

UAV common

Generated by Doxygen 1.9.1



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 AeroCoefficients Struct Reference	7
4.1.1 Detailed Description	7
4.1.2 Member Data Documentation	7
4.1.2.1 C0	7
4.1.2.2 Cab	8
4.1.2.3 Cpqr	8
4.1.2.4 d	8
4.1.2.5 eAR	8
4.1.2.6 S	8
4.1.2.7 stallLimit	8
4.2 AHRSPParams Struct Reference	8
4.2.1 Detailed Description	9
4.2.2 Member Data Documentation	9
4.2.2.1 alpha	9
4.2.2.2 Q	9
4.2.2.3 R	9
4.2.2.4 type	9
4.3 Ammo Class Reference	10
4.3.1 Constructor & Destructor Documentation	10
4.3.1.1 Ammo() [1/2]	10
4.3.1.2 Ammo() [2/2]	10
4.3.2 Member Function Documentation	10
4.3.2.1 getV0()	11
4.3.2.2 operator=()	11
4.3.3 Member Data Documentation	11
4.3.3.1 _V0	11
4.4 Cargo Class Reference	11
4.4.1 Constructor & Destructor Documentation	12
4.4.1.1 Cargo() [1/2]	12
4.4.1.2 Cargo() [2/2]	12
4.5 ControlSurfaces Class Reference	12
4.5.1 Detailed Description	12
4.5.2 Constructor & Destructor Documentation	13

4.5.2.1 ControlSurfaces() [1/2]	13
4.5.2.2 ControlSurfaces() [2/2]	13
4.5.3 Member Function Documentation	13
4.5.3.1 getCoefficients()	13
4.5.3.2 getNoOfSurface()	13
4.5.3.3 getValues()	14
4.5.3.4 restoreTrim()	14
4.5.3.5 setValues()	14
4.6 Drive Struct Reference	14
4.6.1 Detailed Description	14
4.6.2 Member Data Documentation	14
4.6.2.1 axis	15
4.6.2.2 hinges	15
4.6.2.3 noOfHinges	15
4.6.2.4 position	15
4.7 EKFSalers Struct Reference	15
4.7.1 Detailed Description	15
4.7.2 Member Data Documentation	16
4.7.2.1 baroScaler	16
4.7.2.2 predictScaler	16
4.7.2.3 updateScaler	16
4.7.2.4 zScaler	16
4.8 Hinge Class Reference	16
4.8.1 Detailed Description	17
4.8.2 Constructor & Destructor Documentation	17
4.8.2.1 Hinge() [1/3]	17
4.8.2.2 Hinge() [2/3]	17
4.8.2.3 Hinge() [3/3]	17
4.8.3 Member Function Documentation	17
4.8.3.1 getRot()	17
4.8.3.2 operator=()	18
4.8.3.3 updateValue()	18
4.9 Jet Class Reference	19
4.9.1 Detailed Description	19
4.9.2 Member Function Documentation	19
4.9.2.1 getLastThrust()	20
4.9.2.2 getThrust()	20
4.9.2.3 start()	20
4.9.3 Member Data Documentation	20
4.9.3.1 phases	21
4.9.3.2 thrust	21
4.9.3.3 time	21

4.10 Load Class Reference	21
4.10.1 Detailed Description	22
4.10.2 Constructor & Destructor Documentation	22
4.10.2.1 Load() [1/2]	22
4.10.2.2 Load() [2/2]	22
4.10.3 Member Function Documentation	22
4.10.3.1 getMass()	22
4.10.3.2 getOffset()	22
4.10.3.3 operator=()	23
4.10.3.4 release()	23
4.11 Logger Class Reference	23
4.11.1 Detailed Description	24
4.11.2 Constructor & Destructor Documentation	24
4.11.2.1 Logger()	24
4.11.2.2 ~Logger()	24
4.11.3 Member Function Documentation	24
4.11.3.1 log() [1/2]	24
4.11.3.2 log() [2/2]	25
4.11.3.3 setFmt()	25
4.11.3.4 setLogDirectory()	25
4.12 ODE Class Reference	26
4.12.1 Detailed Description	26
4.12.2 Member Enumeration Documentation	26
4.12.2.1 ODEMethod	26
4.12.3 Constructor & Destructor Documentation	27
4.12.3.1 ODE()	27
4.12.3.2 ~ODE()	27
4.12.4 Member Function Documentation	27
4.12.4.1 factory()	27
4.12.4.2 fromString()	28
4.12.4.3 getMicrosteps() [1/2]	28
4.12.4.4 getMicrosteps() [2/2]	28
4.12.4.5 step()	29
4.13 ODE_Euler Class Reference	29
4.13.1 Detailed Description	29
4.13.2 Constructor & Destructor Documentation	30
4.13.2.1 ODE_Euler()	30
4.13.3 Member Function Documentation	30
4.13.3.1 step()	30
4.14 ODE_Heun Class Reference	30
4.14.1 Detailed Description	31
4.14.2 Constructor & Destructor Documentation	31

4.14.2.1 ODE_Heun()	31
4.14.3 Member Function Documentation	31
4.14.3.1 step()	31
4.15 ODE_RK4 Class Reference	32
4.15.1 Detailed Description	32
4.15.2 Constructor & Destructor Documentation	32
4.15.2.1 ODE_RK4()	32
4.15.3 Member Function Documentation	32
4.15.3.1 step()	32
4.16 ODETest Class Reference	33
4.16.1 Member Function Documentation	33
4.16.1.1 SetUp()	33
4.16.1.2 TearDown()	33
4.17 PID Class Reference	34
4.17.1 Detailed Description	34
4.17.2 Constructor & Destructor Documentation	34
4.17.2.1 PID()	34
4.17.2.2 ~PID()	34
4.17.3 Member Function Documentation	34
4.17.3.1 calc() [1/2]	34
4.17.3.2 calc() [2/2]	35
4.17.3.3 clear()	35
4.17.3.4 set_dt()	35
4.18 Rotor Struct Reference	36
4.18.1 Detailed Description	36
4.18.2 Member Data Documentation	36
4.18.2.1 direction	36
4.18.2.2 forceCoff	36
4.18.2.3 hoverSpeed	37
4.18.2.4 maxSpeed	37
4.18.2.5 timeConstant	37
4.18.2.6 torqueCoff	37
4.19 SensorParams Struct Reference	37
4.19.1 Detailed Description	37
4.19.2 Member Data Documentation	38
4.19.2.1 bias	38
4.19.2.2 name	38
4.19.2.3 refreshTime	38
4.19.2.4 sd	38
4.20 TimedLoop Class Reference	38
4.20.1 Detailed Description	39
4.20.2 Constructor & Destructor Documentation	39

4.20.2.1 TimedLoop()	39
4.20.3 Member Function Documentation	39
4.20.3.1 go() [1/2]	39
4.20.3.2 go() [2/2]	39
4.21 UAVparams Struct Reference	40
4.21.1 Detailed Description	41
4.21.2 Constructor & Destructor Documentation	41
4.21.2.1 UAVparams()	41
4.21.2.2 ~UAVparams()	41
4.21.3 Member Function Documentation	41
4.21.3.1 getRotorHoverSpeeds()	41
4.21.3.2 getRotorMaxSpeeds()	41
4.21.3.3 getRotorTimeContants()	41
4.21.3.4 getSingleton()	42
4.21.3.5 loadConfig()	42
4.21.4 Member Data Documentation	42
4.21.4.1 aero_coffs	42
4.21.4.2 ahrs	42
4.21.4.3 ammo	42
4.21.4.4 cargo	42
4.21.4.5 ekf	42
4.21.4.6 initialMode	43
4.21.4.7 initialOrientation	43
4.21.4.8 initialPosition	43
4.21.4.9 initialVelocity	43
4.21.4.10 instantRun	43
4.21.4.11 lx	43
4.21.4.12 lxy	43
4.21.4.13 lxz	43
4.21.4.14 ly	44
4.21.4.15 lyz	44
4.21.4.16 lz	44
4.21.4.17 jets	44
4.21.4.18 m	44
4.21.4.19 name	44
4.21.4.20 noOfAmmo	44
4.21.4.21 noOfCargo	44
4.21.4.22 noOfJets	45
4.21.4.23 noOfRotors	45
4.21.4.24 pids	45
4.21.4.25 rotorMixer	45
4.21.4.26 rotors	45

4.21.4.27 sensors . . . . .	45
4.21.4.28 surfaceMixer . . . . .	45
4.21.4.29 surfaces . . . . .	45
<b>5 File Documentation</b>	<b>47</b>
5.1 header/common.hpp File Reference . . . . .	47
5.2 src/components/aero_coefficients.hpp File Reference . . . . .	47
5.3 src/components/components.hpp File Reference . . . . .	47
5.4 src/components/control_surfaces.cpp File Reference . . . . .	48
5.5 src/components/control_surfaces.hpp File Reference . . . . .	48
5.6 src/components/drive.cpp File Reference . . . . .	48
5.7 src/components/drive.hpp File Reference . . . . .	48
5.8 src/components/hinge.cpp File Reference . . . . .	48
5.8.1 Function Documentation . . . . .	49
5.8.1.1 asSkewMatrix() . . . . .	49
5.9 src/components/hinge.hpp File Reference . . . . .	49
5.10 src/components/loads.cpp File Reference . . . . .	49
5.11 src/components/loads.hpp File Reference . . . . .	49
5.12 src/components/navi.hpp File Reference . . . . .	50
5.13 src/logger/logger.cpp File Reference . . . . .	50
5.13.1 Function Documentation . . . . .	50
5.13.1.1 shouldLog() . . . . .	50
5.14 src/logger/logger.hpp File Reference . . . . .	50
5.14.1 Macro Definition Documentation . . . . .	51
5.14.1.1 LOGGER_MASK . . . . .	51
5.15 src/ode/ode.cpp File Reference . . . . .	51
5.16 src/ode/ode.hpp File Reference . . . . .	51
5.17 src/ode/ode_impl.hpp File Reference . . . . .	51
5.18 src/ode/ode_test.cpp File Reference . . . . .	52
5.18.1 Function Documentation . . . . .	52
5.18.1.1 getMethodsToTest() . . . . .	52
5.18.1.2 INSTITUTE_TEST_SUITE_P() . . . . .	52
5.18.1.3 main() . . . . .	53
5.18.1.4 TEST_F() [1/2] . . . . .	53
5.18.1.5 TEST_F() [2/2] . . . . .	53
5.18.1.6 TEST_P() [1/3] . . . . .	53
5.18.1.7 TEST_P() [2/3] . . . . .	53
5.18.1.8 TEST_P() [3/3] . . . . .	53
5.19 src/parser/parser.cpp File Reference . . . . .	54
5.19.1 Function Documentation . . . . .	54
5.19.1.1 parseMatrixXd() . . . . .	54
5.19.1.2 parseVectorXd() . . . . .	54



