

openEVario
0.1

Generated by Doxygen 1.8.9.1

Sat Mar 5 2016 23:00:01

Contents

1	Todo List	1
2	Namespace Index	3
2.1	Namespace List	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Namespace Documentation	9
5.1	openEV Namespace Reference	9
5.1.1	Typedef Documentation	9
5.1.1.1	FloatType	9
5.1.1.2	Vector3DType	9
5.1.2	Variable Documentation	10
5.1.2.1	GRAVITY	10
5.1.2.2	lenLatitude	10
6	Class Documentation	11
6.1	openEV::FastMath Class Reference	11
6.1.1	Detailed Description	12
6.1.2	Constructor & Destructor Documentation	12
6.1.2.1	FastMath	12
6.1.2.2	~FastMath	12
6.1.3	Member Function Documentation	12
6.1.3.1	fastCos	12
6.1.3.2	fastSin	12
6.1.3.3	fastSinPositive	12
6.1.3.4	fastSinRaw	13
6.1.4	Member Data Documentation	13
6.1.4.1	degToRad	13

6.1.4.2	radToDeg	13
6.1.4.3	sineSamplesPerDegree	13
6.1.4.4	sinusTable	13
6.1.4.5	sizeSineTable	13
6.2	openEV::GliderVarioMeasurementMatrix Class Reference	14
6.2.1	Detailed Description	14
6.2.2	Member Typedef Documentation	14
6.2.2.1	MeasureMatrixType	14
6.2.3	Constructor & Destructor Documentation	14
6.2.3.1	GliderVarioMeasurementMatrix	14
6.2.3.2	~GliderVarioMeasurementMatrix	14
6.2.4	Member Function Documentation	14
6.2.4.1	calcMeasurementMatrix	14
6.2.4.2	getMeasureMatrix	15
6.2.5	Member Data Documentation	15
6.2.5.1	measurementMatrix	15
6.3	openEV::GliderVarioMeasurementVector Class Reference	15
6.3.1	Detailed Description	16
6.3.2	Member Typedef Documentation	16
6.3.2.1	MeasureVectorType	16
6.3.3	Member Enumeration Documentation	16
6.3.3.1	MeasureComponentIndex	16
6.3.4	Constructor & Destructor Documentation	17
6.3.4.1	GliderVarioMeasurementVector	17
6.3.4.2	~GliderVarioMeasurementVector	17
6.3.5	Member Function Documentation	17
6.3.5.1	getMeasureVector	17
6.3.6	Member Data Documentation	17
6.3.6.1	accelX	17
6.3.6.2	accelY	17
6.3.6.3	accelZ	18
6.3.6.4	gpsHeading	18
6.3.6.5	gpsLatitude	18
6.3.6.6	gpsLongitude	18
6.3.6.7	gpsMSL	18
6.3.6.8	gpsSpeed	18
6.3.6.9	gyroRateX	18
6.3.6.10	gyroRateY	18
6.3.6.11	gyroRateZ	19
6.3.6.12	magX	19

6.3.6.13	magY	19
6.3.6.14	magZ	19
6.3.6.15	measureVector	19
6.3.6.16	pressAlt	19
6.3.6.17	trueAirSpeed	19
6.4	openEV::GliderVarioStatus Class Reference	19
6.4.1	Detailed Description	21
6.4.2	Member Typedef Documentation	22
6.4.2.1	StatusVectorType	22
6.4.3	Member Enumeration Documentation	23
6.4.3.1	StatusComponentIndex	23
6.4.4	Constructor & Destructor Documentation	23
6.4.4.1	GliderVarioStatus	23
6.4.4.2	~GliderVarioStatus	24
6.4.5	Member Function Documentation	24
6.4.5.1	getStatusVector	24
6.4.5.2	getStatusVector	24
6.4.5.3	normalizeAngles	24
6.4.6	Member Data Documentation	24
6.4.6.1	accelX	24
6.4.6.2	accelY	24
6.4.6.3	accelZ	24
6.4.6.4	altMSL	24
6.4.6.5	groundSpeedEast	24
6.4.6.6	groundSpeedNorth	25
6.4.6.7	gyroBiasX	25
6.4.6.8	gyroBiasY	25
6.4.6.9	gyroBiasZ	25
6.4.6.10	heading	25
6.4.6.11	latitude	25
6.4.6.12	longitude	25
6.4.6.13	pitchAngle	25
6.4.6.14	pitchRateY	25
6.4.6.15	rateOfSink	26
6.4.6.16	rollAngle	26
6.4.6.17	rollRateX	26
6.4.6.18	statusVector	26
6.4.6.19	thermalSpeed	26
6.4.6.20	trueAirSpeed	26
6.4.6.21	verticalSpeed	26

