verfügung. Diese Daten werden dann bezogen auf die **SOLL** Daten (Die RC-Commands) verrechnet, gefiltert dem PIDController zur Verfügung gestellt. Anschließend gemixt und den Motoren als neue.

Jeder GYRO besitzt werkseitig einen internen LowPass-Filter. Je nach Bautyp des Gyros unterscheiden sich wie gut dieser interne Filter tatsächlich ist.

Die Gyro-Filter Parameter umfassen folgende Filterarten

Bei den weiteren beschriebenen Gyro Filtern wird nicht nochmals auf IN/OUT eingegangen.

IN

`gyro_scaled` Daten direkt aus dem Gyro.

OUT

Bereitstellung der Daten für nachgelagerte `DTerm-Filter` und als Mix-Daten für den `P-Controller` => `Vorab-Fehler P-Berechnung`

Gyro Filterarten

Es wird unterschieden zwischen den Gyro-RPM Filtern, einem Dynamic NotchFilter und Static Notchfilter 1+2, einem Dynamische LowPassfilter und einem statischen Lowpassfilter.

- Gyro-RPM-Notch Filter

`gyro_rpm_notch_harmonics=3`

`gyro_rpm_notch_q=500`

`gyro_rpm_notch_min=100`

- Dynamic-Notch Filter `dyn_notch_min_hz`

`dyn_notch_max_hz`

`dyn_notch_width_percent`

`dyn_notch_q`

- Static Gyro-Notch Filter (1 und 2) `gyro_notch1_hz`

`gyro_notch1_cutoff`

`gyro_notch2_hz`

`gyro_notch2_cutoff`

- Dynamic Gyro LowPass Filter `gyro_lowpass_type`

`gyro_lowpass_hz`

`dyn_lpf_gyro_min_hz`

`dyn_lpf_gyro_max_hz`

- Static Gyro LowPass Filter `gyro_lowpass2_type`

`gyro_lowpass2_hz`

Gyro Filter => GYRO-RPM Notch Filter

Allgemeines

Der Gyro-RPM Notch Filter nutzt die vom ESC zurückgegeben RPM-Daten und liegt als erste Filterstufe direkt hinter dem `gyro_scaled` Daten.

BF 4.1/4.2 Bidirectional DSHOT and RPM Filter Guide (<https://github.com/betaflight/betaflight/wiki/Bidirectional-DSHOT-and-RPM-Filter#Tuning>)

Parameter

| Parameter | BFD | Default | Bezeichnung |
|---------------------------------------|-----|---------|--|
| | | | Schwingungen treten in wiederkehrenden Amplituden auf. Eine harmonische Schwingung kann durch eine Sinusfunktion beschrieben werden (https://de.wikipedia.org/wiki/Schwingung#Harmonische_Schwingung). Das bedeutet, dass eine Vibration sich alle xHz wiederholt! Betaflight generiert somit pro Motor 3 (Anzahl Harmonics) Notch-Filter, somit werden alle Motordaten durch 12 individuellen RPM-Notch-Filter analysiert und schon vorgefiltert. Diesen Sachverhalt kann man in einer Blackboxauswertung sehr gut sehen (FFT-Spektrum). |
| <code>gyro_rpm_notch_harmonics</code> | 3 | | Der Q-Faktor des Notchfilters gibt die Breite der Kerbe (Notch=Kerbe) an. Je größer die Zahl (max 1000) umso schmaler wird der Notchfilter. Der <code>Q-Faktor</code> wird auch als Güte-Faktor bezeichnet. Je höher die Güte-Faktor (<code>Q-Faktor</code>) desto geringer die Dämpfung, desto schmaler der Notch-Filter. Kleine Q-Faktoren vergrößern den RPM-Filter Delay - was unerwünscht ist |
| <code>gyro_rpm_notch_q</code> | 500 | | Beschreibt die untere Grenzfrequenz des Notch-Filters |
| <code>gyro_rpm_notch_min</code> | 100 | | |

Gyro Filter => Dynamic-Notch Filter

Allgemeines

Dieser Filter ist dem RPM-Filter nachgelagert und filtern nochmals bestimmte Frequenzen aus. Ohne RPM-Filter wird der Filter als Doppel-Notch Filter betrieben, dies wird über `dyn_notch_width_percent` > 0 definiert. Ist dieser Wert 0, wird nur ein Notch-Filter erzeugt.

Das besondere an dynamic-notch-Filter ist, dass sie dynamisch sich an der aktuellen *RPM* des Systems orientieren und so laufend rund um den höchsten Frequenzbereich der Motor-Vibrationen arbeiten.