5.20 src/parser/parser.hpp File Reference . . . . .	55
5.20.1 Function Documentation . . . . .	55
5.20.1.1 parseMatrixXd() . . . . .	55
5.20.1.2 parseVectorXd() . . . . .	56
5.21 src/parser/uav_params.cpp File Reference . . . . .	56
5.21.1 Function Documentation . . . . .	56
5.21.1.1 parseHinge() . . . . .	56
5.21.1.2 parsePID() . . . . .	57
5.22 src/parser/uav_params.hpp File Reference . . . . .	57
5.23 src/PID/PID.cpp File Reference . . . . .	57
5.24 src/PID/PID.hpp File Reference . . . . .	57
5.24.1 Enumeration Type Documentation . . . . .	57
5.24.1.1 AntiWindUpMode . . . . .	57
5.25 src/timed_loop/status.hpp File Reference . . . . .	58
5.25.1 Enumeration Type Documentation . . . . .	58
5.25.1.1 Status . . . . .	58
5.26 src/timed_loop/timed_loop.cpp File Reference . . . . .	58
5.27 src/timed_loop/timed_loop.hpp File Reference . . . . .	59
<b>Index</b>	<b>61</b>



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AeroCoefficients . . . . .	7
AHRSParams . . . . .	8
ControlSurfaces . . . . .	12
Drive . . . . .	14
Jet . . . . .	19
Rotor . . . . .	36
EKFScalers . . . . .	15
Hinge . . . . .	16
Load . . . . .	21
Ammo . . . . .	10
Cargo . . . . .	11
Logger . . . . .	23
ODE . . . . .	26
ODE_Euler . . . . .	29
ODE_Heun . . . . .	30
ODE_RK4 . . . . .	32
PID . . . . .	34
SensorParams . . . . .	37
testing::TestWithParam	
ODETest . . . . .	33
TimedLoop . . . . .	38
UAVparams . . . . .	40



## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">AeroCoefficients</a>	
Aerodynamic coefficient . . . . .	7
<a href="#">AHRSParams</a>	
AHRS parameters . . . . .	8
<a href="#">Ammo</a>	10
<a href="#">Cargo</a>	11
<a href="#">ControlSurfaces</a>	
Aircraft's control surfaces . . . . .	12
<a href="#">Drive</a>	
<a href="#">Drive</a> propelling aircraft . . . . .	14
<a href="#">EKFScalers</a>	
Scalers for EKF . . . . .	15
<a href="#">Hinge</a>	
<a href="#">Hinge</a> connecting aircraft with drives . . . . .	16
<a href="#">Jet</a>	
<a href="#">Jet</a> rocket engine . . . . .	19
<a href="#">Load</a>	
<a href="#">Load</a> of aircraft that can be dropped or launched . . . . .	21
<a href="#">Logger</a>	
Log vector data with timestamp in file . . . . .	23
<a href="#">ODE</a>	
Ordinal differential equation solver . . . . .	26
<a href="#">ODE_Euler</a>	
Explicit Euler algorithm . . . . .	29
<a href="#">ODE_Heun</a>	
Second order explicit Heun algorithm . . . . .	30
<a href="#">ODE_RK4</a>	
Fourth order Runge Kutta algorithm . . . . .	32
<a href="#">ODETest</a>	33
<a href="#">PID</a>	
<a href="#">PID</a> discrete controller . . . . .	34
<a href="#">Rotor</a>	
<a href="#">Rotor</a> engine with controlled speed . . . . .	36
<a href="#">SensorParams</a>	
Base parameters of a sensor . . . . .	37

<a href="#">TimedLoop</a>	
Simulation of real-time synchronized loop . . . . .	<a href="#">38</a>
<a href="#">UAVparams</a>	
Parsed UAV configuration from XML . . . . .	<a href="#">40</a>

## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

header/ <a href="#">common.hpp</a>	47
src/components/ <a href="#">aero_coefficients.hpp</a>	47
src/components/ <a href="#">components.hpp</a>	47
src/components/ <a href="#">control_surfaces.cpp</a>	48
src/components/ <a href="#">control_surfaces.hpp</a>	48
src/components/ <a href="#">drive.cpp</a>	48
src/components/ <a href="#">drive.hpp</a>	48
src/components/ <a href="#">hinge.cpp</a>	48
src/components/ <a href="#">hinge.hpp</a>	49
src/components/ <a href="#">loads.cpp</a>	49
src/components/ <a href="#">loads.hpp</a>	49
src/components/ <a href="#">navi.hpp</a>	50
src/logger/ <a href="#">logger.cpp</a>	50
src/logger/ <a href="#">logger.hpp</a>	50
src/ode/ <a href="#">ode.cpp</a>	51
src/ode/ <a href="#">ode.hpp</a>	51
src/ode/ <a href="#">ode_impl.hpp</a>	51
src/ode/ <a href="#">ode_test.cpp</a>	52
src/parser/ <a href="#">parser.cpp</a>	54
src/parser/ <a href="#">parser.hpp</a>	55
src/parser/ <a href="#">uav_params.cpp</a>	56
src/parser/ <a href="#">uav_params.hpp</a>	57
src/PID/ <a href="#">PID.cpp</a>	57
src/PID/ <a href="#">PID.hpp</a>	57
src/timed_loop/ <a href="#">status.hpp</a>	58
src/timed_loop/ <a href="#">timed_loop.cpp</a>	58
src/timed_loop/ <a href="#">timed_loop.hpp</a>	59





## Chapter 4

# Class Documentation

### 4.1 AeroCoefficients Struct Reference

Aerodynamic coefficient.

```
#include <aero_coefficients.hpp>
```

#### Public Attributes

- double [S](#)
- double [d](#)
- double [eAR](#)
- Eigen::Vector< double, 6 > [C0](#)
- Eigen::Matrix< double, 6, 3 > [Cpqr](#)
- Eigen::Matrix< double, 6, 4 > [Cab](#)
- double [stallLimit](#)

#### 4.1.1 Detailed Description

Aerodynamic coefficient.

#### 4.1.2 Member Data Documentation

##### 4.1.2.1 C0

```
Eigen::Vector<double,6> AeroCoefficients::C0
```

#### 4.1.2.2 Cab

```
Eigen::Matrix<double,6,4> AeroCoefficients::Cab
```

#### 4.1.2.3 Cpqr

```
Eigen::Matrix<double,6,3> AeroCoefficients::Cpqr
```

#### 4.1.2.4 d

```
double AeroCoefficients::d
```

#### 4.1.2.5 eAR

```
double AeroCoefficients::eAR
```

#### 4.1.2.6 S

```
double AeroCoefficients::S
```

#### 4.1.2.7 stallLimit

```
double AeroCoefficients::stallLimit
```

The documentation for this struct was generated from the following file:

- [src/components/aero\\_coefficients.hpp](#)

## 4.2 AHRSParams Struct Reference

AHRS parameters.

```
#include <navi.hpp>
```

## Public Attributes

- std::string [type](#)
- double [alpha](#)
- double [Q](#)
- double [R](#)

### 4.2.1 Detailed Description

AHRS parameters.

### 4.2.2 Member Data Documentation

#### 4.2.2.1 [alpha](#)

```
double AHRSPParams::alpha
```

#### 4.2.2.2 [Q](#)

```
double AHRSPParams::Q
```

#### 4.2.2.3 [R](#)

```
double AHRSPParams::R
```

#### 4.2.2.4 [type](#)

```
std::string AHRSPParams::type
```

The documentation for this struct was generated from the following file:

- src/components/[navi.hpp](#)

## 4.3 Ammo Class Reference

```
#include <loads.hpp>
```

Inheritance diagram for Ammo:

Collaboration diagram for Ammo:

### Public Member Functions

- [Ammo](#) ()=default
- [Ammo](#) (int ammount, double [reload](#), Eigen::Vector3d offset, double mass, Eigen::Vector3d V0)
- [Ammo](#) & [operator=](#) (const [Ammo](#) &other)
- Eigen::Vector3d [getV0](#) ()  
*get start velocity of ammo when launched*

### Protected Attributes

- Eigen::Vector3d [\\_V0](#)

### Additional Inherited Members

#### 4.3.1 Constructor & Destructor Documentation

##### 4.3.1.1 Ammo() [1/2]

```
Ammo::Ammo ( ) [default]
```

##### 4.3.1.2 Ammo() [2/2]

```
Ammo::Ammo (
    int ammount,
    double reload,
    Eigen::Vector3d offset,
    double mass,
    Eigen::Vector3d V0 )
```