6.4.6.22	windSpeedEast	26
6.4.6.23	windSpeedNorth	26
6.4.6.24	yawAngle	27
6.4.6.25	yawRateZ	27
6.5	openEV::GliderVarioTransitionMatrix Class Reference	27
6.5.1	Detailed Description	27
6.5.2	Member Typedef Documentation	27
6.5.2.1	TransitionMatrixType	27
6.5.3	Constructor & Destructor Documentation	28
6.5.3.1	GliderVarioTransitionMatrix	28
6.5.3.2	~GliderVarioTransitionMatrix	28
6.5.4	Member Function Documentation	28
6.5.4.1	calcTransitionMatrix	28
6.5.4.2	getTransitionMatrix	28
6.5.4.3	updateStatus	28
6.5.5	Member Data Documentation	28
6.5.5.1	transitionMatrix	28
6.6	openEV::MeasureMatrix Class Reference	29
6.6.1	Detailed Description	29
6.6.2	Constructor & Destructor Documentation	29
6.6.2.1	MeasureMatrix	29
6.6.2.2	~MeasureMatrix	29
6.7	openEV::RotationMatrix Class Reference	29
6.7.1	Detailed Description	30
6.7.2	Member Typedef Documentation	30
6.7.2.1	RotationMatrixType	30
6.7.3	Constructor & Destructor Documentation	30
6.7.3.1	RotationMatrix	30
6.7.3.2	RotationMatrix	30
6.7.3.3	~RotationMatrix	31
6.7.4	Member Function Documentation	31
6.7.4.1	calcPlaneVectorToWorldVector	31
6.7.4.2	calculateRotationMatrixGloToPlane	31
6.7.4.3	calculateRotationMatrixPlaneToGlo	31
6.7.4.4	calcWorldVectorToPlaneVector	31
6.7.4.5	getMatrixGloToPlane	31
6.7.4.6	getMatrixPlaneToGlo	32
6.7.4.7	getPitch	32
6.7.4.8	getRoll	32
6.7.4.9	getYaw	32

6.7.4.10	setPitch	32
6.7.4.11	setRoll	32
6.7.4.12	setYaw	32
6.7.5	Member Data Documentation	32
6.7.5.1	matrixGloToPlane	32
6.7.5.2	matrixGloToPlanelValid	32
6.7.5.3	matrixPlaneToGlo	33
6.7.5.4	matrixPlaneToGloValid	33
6.7.5.5	pitch	33
6.7.5.6	roll	33
6.7.5.7	yaw	33
7	File Documentation	35
7.1	src/FastMath.cpp File Reference	35
7.2	src/FastMath.h File Reference	35
7.2.1	Macro Definition Documentation	36
7.2.1.1	M_PI	36
7.3	src/FastMath_test.cpp File Reference	37
7.4	src/FastMathSineTable.cpp File Reference	38
7.5	src/genSineTables.cpp File Reference	38
7.5.1	Function Documentation	39
7.5.1.1	main	39
7.5.1.2	printSineTable	39
7.5.1.3	usage	40
7.6	src/GliderVarioMeasurementMatrix.cpp File Reference	40
7.7	src/GliderVarioMeasurementMatrix.h File Reference	40
7.8	src/GliderVarioMeasurementMatrix_test.cpp File Reference	41
7.9	src/GliderVarioMeasurementVector.cpp File Reference	42
7.10	src/GliderVarioMeasurementVector.h File Reference	43
7.11	src/GliderVarioMeasurementVector_test.cpp File Reference	44
7.12	src/GliderVarioStatus.cpp File Reference	45
7.12.1	Function Documentation	45
7.12.1.1	operator<<	45
7.13	src/GliderVarioStatus.h File Reference	46
7.13.1	Function Documentation	46
7.13.1.1	operator<<	46
7.14	src/GliderVarioStatus_test.cpp File Reference	47
7.15	src/GliderVarioTransitionMatrix.cpp File Reference	47
7.16	src/GliderVarioTransitionMatrix.h File Reference	48
7.17	src/GliderVarioTransitionMatrix_test.cpp File Reference	49