Grundsätzlich gilt, dynamische Notch-Filter haben eine geringere Latenzzeit als statische Notch-Filter. Bei einem gut abgestimmten Copter können andere Low-Pass Filter deaktiviert werden.

Parameter

| Parameter | BFDDefaultBezeichnung |
|--------------------------------------|--|
| <code>dyn_notch_min_hz</code> | Beschreibt die untere Grenzfrequenz dieses Notch-Filters in Hz |
| <code>dyn_notch_max_hz</code> | Beschreibt die obere Grenzfrequenz dieses Notch-Filters in Hz |
| <code>dyn_notch_width_percent</code> | Beschreibt (wenn > 0), wie weit beide Notch-Filter voneinander getrennt sind. Der Prozentsatz berechnet sich aus der Breite des Notch-Filters. |
| <code>dyn_notch_q</code> | Q-Faktor des Notch-Filters. (siehe hierzu Beschreibung weiter oben) |

Gyro Filter => Static Gyro-Notch Filter (1 und 2)

Allgemeines

Zwei statische Notch-Filter für ein bestimmtes Frequenzband. Dieses Frequenzband wird während des Fluges nicht mehr angepasst (statisch).

Parameter

| Parameter | BFDDefaultBezeichnung |
|---------------------------------|-----------------------------------|
| <code>gyro_notch1_hz</code> | Center-Frequenz des Notch-Filters |
| <code>gyro_notch1_cutoff</code> | todo |
| <code>gyro_notch2_hz</code> | Center-Frequenz des Notch-Filters |
| <code>gyro_notch2_cutoff</code> | todo |

Gyro Filter => Dynamic Gyro LowPass Filter

Allgemeines

Die Auswahl den Dyn-Notchfilters Frequenzbereiches kann über drei Auswahlmöglichkeiten voreingestellt werden * **LOW** : `dyn_lpf_gyro_max_hz` liegt bei 334hz oder ist 0 (deaktiviert) * **MEDIUM** : `dyn_lpf_gyro_max_hz` liegt bei 610hz * **HIGH** : `dyn_lpf_gyro_max_hz` liegt bei > 610hz

Die durchschnittlichen Werte für optimale Werte für diese Ranges liegen * **LOW** : 80-330hz (für Copter mit niedrigen Drehzahlen oder wenn Resonanzen in niedrigen Frequenzen auftreten) * **MEDIUM** : 140-550hz (für gut eingestellte 5" Copter * **HIGH** : 230-800hz (für Copter mit hohen Drehzahlen 2,5" - 3")

Ab BF 4.0 wird zusätzlicher `dyn_notch_min_hz` Parameter zur Verfügung gestellt. Dieser Wert fängt den Bereich unterhalb des Dyn-LPF ab und hat seinen Default bei 150Hz.

Um 100Hz Peaks heraus zu filtern muss `LOW` aktiviert werden und der

Parameter

| Parameter | BFDDefaultBezeichnung |
|--------------------------------|--------------------------------------|
| <code>dyn_notch_min_hz</code> | |
| <code>gyro_lowpass_type</code> | LOW/MEDIUM/HIGH (siehe Beschreibung) |

| Parameter | BFDefault | Bezeichnung |
|---------------------|-----------|--|
| gyro_lowpass_hz | | Static Gyro LPF, sind dyn_lpf gesetzt, dann ist der Static LPF deaktiviert |
| dyn_lpf_gyro_min_hz | | untere Grenzfrequenz es DynLPF |
| dyn_lpf_gyro_max_hz | | obere Grenzfrequenz des DynLPF |

Gyro Filter => Static Gyro LowPass Filter

Allgemeines

IN

Throttle-Daten

Parameter

| Parameter | BFDefault | Bezeichnung |
|--------------------|-----------|---|
| gyro_lowpass2_type | 4.0 | PT1/ BIQUAD |
| gyro_lowpass2_hz | 4.0 | unter Grenzfrequenz des LPF in Hz, wenn auf 0, dann ist der LPF deaktiviert |

DTerm Filter

Allgemeines

Der DTerm-Filter besitzt eine Reihe von Parametern die dazu genutzt werden, das DTerm-Ausgangssignal zu bearbeiten und von Störungen (Vibrations-Frequenzen zu befreien). **Wichtig::** Der DTerm des PID-Controllers verstärkt Vibrationen, daher ist es wichtig, dass dieses Signal möglichst frei von Störungs- / Vibrationssignalen ist.