#### 4.3.2 Member Function Documentation

#### 4.3.2.1 getV0()

```
Eigen::Vector3d Ammo::getV0 ( ) [inline]
```

get start velocity of ammo when launched

##### Returns

start velocity vector

#### 4.3.2.2 operator=()

```
Ammo & Ammo::operator= (
    const Ammo & other )
```

### 4.3.3 Member Data Documentation

#### 4.3.3.1 \_V0

```
Eigen::Vector3d Ammo::_V0 [protected]
```

The documentation for this class was generated from the following files:

- [src/components/loads.hpp](#)
- [src/components/loads.cpp](#)

## 4.4 Cargo Class Reference

```
#include <loads.hpp>
```

Inheritance diagram for Cargo:

Collaboration diagram for Cargo:

### Public Member Functions

- [Cargo](#) ()=default
- [Cargo](#) (int ammount, double [reload](#), Eigen::Vector3d offset, double mass)

## Additional Inherited Members

### 4.4.1 Constructor & Destructor Documentation

#### 4.4.1.1 Cargo() [1/2]

```
Cargo::Cargo ( ) [default]
```

#### 4.4.1.2 Cargo() [2/2]

```
Cargo::Cargo (
    int ammount,
    double reload,
    Eigen::Vector3d offset,
    double mass )
```

The documentation for this class was generated from the following files:

- [src/components/loads.hpp](#)
- [src/components/loads.cpp](#)

## 4.5 ControlSurfaces Class Reference

Aircraft's control surfaces.

```
#include <control_surfaces.hpp>
```

### Public Member Functions

- [ControlSurfaces](#) ()
- [ControlSurfaces](#) (int noOfSurfaces, Eigen::Matrix< double, 6,-1 > matrix, Eigen::VectorXd min, Eigen::VectorXd max, Eigen::VectorXd trim)  
*Constructor.*
- Eigen::Vector< double, 6 > [getCoefficients](#) () const
- bool [setValues](#) (Eigen::VectorXd new\_values)
- void [restoreTrim](#) ()
- int [getNoOfSurface](#) () const
- Eigen::VectorXd [getValues](#) () const

#### 4.5.1 Detailed Description

Aircraft's control surfaces.

## 4.5.2 Constructor & Destructor Documentation

### 4.5.2.1 ControlSurfaces() [1/2]

```
ControlSurfaces::ControlSurfaces ( )
```

### 4.5.2.2 ControlSurfaces() [2/2]

```
ControlSurfaces::ControlSurfaces (
    int noOfSurfaces,
    Eigen::Matrix< double, 6,-1 > matrix,
    Eigen::VectorXd min,
    Eigen::VectorXd max,
    Eigen::VectorXd trim )
```

Constructor.

#### Parameters

<i>noOfSurfaces</i>	number of independent surfaces
<i>matrix</i>	coefficients matrix
<i>min</i>	vector of min angles
<i>max</i>	vector of max angles
<i>trim</i>	vector of trim angles

## 4.5.3 Member Function Documentation

### 4.5.3.1 getCoefficients()

```
Eigen::Vector< double, 6 > ControlSurfaces::getCoefficients ( ) const
```

### 4.5.3.2 getNoOfSurface()

```
int ControlSurfaces::getNoOfSurface ( ) const [inline]
```

#### 4.5.3.3 `getValues()`

```
Eigen::VectorXd ControlSurfaces::getValues ( ) const [inline]
```

#### 4.5.3.4 `restoreTrim()`

```
void ControlSurfaces::restoreTrim ( )
```

#### 4.5.3.5 `setValues()`

```
bool ControlSurfaces::setValues (
    Eigen::VectorXd new_values )
```

The documentation for this class was generated from the following files:

- [src/components/control\\_surfaces.hpp](#)
- [src/components/control\\_surfaces.cpp](#)

## 4.6 Drive Struct Reference

[Drive](#) propelling aircraft.

```
#include <drive.hpp>
```

Inheritance diagram for Drive:

Collaboration diagram for Drive:

### Public Attributes

- Eigen::Vector3d [position](#)
- Eigen::Vector3d [axis](#)
- int [noOfHinges](#)
- [Hinge](#) [hinges](#) [2]

#### 4.6.1 Detailed Description

[Drive](#) propelling aircraft.

#### 4.6.2 Member Data Documentation



#### 4.6.2.1 axis

```
Eigen::Vector3d Drive::axis
```

#### 4.6.2.2 hinges

```
Hinge Drive::hinges[2]
```

#### 4.6.2.3 noOfHinges

```
int Drive::noOfHinges
```

#### 4.6.2.4 position

```
Eigen::Vector3d Drive::position
```

The documentation for this struct was generated from the following file:

- [src/components/drive.hpp](#)

## 4.7 EKFSalers Struct Reference

Scalers for EKF.

```
#include <navi.hpp>
```

### Public Attributes

- double [predictScaler](#)
- double [updateScaler](#)
- double [baroScaler](#)
- double [zScaler](#)

#### 4.7.1 Detailed Description

Scalers for EKF.

## 4.7.2 Member Data Documentation

### 4.7.2.1 baroScaler

```
double EKFScalers::baroScaler
```

### 4.7.2.2 predictScaler

```
double EKFScalers::predictScaler
```

### 4.7.2.3 updateScaler

```
double EKFScalers::updateScaler
```

### 4.7.2.4 zScaler

```
double EKFScalers::zScaler
```

The documentation for this struct was generated from the following file:

- [src/components/navi.hpp](#)

## 4.8 Hinge Class Reference

[Hinge](#) connecting aircraft with drives.

```
#include <hinge.hpp>
```

### Public Member Functions

- [Hinge](#) ()=default
- [Hinge](#) (Eigen::Vector3d axis, double max, double min, double trim)
- [Hinge](#) (const [Hinge](#) &old)
- [Hinge](#) & [operator=](#) (const [Hinge](#) &old)
- void [updateValue](#) (double newValue)  
*set new angle on hinge*
- const Eigen::Matrix3d [getRot](#) ()  
*Get rotation matrix of orientation change due to hinge.*

## 4.8.1 Detailed Description

[Hinge](#) connecting aircraft with drives.

## 4.8.2 Constructor & Destructor Documentation

### 4.8.2.1 Hinge() [1/3]

```
Hinge::Hinge ( ) [default]
```

### 4.8.2.2 Hinge() [2/3]

```
Hinge::Hinge (
    Eigen::Vector3d axis,
    double max,
    double min,
    double trim )
```

### 4.8.2.3 Hinge() [3/3]

```
Hinge::Hinge (
    const Hinge & old )
```

## 4.8.3 Member Function Documentation

### 4.8.3.1 getRot()

```
const Eigen::Matrix3d Hinge::getRot ( )
```

Get rotation matrix of orientation change due to hinge.

#### Returns

rotation matrix

#### 4.8.3.2 operator=()

```
Hinge & Hinge::operator= (
    const Hinge & old )
```

#### 4.8.3.3 updateValue()

```
void Hinge::updateValue (
    double newValue )
```

set new angle on hinge

## Parameters

<i>newValue</i>	new angle of hinge
-----------------	--------------------

The documentation for this class was generated from the following files:

- [src/components/hinge.hpp](#)
- [src/components/hinge.cpp](#)

## 4.9 Jet Class Reference

[Jet](#) rocket engine.

```
#include <drive.hpp>
```

Inheritance diagram for Jet:

Collaboration diagram for Jet:

### Public Member Functions

- bool [start](#) (double [time](#))  
*start jet engine*
- double [getThrust](#) (double [time](#))  
*get thrust in specific time*
- double [getLastThrust](#) ()  
*get last calculated thrust*

### Public Attributes

- int [phases](#)
- Eigen::VectorXd [thrust](#)
- Eigen::VectorXd [time](#)

#### 4.9.1 Detailed Description

[Jet](#) rocket engine.

#### 4.9.2 Member Function Documentation

#### 4.9.2.1 getLastThrust()

```
double Jet::getLastThrust ( ) [inline]
```

get last calculated thrust

##### Returns

last calculated thrust

#### 4.9.2.2 getThrust()

```
double Jet::getThrust (
    double time )
```

get thrust in specific time

##### Parameters

<i>time</i>	timestamp
-------------	-----------

##### Returns

thrust value in Newtons

#### 4.9.2.3 start()

```
bool Jet::start (
    double time )
```

start jet engine

##### Parameters

<i>time</i>	timestamp of start
-------------	--------------------

##### Returns

true if start succesful, false if already started

### 4.9.3 Member Data Documentation

#### 4.9.3.1 phases

```
int Jet::phases
```

#### 4.9.3.2 thrust

```
Eigen::VectorXd Jet::thrust
```

#### 4.9.3.3 time

```
Eigen::VectorXd Jet::time
```

The documentation for this class was generated from the following files:

- [src/components/drive.hpp](#)
- [src/components/drive.cpp](#)

## 4.10 Load Class Reference

[Load](#) of aircraft that can be dropped or launched.

```
#include <loads.hpp>
```

Inheritance diagram for Load:

### Public Member Functions

- double [getMass](#) ()  
*get mass of load*
- Eigen::Vector3d [getOffset](#) ()  
*get offset of load*
- int [release](#) (double time)  
*Try to release load.*

### Protected Member Functions

- [Load](#) ()=default
- [Load](#) (int ammount, double [reload](#), Eigen::Vector3d offset, double mass)
- [Load](#) & [operator=](#) (const [Load](#) &other)

### 4.10.1 Detailed Description

[Load](#) of aircraft that can be dropped or launched.