7.18	src/MeasureMatrix.cpp File Reference	49
7.19	src/MeasureMatrix.h File Reference	50
7.20	src/MeasureMatrix_test.cpp File Reference	50
7.21	src/openEVario.cpp File Reference	51
7.21.1	Function Documentation	51
7.21.1.1	main	51
7.21.2	Variable Documentation	52
7.21.2.1	randomGenerator	52
7.21.2.2	x	52
7.22	src/RotationMatrix.cpp File Reference	52
7.23	src/RotationMatrix.h File Reference	53
Index		55

Chapter 1

Todo List

Member `openEV::GliderVarioTransitionMatrix::calcTransitionMatrix` (`FloatType timeDiff`, `GliderVarioStatus const &lastStatus`)

Calculation of Rate of Sink: Refine the vario compensation by considering the decrease of drag based on the polar.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

[openEV](#) ??

Chapter 3

Class Index

3.1 Class List

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

openEV::FastMath	??
openEV::GliderVarioMeasurementMatrix	??
openEV::GliderVarioMeasurementVector	??
openEV::GliderVarioStatus	
GliderVarioStatus manages the Kalman filter state x	??
openEV::GliderVarioTransitionMatrix	??
openEV::MeasureMatrix	??
openEV::RotationMatrix	??

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/ FastMath.cpp	??
src/ FastMath.h	??
src/ FastMath_test.cpp	??
src/ FastMathSineTable.cpp	??
src/ genSineTables.cpp	??
src/ GliderVarioMeasurementMatrix.cpp	??
src/ GliderVarioMeasurementMatrix.h	??
src/ GliderVarioMeasurementMatrix_test.cpp	??
src/ GliderVarioMeasurementVector.cpp	??
src/ GliderVarioMeasurementVector.h	??
src/ GliderVarioMeasurementVector_test.cpp	??
src/ GliderVarioStatus.cpp	??
src/ GliderVarioStatus.h	??
src/ GliderVarioStatus_test.cpp	??
src/ GliderVarioTransitionMatrix.cpp	??
src/ GliderVarioTransitionMatrix.h	??
src/ GliderVarioTransitionMatrix_test.cpp	??
src/ MeasureMatrix.cpp	??
src/ MeasureMatrix.h	??
src/ MeasureMatrix_test.cpp	??
src/ openEVario.cpp	??
src/ RotationMatrix.cpp	??
src/ RotationMatrix.h	??

Chapter 5

Namespace Documentation

5.1 openEV Namespace Reference

Classes

- class [FastMath](#)
- class [GliderVarioMeasurementMatrix](#)
- class [GliderVarioMeasurementVector](#)
- class [GliderVarioStatus](#)
[GliderVarioStatus](#) manages the Kalman filter state x .
- class [GliderVarioTransitionMatrix](#)
- class [MeasureMatrix](#)
- class [RotationMatrix](#)

Typedefs

- typedef float [FloatType](#)
- typedef Eigen::Matrix< [FloatType](#), 3, 1 > [Vector3DType](#)

Variables

- [FloatType](#) constexpr [lenLatitude](#) = 111132.0
- static [FloatType](#) constexpr [GRAVITY](#) = 9.81

5.1.1 Typedef Documentation

5.1.1.1 typedef float openEV::FloatType

The global float type. Change this one to double, and the entire system will run in double. For optimal performance this should be *float*. Eigen can use the NEON unit for vectorized arithmetic.

Definition at line 43 of file [GliderVarioStatus.h](#).

5.1.1.2 typedef Eigen::Matrix<FloatType, 3, 1> openEV::Vector3DType

This vector type is used for all 3-dimensional representations of values in Kartesian coordinates

Definition at line 48 of file [GliderVarioStatus.h](#).

5.1.2 Variable Documentation

5.1.2.1 `FloatType constexpr openEV::GRAVITY = 9.81` `[static]`

Constant of gravity acceleration. exact values for Germany can be obtained from the German gravity base mesh Deutsches Schweregrundnetz 1994 (DSGN 94) http://www.bkg.bund.de/nn_175464/SharedDocs/Download/DE-Dok/DSGN94-Punktbeschreibung-PDF-de,templateId=raw,property=publicationFile.pdf/DSGN94-Punktbeschreibung-PDF-de.pdf The constant here is a rough average between Hamburg and Munich (I live in Norther Germany). Since a Kalman filter is not exact numeric science any inaccuracy should be covered by the process variance.