D verstärkt höhere Frequenzen, der D-Anteil wird aber dringend benötigt um Vibrationen zwischen 30-80hz (Z.B. Propwash) auszugleichen. Das bedeutet wir benötigen soviel wie möglich D-Anteil bis 100hz und so wenig wie möglich über 100hz. DTerm-Filter sollten immer in der ersten Stufe als BIQUAD und in der zweiten Stufe als PT eingestellt werden.

DTerm Filter Daten sind zeitabhängig (d/dt)

Folgende DTerm-Filter werden genutzt: * **Dynamic D lowpass**

```

...
dterm_lowpass_type
dyn_lpf_dterm_min_hz
dyn_lpf_dterm_max_hz
dyn_lpf_dterm_curve_expo
...

```

- **Static D lowpass** dterm_lowpass2_type
dterm_lowpass2_hz
- **Static D notch** dterm_notch_hz
dterm_notch_cutoff

IN

Daten kommen aus den `Gyro Filter` Berechnungen

OUT

Daten gehen direkt an den `D-Controller`

DTerm => Dynamic D lowpass

Allgemein

Dynamischer Lowpass filter für den DTerm

Parameter

| Parameter | BFDefault | Bezeichnung |
|-----------------------------------|-----------|--|
| <code>dterm_lowpass_type</code> | 4.0 | PT1 / BIQUAD, sollte auf BIQUAD für LPF 1 stehen, LPF2 = PT1 |
| <code>dyn_lpf_dterm_min_hz</code> | 4.0 | LowPass min Hz |
| <code>dterm_lowpass2_hz</code> | 4.0 | LowPass2 min Hz |

DTerm => Static D notch

AllgemeinHz

Parameter

| Parameter | BFDefault | Bezeichnung |
|---------------------------------|-----------|-------------|
| <code>dterm_notch_hz</code> | | |
| <code>dterm_notch_cutoff</code> | | |

Feedforward

Allgemeines

Feedforward ist dem PID-Controller nachgelagert und unabhängig vom PID. FF verstärkt bzw. wirkt auf Deine Stickbewegung und hilft den Motoren schneller zu reagieren.

Mehr Infos zu Feedforward

- Feedforward BF 4.1 (<https://github.com/betaflight/betaflight/wiki/4.1-Tuning-Notes#feed-forward-boost>)
- Feedforward 2.0 BF 4.2 (<https://github.com/betaflight/betaflight/wiki/Feed-Forward-2.0>)

IN

OUT

Parameter

| Parameter | BFDefault | Bezeichnung |
|--------------------------------|---------------------------------|---|
| <code>ff_boost</code> | 4.115 | Der Booster verstärkt den gesamten FF-Wert aber zu einem sehr frühen Zeitpunkt und verringert damit ein Delay |
| <code>ff_interpolate_sp</code> | 4.1 ² (Average_2) | siehe Anhang der Tabelle |

| Parameter | BFDefault | Bezeichnung |
|--------------------------------|--|---|
| <code>ff_spike_limit</code> | 4.150 | Es wird eine effiziente verzögerungsfreie Dämpfungsmethode verwendet, die Erhöhung des boosts durch Spikes verringert bzw. vermeidet. Liegt der normale Boost-Wert unter dem limit wird er durchgelassen, alles weitere, hohe Boosts die durch Spikes verursacht werden, werden gedämpft. |
| <code>ff_max_rate_limit</code> | 4.1100 | <code>ff_max_rate_limit</code> unterbricht den Feedforward, wenn die Geschwindigkeit mit dem der Stick bewegt wird wahrscheinlich sein Ende des mechanischen Verfahrensweges erreicht. Dadurch wird ein Überspringen gerade bei Beginn von Flips reduziert. |
| <code>ff_smooth_factor</code> | 4.237 | Glättungsfaktor für einkommende Signale. Funktioniert wie ein LowPassfilter. 0 = keine Glättung, höhere Werte wie der Defaultwert, erhöhen auch die Latenzzeit und wirkt dem eigentlich FF-Forward entgegen |
| <code>ff_interpolate_sp</code> | Ausprägung * OFF * ON * AVERAGE-2 : passt für die meisten Copter & Freestyler * AVERAGE-3 : * AVERAGE-4 : | |

VBat

Allgemeines

Ab BF 4.2 wir mit VBAT-SAG-Kompensation Weitere Informationen: (<https://github.com/betaflight/betaflight/wiki/4.2-Tuning-Notes#dynamic-battery-sag-compensation>)

IN

OUT

Parameter

| Parameter | BFDefault | Bezeichnung |
|------------------------------------|-----------|---|
| <code>vbat_sag_compensation</code> | 4.2100 | 100% Kompensation der Batterieentladung |
| <code>vbat_pid_gain</code> | ? OFF | alte Version sollte immer OFF sein |