### 4.10.2 Constructor & Destructor Documentation

#### 4.10.2.1 Load() [1/2]

```
Load::Load ( ) [protected], [default]
```

#### 4.10.2.2 Load() [2/2]

```
Load::Load (
    int ammount,
    double reload,
    Eigen::Vector3d offset,
    double mass ) [protected]
```

### 4.10.3 Member Function Documentation

#### 4.10.3.1 getMass()

```
double Load::getMass ( ) [inline]
```

get mass of load

##### Returns

mass

#### 4.10.3.2 getOffset()

```
Eigen::Vector3d Load::getOffset ( ) [inline]
```

get offset of load

##### Returns

offset vector



#### 4.10.3.3 operator=()

```
Load & Load::operator= (
    const Load & other )    [protected]
```

#### 4.10.3.4 release()

```
int Load::release (
    double time )
```

Try to release load.

##### Parameters

<i>time</i>	
-------------	--

##### Returns

leftover ammount of loads. Return -1 if load is not ready and -2 if out of load

The documentation for this class was generated from the following files:

- [src/components/loads.hpp](#)
- [src/components/loads.cpp](#)

## 4.11 Logger Class Reference

Log vector data with timestamp in file.

```
#include <logger.hpp>
```

### Public Member Functions

- [Logger](#) (std::string path, std::string fmt="", uint8\_t group=0)  
*Constructor.*
- [~Logger](#) ()  
*Destructor.*
- void [setFmt](#) (std::string fmt)  
*Set new format if was not known in constructor.*
- void [log](#) (double time, std::initializer\_list< Eigen::VectorXd > args)  
*Log one row.*
- void [log](#) (double time, std::initializer\_list< double > args)  
*Log one row.*

## Static Public Member Functions

- static void [setLogDirectory](#) (std::string subdirectory)

*Set global path that log should be created at. Path will be added to relative path of specific log instance.*

### 4.11.1 Detailed Description

Log vector data with timestamp in file.

### 4.11.2 Constructor & Destructor Documentation

#### 4.11.2.1 [Logger\(\)](#)

```
Logger::Logger (
    std::string path,
    std::string fmt = "",
    uint8_t group = 0 )
```

Constructor.

##### Parameters

<i>path</i>	relative path with log file name.
<i>fmt</i>	format - information about log structure. First line in log file
<i>group</i>	log group - log will be created only if group is in actual <code>LOGGER_MASK</code>

#### 4.11.2.2 [~Logger\(\)](#)

```
Logger::~~Logger ( )
```

Deconstructor.

### 4.11.3 Member Function Documentation

#### 4.11.3.1 [log\(\)](#) [1/2]

```
void Logger::log (
    double time,
    std::initializer_list< double > args )
```

Log one row.

## Parameters

<i>time</i>	timestamp
<i>args</i>	list of doubles

**4.11.3.2 log()** [2/2]

```
void Logger::log (
    double time,
    std::initializer_list< Eigen::VectorXd > args )
```

Log one row.

## Parameters

<i>time</i>	timestamp
<i>args</i>	list of double vectors

**4.11.3.3 setFmt()**

```
void Logger::setFmt (
    std::string fmt )
```

Set new format if was not known in constructor.

## Parameters

<i>fmt</i>	new format
------------	------------

**4.11.3.4 setLogDirectory()**

```
void Logger::setLogDirectory (
    std::string subdirectory ) [static]
```

Set global path that log should be created at. Path will be added to relative path of specific log instance.

## Parameters

<i>subdirectory</i>	new global log path
---------------------	---------------------

The documentation for this class was generated from the following files:

- [src/logger/logger.hpp](#)
- [src/logger/logger.cpp](#)

## 4.12 ODE Class Reference

Ordinal differential equation solver.

```
#include <ode.hpp>
```

Inheritance diagram for ODE:

### Public Types

- enum [ODEMethod](#) { [Euler](#) , [Heun](#) , [RK4](#) , [NONE](#) }
- Supported solving method.*

### Public Member Functions

- [ODE](#) (int micro\_steps)  
*Constructor.*
- virtual [~ODE](#) ()  
*Virtual destructor.*
- virtual Eigen::VectorXd [step](#) (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs\_fun, double h)=0  
*One step of explicit solving algorithm.*
- int [getMicrosteps](#) () const  
*Return microsteps - number of rhs function calls to calculate on step.*

### Static Public Member Functions

- static [ODEMethod fromString](#) (std::string str)  
*Parse solving method from string.*
- static std::unique\_ptr< [ODE](#) > [factory](#) ([ODEMethod](#) method)  
*Factory constructing ODE solvers.*
- static int [getMicrosteps](#) ([ODEMethod](#) method)  
*Get microsteps of given method.*

#### 4.12.1 Detailed Description

Ordinal differential equation solver.

#### 4.12.2 Member Enumeration Documentation

##### 4.12.2.1 ODEMethod

```
enum ODE::ODEMethod
```

Supported solving method.

## Enumerator

Euler	
Heun	
RK4	
NONE	

### 4.12.3 Constructor & Destructor Documentation

#### 4.12.3.1 ODE()

```
ODE::ODE (
    int micro_steps )
```

Constructor.

#### 4.12.3.2 ~ODE()

```
virtual ODE::~ODE ( ) [inline], [virtual]
```

Virtual destructor.

### 4.12.4 Member Function Documentation

#### 4.12.4.1 factory()

```
std::unique_ptr< ODE > ODE::factory (
    ODEMethod method ) [static]
```

Factory constructing [ODE](#) solvers.

## Parameters

<i>method</i>	type of desired method
---------------	------------------------

## Returns

instance of [ODE](#) solver

#### 4.12.4.2 fromString()

```
ODE::ODEMethod ODE::fromString (
    std::string str ) [static]
```

Parse solving method from string.

##### Parameters

<i>str</i>	input string
------------	--------------

##### Returns

solving method if parsed, NONE if unknown

#### 4.12.4.3 getMicrosteps() [1/2]

```
int ODE::getMicrosteps ( ) const
```

Return microsteps - number of rhs function calls to calculate on step.

##### Returns

microsteps

#### 4.12.4.4 getMicrosteps() [2/2]

```
int ODE::getMicrosteps (
    ODEMethod method ) [static]
```

Get microsteps of given method.

##### Parameters

<i>method</i>	method type
---------------	-------------

##### Returns

number of microstep in one algorithm step

#### 4.12.4.5 step()

```
virtual Eigen::VectorXd ODE::step (
    double t,
    Eigen::VectorXd y0,
    std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun,
    double h ) [pure virtual]
```

One step of explicit solving algorithm.

##### Parameters

<i>t</i>	start time
<i>y0</i>	start variable
<i>rhs_fun</i>	right-hand-side function, calculation of derivative
<i>h</i>	time step

##### Returns

Implemented in [ODE\\_RK4](#), [ODE\\_Heun](#), and [ODE\\_Euler](#).

The documentation for this class was generated from the following files:

- [src/ode/ode.hpp](#)
- [src/ode/ode.cpp](#)

## 4.13 ODE\_Euler Class Reference

Explicit Euler algorithm.

```
#include <ode_impl.hpp>
```

Inheritance diagram for ODE\_Euler:

Collaboration diagram for ODE\_Euler:

### Public Member Functions

- [ODE\\_Euler](#) ()
- Eigen::VectorXd [step](#) (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs\_fun, double h) override  
*One step of explicit solving algorithm.*

### Additional Inherited Members

#### 4.13.1 Detailed Description

Explicit Euler algorithm.

## 4.13.2 Constructor & Destructor Documentation

### 4.13.2.1 ODE\_Euler()

```
ODE_Euler::ODE_Euler ( ) [inline]
```

## 4.13.3 Member Function Documentation

### 4.13.3.1 step()

```
Eigen::VectorXd ODE_Euler::step (
    double t,
    Eigen::VectorXd y0,
    std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun,
    double h ) [inline], [override], [virtual]
```

One step of explicit solving algorithm.

#### Parameters

<i>t</i>	start time
<i>y0</i>	start variable
<i>rhs_fun</i>	right-hand-side function, calculation of derivative
<i>h</i>	time step

#### Returns

Implements [ODE](#).

The documentation for this class was generated from the following file:

- [src/ode/ode\\_impl.hpp](#)

## 4.14 ODE\_Heun Class Reference

Second order explicit Heun algorithm.

```
#include <ode_impl.hpp>
```

Inheritance diagram for ODE\_Heun:

Collaboration diagram for ODE\_Heun:



## Public Member Functions

- [ODE\\_Heun](#) ()
- `Eigen::VectorXd step (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun, double h) override`  
*One step of explicit solving algorithm.*

## Additional Inherited Members

### 4.14.1 Detailed Description

Second order explicit Heun algorithm.

### 4.14.2 Constructor & Destructor Documentation

#### 4.14.2.1 ODE\_Heun()

```
ODE_Heun::ODE_Heun ( ) [inline]
```

### 4.14.3 Member Function Documentation

#### 4.14.3.1 step()

```
Eigen::VectorXd ODE_Heun::step (
    double t,
    Eigen::VectorXd y0,
    std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun,
    double h ) [inline], [override], [virtual]
```

One step of explicit solving algorithm.

#### Parameters

<i>t</i>	start time
<i>y0</i>	start variable
<i>rhs_fun</i>	right-hand-side function, calculation of derivative
<i>h</i>	time step

## Returns

Implements [ODE](#).

The documentation for this class was generated from the following file:

- [src/ode/ode\\_impl.hpp](#)

## 4.15 ODE\_RK4 Class Reference

Fourth order Runge Kutta algorithm.

```
#include <ode_impl.hpp>
```

Inheritance diagram for ODE\_RK4:

Collaboration diagram for ODE\_RK4:

### Public Member Functions

- [ODE\\_RK4](#) ()
- [Eigen::VectorXd step](#) (double t, Eigen::VectorXd y0, std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs\_fun, double h) override  
*One step of explicit solving algorithm.*

### Additional Inherited Members

#### 4.15.1 Detailed Description

Fourth order Runge Kutta algorithm.