Definition at line 42 of file `GliderVarioTransitionMatrix.h`.

5.1.2.2 `FloatType constexpr openEV::lenLatitude = 111132.0`

The rough length of a degree latitude in meter at 45deg North. https://en.wikipedia.org/wiki/Longitude#Noting_and_calculating_longitude

Definition at line 38 of file `GliderVarioTransitionMatrix.cpp`.

Chapter 6

Class Documentation

6.1 openEV::FastMath Class Reference

```
#include <FastMath.h>
```

Public Member Functions

- [FastMath](#) ()
- virtual [~FastMath](#) ()

Static Public Member Functions

- static [FloatType fastSin](#) ([FloatType](#) angle)
- static [FloatType fastCos](#) ([FloatType](#) angle)

Static Public Attributes

- static constexpr unsigned [sineSamplesPerDegree](#) = 8
the sinus table is calculated in 1/8 degree steps
- static constexpr unsigned [sizeSineTable](#) = 360*sineSamplesPerDegree
the sinus table is calculated in 1/8 degree steps
- static constexpr double [radToDeg](#) = 180.0 / [M_PI](#)
- static constexpr double [degToRad](#) = [M_PI](#) / 180.0

Static Protected Member Functions

- static [FloatType fastSinRaw](#) ([FloatType](#) angle)
- static [FloatType fastSinPositive](#) ([FloatType](#) angle)

Static Protected Attributes

- static const double [sinusTable](#) [[sizeSineTable](#)+1]
The table of per-computed sinus values. The table is one item longer than sizeSineTable because I need the interpolation to +360 degrees!

6.1.1 Detailed Description

FastMath

Faster implementations of CPU and time intensive functions, particular trigonometric functions. For a Kalman filter the last bit of accuracy is not required. That is what the process (co)variance is for (within other inaccuracies :)).

All trigonometric functions here are used in degrees (0-360 deg)!

Definition at line 54 of file FastMath.h.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 openEV::FastMath::FastMath ()

Definition at line 31 of file FastMath.cpp.

6.1.2.2 openEV::FastMath::~~FastMath () [virtual]

Definition at line 36 of file FastMath.cpp.

6.1.3 Member Function Documentation

6.1.3.1 static FloatType openEV::FastMath::fastCos (FloatType *angle*) [inline],[static]

Parameters

<i>angle[in]</i>	in degrees
------------------	------------

Returns

The cosine value of the angle

Definition at line 88 of file FastMath.h.

6.1.3.2 static FloatType openEV::FastMath::fastSin (FloatType *angle*) [inline],[static]

Parameters

<i>angle[in]</i>	in degrees.
------------------	-------------

Returns

The sine value of the angle

Definition at line 73 of file FastMath.h.

6.1.3.3 static FloatType openEV::FastMath::fastSinPositive (FloatType *angle*) [inline],[static],[protected]

Parameters

<i>angle[in]</i>	in degrees. The angle MUST be ≥ 0 .
------------------	--

Returns

The sine value of the angle

Definition at line 122 of file FastMath.h.

6.1.3.4 `static FloatType openEV::FastMath::fastSinRaw (FloatType angle)` `[inline]`, `[static]`, `[protected]`

Parameters

<i>angle[in]</i>	in degrees. The angle <i>must</i> ≥ 0.0 and < 360.0
------------------	--

Returns

The sine value of the angle

Definition at line 103 of file FastMath.h.

6.1.4 Member Data Documentation

6.1.4.1 `constexpr double openEV::FastMath::degToRad = M_PI / 180.0` `[static]`

Definition at line 61 of file FastMath.h.

6.1.4.2 `constexpr double openEV::FastMath::radToDeg = 180.0 / M_PI` `[static]`

Definition at line 60 of file FastMath.h.

6.1.4.3 `constexpr unsigned openEV::FastMath::sineSamplesPerDegree = 8` `[static]`

the sinus table is calculated in 1/8 degree steps

Definition at line 58 of file FastMath.h.