RC-Command

Allgemeines

IN

Receiver-Daten Signal

OUT

Daten werden mit den eingestellten `Rates` verrechnet und gelten dann als das angewendete `RCCommand-Eingangs-Signal`

Parameter

| Parameter | BFDefault | Bezeichnung |
|-------------------------------|-----------|-------------|
| <code>rc_interpolation</code> | | |
| <code>rc_interp</code> | | |

| Parameter | BFDDefaultBezeichnung |
|-----------|-----------------------|
|-----------|-----------------------|

| | |
|-------------|--|
| rc_inter_ch | |
|-------------|--|

| | |
|--------------|--|
| rc_inter_int | |
|--------------|--|

rc_smoothing_auto_smoothness (Default:10)

rc-smoothing-auto-smoothness setzt wie glatt der die RC-Signale sein sollen. Größere Werte erhöhen die Glättung vergrößern aber das RC-Delay. 10 ist optimal für die meisten allgemeinen Flüge. Racer bevorzugen 8 oder sogar 5. Das RC-Delay nimmt zwar ab, dafür können die Motor-Signale etwas unruhiger werden.

Setpoint

Allgemeines

In der weiteren Verarbeitung der Eingangssignale werden diese als **Setpoint** bezeichnet und spiegeln das RC-Signal wieder allerdings durch eine Reihe von Parametern geglättet

IN

Aufbereitetes RC-Command Signal

OUT

Daten die mittels **setpoint_smoothing** nochmals geglättet werden werden an *

Vorab-Fehler P-Berechnung * **d/dt**

Parameter

Setpoint/Setpoint smoothing beinhaltet eine Reihe von Parametern die das eigentlich Signal nochmals aufbereiten.

| Parameter | BFDDefaultBezeichnung |
|-----------|-----------------------|
|-----------|-----------------------|

| | |
|-------------------|--|
| rc_smoothing_type | |
|-------------------|--|

| | |
|------------------------------|----|
| rc_smoothing_auto_smoothness | 10 |
|------------------------------|----|

setzt wie glatt der die RC-Signale sein sollen. Größere Werte erhöhen die Glättung vergrößern aber das RC-Delay. 10 ist optimal für die meisten allgemeinen Flüge. Racer bevorzugen 8 oder sogar 5. Das RC-Delay nimmt zwar ab, dafür können die Motor-Signale etwas unruhiger werden

| | |
|-----------------------|--|
| rc_smoothing_input_Hz | |
|-----------------------|--|

| | |
|-------------------------|--|
| rc_smoothing_input_type | |
|-------------------------|--|

ITerm Parameter

Allgemeines

ITerm Parameter dienen dazu das I-Signal die PID-Controllers entweder vor der Bearbeitung von ITerm oder nach ITerm zu beeinflussen.

Insbesondere sollen hier Peaks im ITerm eliminiert werden.

Die Nachfolgende Tabelle beinhaltet zwei zusätzliche Spalten **IN** und **OUT** sie bezeichnen woher die Daten kommen (**IN**) und wer sie verwertet (**OUT**)

| Parameter | BFIN | OUT | Default | Bezeichnung |
|--------------------------------|------|---------------|---------|---|
| <code>item_windup</code> | | MIXER ITerm | | <p><code>item_windup</code> ist eine alte Methode zur Unterdrückung der iTerm-Akkumulation, wenn das Motordifferential einen benutzerdefinierten Schwellenwert überschreitet. In BF 4.2 wirkt <code>item_windup</code> nur noch auf YAW</p> <p><code>item_relax</code> hat <code>item_windup</code> zur Vermeidung von I-Anhäufungen auf Mini-Quads weitgehend ersetzt. <code>item_relax</code> ist hauptsächlich zur Vermeidung von bounce-backs</p> |
| <code>item_relax</code> | | ITERM PID_SUM | | |
| <code>item_relax_type</code> | | ITERM PID_SUM | | <p>Wenn der Pilot eine Änderung der Drehgeschwindigkeit anfordert, die für das Quad zu schnell ist, eilt das Gyrosignal dem Sollwert (SetPoint) hinterher und daraus entsteht ein mehr oder minder großes Fehlersignal. Der I-Term versucht nun diesen Fehler zu akkumulieren (aufsummieren) und versucht diese zu korrigieren. <code>item_relax</code> versucht nun diese Akkumulation zu kontrollieren. Reicht der <code>item_relax</code> nicht aus, sammeln sich die ITerm Fehler immer mehr an. Stoppt nun der Pilot seine Stickbewegung (z.B. in einem Flip), dann wird all der angesammelte ITerm-Fehler eine Gegenbewegung des Copters verursachen (BounceBack), der dann langsam ausklingen wird bis er wieder auf 0 ist. <code>item_relax</code> für Flip & Rolls und <code>item_windup</code> für YAW, versuchen diese Bouncebacks zu kontrollieren und abzumildern. <code>item_relax_cutoff</code> begrenzt die ITerm Akkumulation.</p> |
| <code>item_relax_cutoff</code> | | ITERM PID_SUM | | |
| <code>item_rotation</code> | | GYRO-Filter | ITerm | |