#### 4.15.2 Constructor & Destructor Documentation

##### 4.15.2.1 ODE\_RK4()

```
ODE_RK4::ODE_RK4 ( ) [inline]
```

#### 4.15.3 Member Function Documentation

##### 4.15.3.1 step()

```
Eigen::VectorXd ODE_RK4::step (
    double t,
    Eigen::VectorXd y0,
    std::function< Eigen::VectorXd(double, Eigen::VectorXd)> rhs_fun,
    double h ) [inline], [override], [virtual]
```

One step of explicit solving algorithm.

**Parameters**

<i>t</i>	start time
<i>y0</i>	start variable
<i>rhs_fun</i>	right-hand-side function, calculation of derivative
<i>h</i>	time step

**Returns**

Implements [ODE](#).

The documentation for this class was generated from the following file:

- [src/ode/ode\\_impl.hpp](#)

## 4.16 ODETest Class Reference

Inheritance diagram for ODETest:

Collaboration diagram for ODETest:

**Protected Member Functions**

- void [SetUp](#) () override
- void [TearDown](#) () override

### 4.16.1 Member Function Documentation

#### 4.16.1.1 SetUp()

```
void ODETest::SetUp ( ) [inline], [override], [protected]
```

#### 4.16.1.2 TearDown()

```
void ODETest::TearDown ( ) [inline], [override], [protected]
```

The documentation for this class was generated from the following file:

- [src/ode/ode\\_test.cpp](#)

## 4.17 PID Class Reference

PID discrete controller.

```
#include <PID.hpp>
```

### Public Member Functions

- [PID](#) (double Kp, double Ki, double Kd, double min=std::numeric\_limits< double >::min(), double max=std::numeric\_limits< double >::max(), [AntiWindUpMode](#) antiWindUp=[AntiWindUpMode::Clamping](#))
- [~PID](#) ()
- void [set\\_dt](#) (double dt)  
*Set new time step.*
- double [calc](#) (double error)  
*calc output of controller*
- double [calc](#) (double error, double dt)  
*calc output of controller with specific time step*
- void [clear](#) ()  
*clear internal state*

### 4.17.1 Detailed Description

PID discrete controller.

### 4.17.2 Constructor & Destructor Documentation

#### 4.17.2.1 PID()

```
PID::PID (
    double Kp,
    double Ki,
    double Kd,
    double min = std::numeric_limits<double>::min(),
    double max = std::numeric_limits<double>::max(),
    AntiWindUpMode antiWindUp = AntiWindUpMode::Clamping )
```

#### 4.17.2.2 ~PID()

```
PID::~PID ( )
```

### 4.17.3 Member Function Documentation

#### 4.17.3.1 calc() [1/2]

```
double PID::calc (
    double error )
```

calc output of controller

**Parameters**

<i>error</i>	input of controller
--------------	---------------------

**Returns**

output of controller

**4.17.3.2 calc() [2/2]**

```
double PID::calc (
    double error,
    double dt )
```

calc output of controller with specific time step

**Parameters**

<i>error</i>	input of controller
<i>dt</i>	time step

**Returns**

output of controller

**4.17.3.3 clear()**

```
void PID::clear ( )
```

clear internal state

**4.17.3.4 set\_dt()**

```
void PID::set_dt (
    double dt )
```

Set new time step.

**Parameters**

<i>dt</i>	new time step
-----------	---------------

The documentation for this class was generated from the following files:

- [src/PID/PID.hpp](#)
- [src/PID/PID.cpp](#)

## 4.18 Rotor Struct Reference

[Rotor](#) engine with controlled speed.

```
#include <drive.hpp>
```

Inheritance diagram for Rotor:

Collaboration diagram for Rotor:

### Public Attributes

- double [forceCoff](#)
- double [torqueCoff](#)
- int [direction](#)
- double [timeConstant](#)
- double [maxSpeed](#)
- double [hoverSpeed](#)

### 4.18.1 Detailed Description

[Rotor](#) engine with controlled speed.

### 4.18.2 Member Data Documentation

#### 4.18.2.1 direction

```
int Rotor::direction
```

#### 4.18.2.2 forceCoff

```
double Rotor::forceCoff
```

#### 4.18.2.3 hoverSpeed

```
double Rotor::hoverSpeed
```

#### 4.18.2.4 maxSpeed

```
double Rotor::maxSpeed
```

#### 4.18.2.5 timeConstant

```
double Rotor::timeConstant
```

#### 4.18.2.6 torqueCoff

```
double Rotor::torqueCoff
```

The documentation for this struct was generated from the following file:

- [src/components/drive.hpp](#)

## 4.19 SensorParams Struct Reference

Base parameters of a sensor.

```
#include <navi.hpp>
```

### Public Attributes

- `std::string` [name](#)
- `double` [sd](#)
- `Eigen::Vector3d` [bias](#)
- `double` [refreshTime](#)

#### 4.19.1 Detailed Description

Base parameters of a sensor.

## 4.19.2 Member Data Documentation

### 4.19.2.1 bias

```
Eigen::Vector3d SensorParams::bias
```

### 4.19.2.2 name

```
std::string SensorParams::name
```

### 4.19.2.3 refreshTime

```
double SensorParams::refreshTime
```

### 4.19.2.4 sd

```
double SensorParams::sd
```

The documentation for this struct was generated from the following file:

- [src/components/navi.hpp](#)

## 4.20 TimedLoop Class Reference

Simulation of real-time synchronized loop.

```
#include <timed_loop.hpp>
```

### Public Member Functions

- [TimedLoop](#) (int periodInMs, std::function< void(void)> func, [Status](#) &status)  
*Constructor.*
- void [go](#) ()  
*start infinite loop*
- void [go](#) (uint32\_t loops)  
*start loop for specific cycle numbers*



### 4.20.1 Detailed Description

Simulation of real-time synchronized loop.

### 4.20.2 Constructor & Destructor Documentation

#### 4.20.2.1 TimedLoop()

```
TimedLoop::TimedLoop (
    int periodInMs,
    std::function< void(void)> func,
    Status & status )
```

Constructor.

##### Parameters

<i>periodInMs</i>	loop period in milliseconds
<i>func</i>	function that should be called in loop
<i>status</i>	reference to controlling status

### 4.20.3 Member Function Documentation

#### 4.20.3.1 go() [1/2]

```
void TimedLoop::go ( )
```

start infinite loop

#### 4.20.3.2 go() [2/2]

```
void TimedLoop::go (
    uint32_t loops )
```

start loop for specific cycle numbers

##### Parameters

<i>loops</i>	how many cycles should be done
--------------	--------------------------------

The documentation for this class was generated from the following files:

- [src/timed\\_loop/timed\\_loop.hpp](#)
- [src/timed\\_loop/timed\\_loop.cpp](#)

## 4.21 UAVparams Struct Reference

Parsed UAV configuration from XML.

```
#include <uav_params.hpp>
```

Collaboration diagram for UAVparams:

### Public Member Functions

- [UAVparams](#) ()  
*Initialize default data.*
- [~UAVparams](#) ()
- void [loadConfig](#) (std::string configFile)
- Eigen::VectorXd [getRotorTimeConstants](#) () const
- Eigen::VectorXd [getRotorMaxSpeeds](#) () const
- Eigen::VectorXd [getRotorHoverSpeeds](#) () const

### Static Public Member Functions

- static const [UAVparams](#) \* [getSingleton](#) ()

### Public Attributes

- std::string [name](#)
- bool [instantRun](#)
- std::string [initialMode](#)
- Eigen::Vector3d [initialPosition](#)
- Eigen::Vector3d [initialOrientation](#)
- Eigen::Vector3d [initialVelocity](#)
- double [m](#)
- double [lx](#)
- double [ly](#)
- double [lz](#)
- double [lxy](#)
- double [lxz](#)
- double [lyz](#)
- int [noOfRotors](#)
- std::unique\_ptr< [Rotor](#)[]> [rotors](#)
- int [noOfJets](#)
- std::unique\_ptr< [Jet](#)[]> [jets](#)
- [ControlSurfaces](#) [surfaces](#)
- [AeroCoefficients](#) [aero\\_coffs](#)
- std::map< std::string, [PID](#) > [pids](#)
- std::vector< [SensorParams](#) > [sensors](#)
- [AHRSParams](#) [ahrs](#)
- [EKFSalers](#) [ekf](#)
- Eigen::MatrixX4d [rotorMixer](#)
- Eigen::MatrixX4d [surfaceMixer](#)
- int [noOfAmmo](#)
- std::unique\_ptr< [Ammo](#)[]> [ammo](#)
- int [noOfCargo](#)
- std::unique\_ptr< [Cargo](#)[]> [cargo](#)

### 4.21.1 Detailed Description

Parsed UAV configuration from XML.

### 4.21.2 Constructor & Destructor Documentation

#### 4.21.2.1 UAVparams()

```
UAVparams::UAVparams ( )
```

Initialize default data.

#### 4.21.2.2 ~UAVparams()

```
UAVparams::~~UAVparams ( )
```

### 4.21.3 Member Function Documentation

#### 4.21.3.1 getRotorHoverSpeeds()

```
Eigen::VectorXd UAVparams::getRotorHoverSpeeds ( ) const
```

#### 4.21.3.2 getRotorMaxSpeeds()

```
Eigen::VectorXd UAVparams::getRotorMaxSpeeds ( ) const
```

#### 4.21.3.3 getRotorTimeContants()

```
Eigen::VectorXd UAVparams::getRotorTimeContants ( ) const
```

#### 4.21.3.4 getSingleton()

```
const UAVparams * UAVparams::getSingleton ( ) [static]
```

#### 4.21.3.5 loadConfig()

```
void UAVparams::loadConfig (
    std::string configFile )
```

### 4.21.4 Member Data Documentation

#### 4.21.4.1 aero\_coffs

```
AeroCoefficients UAVparams::aero_coffs
```

#### 4.21.4.2 ahrs

```
AHRSParams UAVparams::ahrs
```

#### 4.21.4.3 ammo

```
std::unique_ptr<Ammo[ ]> UAVparams::ammo
```

#### 4.21.4.4 cargo

```
std::unique_ptr<Cargo[ ]> UAVparams::cargo
```

#### 4.21.4.5 ekf

```
EKFScalers UAVparams::ekf
```

#### 4.21.4.6 initialMode

`std::string UAVparams::initialMode`

#### 4.21.4.7 initialOrientation

`Eigen::Vector3d UAVparams::initialOrientation`

#### 4.21.4.8 initialPosition

`Eigen::Vector3d UAVparams::initialPosition`

#### 4.21.4.9 initialVelocity

`Eigen::Vector3d UAVparams::initialVelocity`

#### 4.21.4.10 instantRun

`bool UAVparams::instantRun`

#### 4.21.4.11 Ix

`double UAVparams::Ix`

#### 4.21.4.12 Ixy

`double UAVparams::Ixy`

#### 4.21.4.13 Ixz

`double UAVparams::Ixz`

**4.21.4.14 ly**

```
double UAVparams::Iy
```

**4.21.4.15 Iyz**

```
double UAVparams::Iyz
```

**4.21.4.16 Iz**

```
double UAVparams::Iz
```

**4.21.4.17 jets**

```
std::unique_ptr<Jet[]> UAVparams::jets
```

**4.21.4.18 m**

```
double UAVparams::m
```

**4.21.4.19 name**

```
std::string UAVparams::name
```

**4.21.4.20 noOfAmmo**

```
int UAVparams::noOfAmmo
```

**4.21.4.21 noOfCargo**

```
int UAVparams::noOfCargo
```

#### 4.21.4.22 noOfJets

```
int UAVparams::noOfJets
```

#### 4.21.4.23 noOfRotors

```
int UAVparams::noOfRotors
```

#### 4.21.4.24 pids

```
std::map<std::string, PID> UAVparams::pids
```

#### 4.21.4.25 rotorMixer

```
Eigen::MatrixX4d UAVparams::rotorMixer
```

#### 4.21.4.26 rotors

```
std::unique_ptr<Rotor[]> UAVparams::rotors
```

#### 4.21.4.27 sensors

```
std::vector<SensorParams> UAVparams::sensors
```

#### 4.21.4.28 surfaceMixer

```
Eigen::MatrixX4d UAVparams::surfaceMixer
```

#### 4.21.4.29 surfaces

```
ControlSurfaces UAVparams::surfaces
```

The documentation for this struct was generated from the following files:

- [src/parser/uav\\_params.hpp](#)
- [src/parser/uav\\_params.cpp](#)





## Chapter 5

# File Documentation

### 5.1 header/common.hpp File Reference

```
#include "../src/logger/logger.hpp"
#include "../src/ode/ode.hpp"
#include "../src/PID/PID.hpp"
#include "../src/timed_loop/timed_loop.hpp"
#include "../src/timed_loop/status.hpp"
#include "../src/parser/parser.hpp"
#include "../src/parser/uav_params.hpp"
#include "../src/components/components.hpp"
Include dependency graph for common.hpp:
```

### 5.2 src/components/aero\_coefficients.hpp File Reference

```
#include <Eigen/Dense>
```

Include dependency graph for aero\_coefficients.hpp: This graph shows which files directly or indirectly include this file:

#### Classes

- struct [AeroCoefficients](#)  
*Aerodynamic coefficient.*

### 5.3 src/components/components.hpp File Reference

```
#include "drive.hpp"
#include "control_surfaces.hpp"
#include "aero_coefficients.hpp"
#include "loads.hpp"
#include "navi.hpp"
```

Include dependency graph for components.hpp: This graph shows which files directly or indirectly include this file:

## 5.4 src/components/control\_surfaces.cpp File Reference

```
#include "control_surfaces.hpp"
```

Include dependency graph for control\_surfaces.cpp:

## 5.5 src/components/control\_surfaces.hpp File Reference

```
#include <Eigen/Dense>
```

Include dependency graph for control\_surfaces.hpp: This graph shows which files directly or indirectly include this file:

### Classes

- class [ControlSurfaces](#)  
*Aircraft's control surfaces.*

## 5.6 src/components/drive.cpp File Reference

```
#include "drive.hpp"
```

Include dependency graph for drive.cpp:

## 5.7 src/components/drive.hpp File Reference

```
#include <Eigen/Dense>
```

```
#include "hinge.hpp"
```

Include dependency graph for drive.hpp: This graph shows which files directly or indirectly include this file:

### Classes

- struct [Drive](#)  
*Drive propelling aircraft.*
- struct [Rotor](#)  
*Rotor engine with controlled speed.*
- class [Jet](#)  
*Jet rocket engine.*

## 5.8 src/components/hinge.cpp File Reference

```
#include "hinge.hpp"
```

Include dependency graph for hinge.cpp:

## Functions

- Eigen::Matrix3d [asSkewMatrix](#) (Eigen::Vector3d v)

### 5.8.1 Function Documentation

#### 5.8.1.1 asSkewMatrix()

```
Eigen::Matrix3d asSkewMatrix (  
    Eigen::Vector3d v )
```

## 5.9 src/components/hinge.hpp File Reference

```
#include <Eigen/Dense>  
#include <mutex>  
#include <memory>
```

Include dependency graph for hinge.hpp: This graph shows which files directly or indirectly include this file:

## Classes

- class [Hinge](#)  
*[Hinge](#) connecting aircraft with drives.*

## 5.10 src/components/loads.cpp File Reference

```
#include "loads.hpp"  
#include <limits>
```

Include dependency graph for loads.cpp:

## 5.11 src/components/loads.hpp File Reference

```
#include <Eigen/Dense>  
#include <atomic>
```

Include dependency graph for loads.hpp: This graph shows which files directly or indirectly include this file:

## Classes

- class [Load](#)  
*[Load](#) of aircraft that can be dropped or launched.*
- class [Ammo](#)
- class [Cargo](#)

## 5.12 src/components/navi.hpp File Reference

```
#include <Eigen/Dense>
```

Include dependency graph for navi.hpp: This graph shows which files directly or indirectly include this file:

### Classes

- struct [SensorParams](#)  
*Base parameters of a sensor.*
- struct [AHRSParams](#)  
*AHRS parameters.*
- struct [EKFScalers](#)  
*Scalers for EKF.*

## 5.13 src/logger/logger.cpp File Reference

```
#include "logger.hpp"  
#include <Eigen/Dense>  
#include <iostream>  
#include <fstream>  
#include <initializer_list>  
#include <string>  
#include <filesystem>
```

Include dependency graph for logger.cpp:

### Functions

- bool [shouldLog](#) (uint8\_t group)

### 5.13.1 Function Documentation

#### 5.13.1.1 shouldLog()

```
bool shouldLog (  
    uint8_t group )
```

## 5.14 src/logger/logger.hpp File Reference

```
#include <Eigen/Dense>  
#include <iostream>  
#include <fstream>  
#include <initializer_list>  
#include <string>  
#include <filesystem>
```

Include dependency graph for logger.hpp: This graph shows which files directly or indirectly include this file:

## Classes

- class [Logger](#)  
*Log vector data with timestamp in file.*

## Macros

- #define [LOGGER\\_MASK](#) -1

### 5.14.1 Macro Definition Documentation

#### 5.14.1.1 [LOGGER\\_MASK](#)

```
#define LOGGER\_MASK -1
```

## 5.15 src/ode/ode.cpp File Reference

```
#include "ode.hpp"  
#include "ode_impl.hpp"  
Include dependency graph for ode.cpp:
```

## 5.16 src/ode/ode.hpp File Reference

```
#include <functional>  
#include <memory>  
#include <Eigen/Dense>  
Include dependency graph for ode.hpp: This graph shows which files directly or indirectly include this file:
```

## Classes

- class [ODE](#)  
*Ordinal differencial equation solver.*

## 5.17 src/ode/ode\_impl.hpp File Reference

```
#include "ode.hpp"  
Include dependency graph for ode_impl.hpp: This graph shows which files directly or indirectly include this file:
```

## Classes

- class [ODE\\_Euler](#)  
*Explicit Euler algorithm.*
- class [ODE\\_Heun](#)  
*Second order explicit Heun algorithm.*
- class [ODE\\_RK4](#)  
*Fourth order Runge Kutta algorithm.*

## 5.18 src/ode/ode\_test.cpp File Reference

```
#include "ode.hpp"
#include <gtest/gtest.h>
#include <numbers>
Include dependency graph for ode_test.cpp:
```

## Classes

- class [ODETest](#)

## Functions

- `std::vector< ODE::ODEMethod > getMethodsToTest ()`
- `TEST\_F (ODETest, FromStringTest)`
- `TEST\_F (ODETest, FactoryTest)`
- `TEST\_P (ODETest, TestConstFunction)`
- `TEST\_P (ODETest, TestFirstOrder)`
- `TEST\_P (ODETest, TestRHSCalls)`
- `INSTANTIATE\_TEST\_SUITE\_P (TestDerivedClasses, ODETest, testing::ValuesIn(getMethodsToTest()))`
- `int main (int argc, char **argv)`

### 5.18.1 Function Documentation

#### 5.18.1.1 [getMethodsToTest\(\)](#)

```
std::vector<ODE::ODEMethod> getMethodsToTest ( )
```

#### 5.18.1.2 [INSTANTIATE\\_TEST\\_SUITE\\_P\(\)](#)

```
INSTANTIATE\_TEST\_SUITE\_P (
    TestDerivedClasses ,
    ODETest ,
    testing::ValuesIn(getMethodsToTest ()) )
```