DMin - Handling

Mit DMin ist es nun möglich unterschiedliche Werte für D zu haben, je nachdem was der Copter gerade macht.

Im Normalflug erlaubt uns DMin mit reduzierten D-Werten zu fliegen und der DMax-Wert wird für schnelle Bewegungen genutzt (z.B. bei Fast-Roles, Flips, Propwash)

Außerdem bleiben die Motoren kühler

Wenn DMin im Konfigurator aktiviert wird, wird D in DMax umbenannt und es gibt eine neue Spalte DMin

DMin ist vom Profil abhängig (genau wie der D-Wert)

`d_min_boost_gain` steuert die Empfindlichkeit des Boost-Effekts (also wenn von DMin der DMax Wert verwendet werden soll, gibt der Booster an wie schnell das gehen soll).

Allgemeines

Was bringen geringere D-Werte

Vorteil

- weniger Vibrationen, kühlere Motoren
- Besseres Verhalten der Motoren bei Vollgas
- D-Wert bezogenes Oszillieren wird verringert

Nachteil

- mehr Propwash
- Größeres Überschießen und Bounce-Backs
- P Oszillation bei schnellen Manövern
- Langsame und Lowlevel Oszillationen bei smoothen Flügen

Setup für den Erstflug

Default für DMin R23, P25, Y0

Beachte

wenn DMin aktiviert ist, wird der reguläre DMax Wert nur dann genutzt, wenn schnelle Manöver (z.B. Flips/Rolls) geflogen werden, in langsameren Flügen wird `DMIN` verwendet

Prüfen des D Wertes im Flug

- Anzeige im OSD, `set debug_mode=D_MIN` und im OSD die Anzeige `debug2 on-screen`. Die Anzeige zeigt dir den 10fachen Wert. Beispiel: Anzeige 350 = 35D
- Über die Log-Aufzeichnung. Auch hier `set debug_mode=D_MIN` DEBUG2 im Blackboxexplorer zeigt dir unmittelbar D für die ROLL-Achse, DEBUG3 für PITCH.

DEBUG2 und DEBUG3 zeigen den D-Wert vor TPA.

DEBUG0 zeigt die Gyro-Anteil

Hinweis

Bei Verwendung von DMin erhöht sich die CPU-Last ein wenig. Implementiert ist ein Biquad-Filter und ein PT1-Filter.

Parameter

| Parameter | BFDefault | Bezeichnung |
|------------------|-----------|--|
| d_min | 0/xx | 0 = disabled DMin, Werte > 0 entsprechend dem DMin-Wert. Wenn DMin deaktiviert ist, ist aktuell immer der DMax (D) der genutzte Wert für D beschleunigt des boost-effekt, wenn sich Änderungen am setpoint ergeben (oder Gyro-Veränderungen). Wird unmittelbar bei Veränderungen durchgeführt bevor der Copter überhaupt die Bewegung ausführt. Der Wert kann zum Overshot beitragen, 0 deaktiviert diese Funktion und sollte bei Racern verwendet werden (und bei den meisten Coptern). |
| d_min_advanced | 20 | Verstärkungsfaktor, wie schnell bei schnellen Bewegungen D angepasst werden soll. 30-35 ist für normale Copter gut geeignet, 40-45 für wirklich sauber gebaute Freestyler. Wenn Propwash das Hauptproblem sind und die Motoren kühl sind, dann DMax erhöhen und den gain-Faktor ebenfalls (moderat!) |
| d_min_boost_gain | | |

Fine-Tuning des DMin Wertes geht nur über eine Blackboxauswertung. Für diese Analyse sollte der Debug-Mode D-MIN ausgewählt werden. Nur dann sieht man die aktuellen D_Min Werte im Flug

LunaX

(c) August 2020, Ritterhude

Documentation built with MkDocs (<http://www.mkdocs.org/>) using Windmill (<https://github.com/gristlabs/mkdocs-windmill>)
theme by Grist Labs.