### 5.18.1.3 main()

```
int main (
    int argc,
    char ** argv )
```

### 5.18.1.4 TEST\_F() [1/2]

```
TEST_F (
    ODETest ,
    FactoryTest )
```

### 5.18.1.5 TEST\_F() [2/2]

```
TEST_F (
    ODETest ,
    FromStringTest )
```

### 5.18.1.6 TEST\_P() [1/3]

```
TEST_P (
    ODETest ,
    TestConstFunction )
```

### 5.18.1.7 TEST\_P() [2/3]

```
TEST_P (
    ODETest ,
    TestFirstOrder )
```

### 5.18.1.8 TEST\_P() [3/3]

```
TEST_P (
    ODETest ,
    TestRHSCalls )
```

## 5.19 src/parser/parser.cpp File Reference

```
#include "parser.hpp"
#include <Eigen/Dense>
#include <iostream>
#include <sstream>
Include dependency graph for parser.cpp:
```

### Functions

- Eigen::MatrixXd [parseMatrixXd](#) (const std::string &input, int R, int C, char delimiter)  
*Parse input string to double matrix of specific shape and delimiter.*
- Eigen::VectorXd [parseVectorXd](#) (std::string str, int noOfElem, char delimiter)  
*Parse input string to double vector of specific length and delimiter.*

### 5.19.1 Function Documentation

#### 5.19.1.1 parseMatrixXd()

```
Eigen::MatrixXd parseMatrixXd (
    const std::string & input,
    int R,
    int C,
    char delimiter = ' ' )
```

Parse input string to double matrix of specific shape and delimiter.

##### Parameters

<i>input</i>	input string
<i>R</i>	number of rows
<i>C</i>	number of columns
<i>delimiter</i>	delimiter

##### Returns

parsed matrix

#### 5.19.1.2 parseVectorXd()

```
Eigen::VectorXd parseVectorXd (
    std::string str,
    int noOfElem,
    char delimiter = ' ' )
```

Parse input string to double vector of specific length and delimiter.



## Parameters

<i>str</i>	input string
<i>noOfElem</i>	length of vector
<i>delimiter</i>	delimiter

## Returns

parsed vector

## 5.20 src/parser/parser.hpp File Reference

```
#include <Eigen/Dense>
```

Include dependency graph for parser.hpp: This graph shows which files directly or indirectly include this file:

### Functions

- Eigen::MatrixXd [parseMatrixXd](#) (const std::string &input, int R, int C, char delimiter=' ')  
*Parse input string to double matrix of specific shape and delimiter.*
- Eigen::VectorXd [parseVectorXd](#) (std::string str, int noOfElem, char delimiter=' ')  
*Parse input string to double vector of specific length and delimiter.*

### 5.20.1 Function Documentation

#### 5.20.1.1 parseMatrixXd()

```
Eigen::MatrixXd parseMatrixXd (
    const std::string & input,
    int R,
    int C,
    char delimiter = ' ' )
```

Parse input string to double matrix of specific shape and delimiter.

## Parameters

<i>input</i>	input string
<i>R</i>	number of rows
<i>C</i>	number of columns
<i>delimiter</i>	delimiter

## Returns

parsed matrix

### 5.20.1.2 parseVectorXd()

```
Eigen::VectorXd parseVectorXd (
    std::string str,
    int noOfElem,
    char delimiter = ' ' )
```

Parse input string to double vector of specific length and delimiter.

#### Parameters

<i>str</i>	input string
<i>noOfElem</i>	length of vector
<i>delimiter</i>	delimiter

#### Returns

parsed vector

## 5.21 src/parser/uav\_params.cpp File Reference

```
#include <Eigen/Dense>
#include "uav_params.hpp"
#include <iostream>
#include <fstream>
#include <filesystem>
#include <mutex>
#include "rapidxml/rapidxml.hpp"
#include "parser.hpp"
Include dependency graph for uav_params.cpp:
```

### Functions

- void [parseHinge](#) (rapidxml::xml\_node<> \*hingeNode, [Hinge](#) \*hinge)
- [PID](#) [parsePID](#) (rapidxml::xml\_node<> \*PIDNode)

### 5.21.1 Function Documentation

#### 5.21.1.1 parseHinge()

```
void parseHinge (
    rapidxml::xml_node<> * hingeNode,
    Hinge * hinge )
```

### 5.21.1.2 parsePID()

```
PID parsePID (
    rapidxml::xml_node<> * PIDNode )
```

## 5.22 src/parser/uav\_params.hpp File Reference

```
#include <Eigen/Dense>
#include <mutex>
#include <memory>
#include <map>
#include "rapidxml/rapidxml.hpp"
#include "../components/components.hpp"
#include "../PID/PID.hpp"
```

Include dependency graph for uav\_params.hpp: This graph shows which files directly or indirectly include this file:

### Classes

- struct [UAVparams](#)  
*Parsed UAV configuration from XML.*

## 5.23 src/PID/PID.cpp File Reference

```
#include "PID.hpp"
#include <limits>
#include <algorithm>
```

Include dependency graph for PID.cpp:

## 5.24 src/PID/PID.hpp File Reference

```
#include <limits>
```

Include dependency graph for PID.hpp: This graph shows which files directly or indirectly include this file:

### Classes

- class [PID](#)  
*PID discrete controller.*

### Enumerations

- enum [AntiWindUpMode](#) { [None](#) , [Clamping](#) }  
*Methods of handling windup in controller.*

### 5.24.1 Enumeration Type Documentation

#### 5.24.1.1 AntiWindUpMode

```
enum AntiWindUpMode
```

Methods of handling windup in controller.

## Enumerator

None	
Clamping	

## 5.25 src/timed\_loop/status.hpp File Reference

This graph shows which files directly or indirectly include this file:

### Enumerations

- enum `Status` { `idle` = 1 , `running` = 2 , `exiting` = 3 , `reload` = 4 }
- status of timed loop. Control it's job*

### 5.25.1 Enumeration Type Documentation

#### 5.25.1.1 Status

enum `Status`

status of timed loop. Control it's job

## Enumerator

<code>idle</code>	loop is ready to run
<code>running</code>	loop is running
<code>exiting</code>	loop will be break in next occasion.
<code>reload</code>	loop job should be reloaded

## 5.26 src/timed\_loop/timed\_loop.cpp File Reference

```
#include "timed_loop.hpp"
#include <stdint.h>
#include <chrono>
#include <thread>
#include "status.hpp"
#include <iostream>
```

Include dependency graph for `timed_loop.cpp`:

## 5.27 src/timed\_loop/timed\_loop.hpp File Reference

```
#include <stdint.h>
#include <functional>
#include "status.hpp"
```

Include dependency graph for timed\_loop.hpp: This graph shows which files directly or indirectly include this file:

### Classes

- class [TimedLoop](#)  
*Simulation of real-time synchronized loop.*



# Index

- [\\_V0](#)
    - [Ammo, 11](#)
  - [~Logger](#)
    - [Logger, 24](#)
  - [~ODE](#)
    - [ODE, 27](#)
  - [~PID](#)
    - [PID, 34](#)
  - [~UAVparams](#)
    - [UAVparams, 41](#)
- [aero\\_coffs](#)
  - [UAVparams, 42](#)
- [AeroCoefficients, 7](#)
  - [C0, 7](#)
  - [Cab, 7](#)
  - [Cpqr, 8](#)
  - [d, 8](#)
  - [eAR, 8](#)
  - [S, 8](#)
  - [stallLimit, 8](#)
- [ahrs](#)
  - [UAVparams, 42](#)
- [AHRSPParams, 8](#)
  - [alpha, 9](#)
  - [Q, 9](#)
  - [R, 9](#)
  - [type, 9](#)
- [alpha](#)
  - [AHRSPParams, 9](#)
- [Ammo, 10](#)
  - [\\_V0, 11](#)
  - [Ammo, 10](#)
  - [getV0, 10](#)
  - [operator=, 11](#)
- [ammo](#)
  - [UAVparams, 42](#)
- [AntiWindUpMode](#)
  - [PID.hpp, 57](#)
- [asSkewMatrix](#)
  - [hinge.cpp, 49](#)
- [axis](#)
  - [Drive, 14](#)
- [baroScaler](#)
  - [EKFScalers, 16](#)
- [bias](#)
  - [SensorParams, 38](#)
- [C0](#)

- [AeroCoefficients, 7](#)
- [Cab](#)
  - [AeroCoefficients, 7](#)
- [calc](#)
  - [PID, 34, 35](#)
- [Cargo, 11](#)
  - [Cargo, 12](#)
- [cargo](#)
  - [UAVparams, 42](#)
- [Clamping](#)
  - [PID.hpp, 58](#)
- [clear](#)
  - [PID, 35](#)
- [ControlSurfaces, 12](#)
  - [ControlSurfaces, 13](#)
  - [getCoefficients, 13](#)
  - [getNoOfSurface, 13](#)
  - [getValues, 13](#)
  - [restoreTrim, 14](#)
  - [setValues, 14](#)
- [Cpqr](#)
  - [AeroCoefficients, 8](#)
- [d](#)
  - [AeroCoefficients, 8](#)
- [direction](#)
  - [Rotor, 36](#)
- [Drive, 14](#)
  - [axis, 14](#)
  - [hinges, 15](#)
  - [noOfHinges, 15](#)
  - [position, 15](#)
- [eAR](#)
  - [AeroCoefficients, 8](#)
- [ekf](#)
  - [UAVparams, 42](#)
- [EKFScalers, 15](#)
  - [baroScaler, 16](#)
  - [predictScaler, 16](#)
  - [updateScaler, 16](#)
  - [zScaler, 16](#)
- [Euler](#)
  - [ODE, 27](#)
- [exiting](#)
  - [status.hpp, 58](#)
- [factory](#)
  - [ODE, 27](#)
- [forceCoff](#)

- Rotor, [36](#)
- fromString
  - ODE, [27](#)
- getCoefficients
  - ControlSurfaces, [13](#)
- getLastThrust
  - Jet, [19](#)
- getMass
  - Load, [22](#)
- getMethodsToTest
  - ode\_test.cpp, [52](#)
- getMicrosteps
  - ODE, [28](#)
- getNoOfSurface
  - ControlSurfaces, [13](#)
- getOffset
  - Load, [22](#)
- getRot
  - Hinge, [17](#)
- getRotorHoverSpeeds
  - UAVparams, [41](#)
- getRotorMaxSpeeds
  - UAVparams, [41](#)
- getRotorTimeConstants
  - UAVparams, [41](#)
- getSingleton
  - UAVparams, [41](#)
- getThrust
  - Jet, [20](#)
- getV0
  - Ammo, [10](#)
- getValues
  - ControlSurfaces, [13](#)
- go
  - TimedLoop, [39](#)
- header/common.hpp, [47](#)
- Heun
  - ODE, [27](#)
- Hinge, [16](#)
  - getRot, [17](#)
  - Hinge, [17](#)
  - operator=, [17](#)
  - updateValue, [18](#)
- hinge.cpp
  - asSkewMatrix, [49](#)
- hinges
  - Drive, [15](#)
- hoverSpeed
  - Rotor, [36](#)
- idle
  - status.hpp, [58](#)
- initialMode
  - UAVparams, [42](#)
- initialOrientation
  - UAVparams, [43](#)
- initialPosition
  - UAVparams, [43](#)
- initialVelocity
  - UAVparams, [43](#)
- INstantiate\_TEST\_SUITE\_P
  - ode\_test.cpp, [52](#)
- instantRun
  - UAVparams, [43](#)
- lx
  - UAVparams, [43](#)
- lxy
  - UAVparams, [43](#)
- lxyz
  - UAVparams, [43](#)
- ly
  - UAVparams, [43](#)
- lyz
  - UAVparams, [44](#)
- lz
  - UAVparams, [44](#)
- Jet, [19](#)
  - getLastThrust, [19](#)
  - getThrust, [20](#)
  - phases, [20](#)
  - start, [20](#)
  - thrust, [21](#)
  - time, [21](#)
- jets
  - UAVparams, [44](#)
- Load, [21](#)
  - getMass, [22](#)
  - getOffset, [22](#)
  - Load, [22](#)
  - operator=, [22](#)
  - release, [23](#)
- loadConfig
  - UAVparams, [42](#)
- log
  - Logger, [24, 25](#)
- Logger, [23](#)
  - ~Logger, [24](#)
  - log, [24, 25](#)
  - Logger, [24](#)
  - setFmt, [25](#)
  - setLogDirectory, [25](#)
- logger.cpp
  - shouldLog, [50](#)
- logger.hpp
  - LOGGER\_MASK, [51](#)
- LOGGER\_MASK
  - logger.hpp, [51](#)
- m
  - UAVparams, [44](#)
- main
  - ode\_test.cpp, [52](#)
- maxSpeed
  - Rotor, [37](#)



- name
  - SensorParams, 38
  - UAVparams, 44
- NONE
  - ODE, 27
- None
  - PID.hpp, 58
- noOfAmmo
  - UAVparams, 44
- noOfCargo
  - UAVparams, 44
- noOfHinges
  - Drive, 15
- noOfJets
  - UAVparams, 44
- noOfRotors
  - UAVparams, 45
- ODE, 26
  - ~ODE, 27
  - Euler, 27
  - factory, 27
  - fromString, 27
  - getMicrosteps, 28
  - Heun, 27
  - NONE, 27
  - ODE, 27
  - ODEMethod, 26
  - RK4, 27
  - step, 28
- ODE\_Euler, 29
  - ODE\_Euler, 30
  - step, 30
- ODE\_Heun, 30
  - ODE\_Heun, 31
  - step, 31
- ODE\_RK4, 32
  - ODE\_RK4, 32
  - step, 32
- ode\_test.cpp
  - getMethodsToTest, 52
  - INstantiate\_TEST\_SUITE\_P, 52
  - main, 52
  - TEST\_F, 53
  - TEST\_P, 53
- ODEMethod
  - ODE, 26
- ODETest, 33
  - SetUp, 33
  - TearDown, 33
- operator=
  - Ammo, 11
  - Hinge, 17
  - Load, 22
- parseHinge
  - uav\_params.cpp, 56
- parseMatrixXd
  - parser.cpp, 54
  - parser.hpp, 55
- parsePID
  - uav\_params.cpp, 56
- parser.cpp
  - parseMatrixXd, 54
  - parseVectorXd, 54
- parser.hpp
  - parseMatrixXd, 55
  - parseVectorXd, 56
- parseVectorXd
  - parser.cpp, 54
  - parser.hpp, 56
- phases
  - Jet, 20
- PID, 34
  - ~PID, 34
  - calc, 34, 35
  - clear, 35
  - PID, 34
  - set\_dt, 35
- PID.hpp
  - AntiWindUpMode, 57
  - Clamping, 58
  - None, 58
- pids
  - UAVparams, 45
- position
  - Drive, 15
- predictScaler
  - EKFScalers, 16
- Q
  - AHRSPParams, 9
- R
  - AHRSPParams, 9
- refreshTime
  - SensorParams, 38
- release
  - Load, 23
- reload
  - status.hpp, 58
- restoreTrim
  - ControlSurfaces, 14
- RK4
  - ODE, 27
- Rotor, 36
  - direction, 36
  - forceCoff, 36
  - hoverSpeed, 36
  - maxSpeed, 37
  - timeConstant, 37
  - torqueCoff, 37
- rotorMixer
  - UAVparams, 45
- rotors
  - UAVparams, 45
- running
  - status.hpp, 58

## S

- AeroCoefficients, 8
- sd
  - SensorParams, 38
- SensorParams, 37
  - bias, 38
  - name, 38
  - refreshTime, 38
  - sd, 38
- sensors
  - UAVparams, 45
- set\_dt
  - PID, 35
- setFmt
  - Logger, 25
- setLogDirectory
  - Logger, 25
- SetUp
  - ODETest, 33
- setValues
  - ControlSurfaces, 14
- shouldLog
  - logger.cpp, 50
- src/components/aero\_coefficients.hpp, 47
- src/components/components.hpp, 47
- src/components/control\_surfaces.cpp, 48
- src/components/control\_surfaces.hpp, 48
- src/components/drive.cpp, 48
- src/components/drive.hpp, 48
- src/components/hinge.cpp, 48
- src/components/hinge.hpp, 49
- src/components/loads.cpp, 49
- src/components/loads.hpp, 49
- src/components/navi.hpp, 50
- src/logger/logger.cpp, 50
- src/logger/logger.hpp, 50
- src/ode/ode.cpp, 51
- src/ode/ode.hpp, 51
- src/ode/ode\_impl.hpp, 51
- src/ode/ode\_test.cpp, 52
- src/parser/parser.cpp, 54
- src/parser/parser.hpp, 55
- src/parser/uav\_params.cpp, 56
- src/parser/uav\_params.hpp, 57
- src/PID/PID.cpp, 57
- src/PID/PID.hpp, 57
- src/timed\_loop/status.hpp, 58
- src/timed\_loop/timed\_loop.cpp, 58
- src/timed\_loop/timed\_loop.hpp, 59
- stallLimit
  - AeroCoefficients, 8
- start
  - Jet, 20
- Status
  - status.hpp, 58
- status.hpp
  - exiting, 58
  - idle, 58
  - reload, 58
  - running, 58
  - Status, 58
- step
  - ODE, 28
  - ODE\_Euler, 30
  - ODE\_Heun, 31
  - ODE\_RK4, 32
- surfaceMixer
  - UAVparams, 45
- surfaces
  - UAVparams, 45
- TearDown
  - ODETest, 33
- TEST\_F
  - ode\_test.cpp, 53
- TEST\_P
  - ode\_test.cpp, 53
- thrust
  - Jet, 21
- time
  - Jet, 21
- timeConstant
  - Rotor, 37
- TimedLoop, 38
  - go, 39
  - TimedLoop, 39
- torqueCoff
  - Rotor, 37
- type
  - AHRSPParams, 9
- uav\_params.cpp
  - parseHinge, 56
  - parsePID, 56
- UAVparams, 40
  - ~UAVparams, 41
  - aero\_coffs, 42
  - ahrs, 42
  - ammo, 42
  - cargo, 42
  - ekf, 42
  - getRotorHoverSpeeds, 41
  - getRotorMaxSpeeds, 41
  - getRotorTimeContants, 41
  - getSingleton, 41
  - initialMode, 42
  - initialOrientation, 43
  - initialPosition, 43
  - initialVelocity, 43
  - instantRun, 43
  - lx, 43
  - lxy, 43
  - lxz, 43
  - ly, 43
  - lyz, 44
  - lz, 44
  - jets, 44

- loadConfig, [42](#)
- m, [44](#)
- name, [44](#)
- noOfAmmo, [44](#)
- noOfCargo, [44](#)
- noOfJets, [44](#)
- noOfRotors, [45](#)
- pids, [45](#)
- rotorMixer, [45](#)
- rotors, [45](#)
- sensors, [45](#)
- surfaceMixer, [45](#)
- surfaces, [45](#)
- UAVparams, [41](#)
- updateScaler
  - EKFScalers, [16](#)
- updateValue
  - Hinge, [18](#)
- zScaler
  - EKFScalers, [16](#)