6.1.4.4 `const double openEV::FastMath::sinusTable` `[static]`, `[protected]`

The table of pre-computed sinus values. The table is one item longer than `sizeSineTable` because I need the interpolation to +360 degrees!

Generated by [genSineTables.cpp](#).

Definition at line 96 of file FastMath.h.

6.1.4.5 `constexpr unsigned openEV::FastMath::sizeSineTable = 360*sineSamplesPerDegree` `[static]`

the sinus table is calculated in 1/8 degree steps

Definition at line 59 of file FastMath.h.

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

- [src/FastMath.h](#)
- [src/FastMath.cpp](#)
- [src/FastMathSineTable.cpp](#)

6.2 openEV::GliderVarioMeasurementMatrix Class Reference

```
#include <GliderVarioMeasurementMatrix.h>
```

Public Types

- typedef Eigen::Matrix< [FloatType](#), [GliderVarioMeasurementVector::MEASURE_NUM_ROWS](#), [GliderVarioStatus::STATUS_NUM_ROWS](#) > [MeasureMatrixType](#)
Multiplication matrix. Dimensions come directly from the status and measurement vector sizes.

Public Member Functions

- [GliderVarioMeasurementMatrix](#) ()
- virtual [~GliderVarioMeasurementMatrix](#) ()
- [MeasureMatrixType](#) const [getMeasureMatrix](#) () const
- void [calcMeasurementMatrix](#) ([FloatType](#) timeDiff, [GliderVarioStatus](#) const &lastStatus)

Protected Attributes

- [MeasureMatrixType](#) [measurementMatrix](#)

6.2.1 Detailed Description

Definition at line 19 of file [GliderVarioMeasurementMatrix.h](#).

6.2.2 Member Typedef Documentation

6.2.2.1 typedef Eigen::Matrix<[FloatType](#),[GliderVarioMeasurementVector::MEASURE_NUM_ROWS](#),[GliderVarioStatus::STATUS_NUM_ROWS](#)> [openEV::GliderVarioMeasurementMatrix::MeasureMatrixType](#)

Multiplication matrix. Dimensions come directly from the status and measurement vector sizes.

Definition at line 25 of file [GliderVarioMeasurementMatrix.h](#).

6.2.3 Constructor & Destructor Documentation

6.2.3.1 [openEV::GliderVarioMeasurementMatrix::GliderVarioMeasurementMatrix](#) ()

Definition at line 12 of file [GliderVarioMeasurementMatrix.cpp](#).

6.2.3.2 [openEV::GliderVarioMeasurementMatrix::~~GliderVarioMeasurementMatrix](#) () [virtual]

Definition at line 41 of file [GliderVarioMeasurementMatrix.cpp](#).

6.2.4 Member Function Documentation

6.2.4.1 void [openEV::GliderVarioMeasurementMatrix::calcMeasurementMatrix](#) ([FloatType](#) timeDiff, [GliderVarioStatus](#) const &lastStatus)

Definition at line 46 of file [GliderVarioMeasurementMatrix.cpp](#).

6.2.4.2 MeasureMatrixType const openEV::GliderVarioMeasurementMatrix::getMeasureMatrix () const [inline]

Definition at line 27 of file GliderVarioMeasurementMatrix.h.

6.2.5 Member Data Documentation

6.2.5.1 MeasureMatrixType openEV::GliderVarioMeasurementMatrix::measurementMatrix [protected]

Definition at line 44 of file GliderVarioMeasurementMatrix.h.

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

- [src/GliderVarioMeasurementMatrix.h](#)
- [src/GliderVarioMeasurementMatrix.cpp](#)

6.3 openEV::GliderVarioMeasurementVector Class Reference

```
#include <GliderVarioMeasurementVector.h>
```

Public Types

- enum [MeasureComponentIndex](#) {
[MEASURE_IND_GPS_LAT](#), [MEASURE_IND_GPS_LON](#), [MEASURE_IND_GPS_ALTMSL](#), [MEASURE_IND_GPS_HEADING](#),
[MEASURE_IND_GPS_SPEED](#), [MEASURE_IND_ACC_X](#), [MEASURE_IND_ACC_Y](#), [MEASURE_IND_ACC_Z](#),
[MEASURE_IND_GYRO_RATE_X](#), [MEASURE_IND_GYRO_RATE_Y](#), [MEASURE_IND_GYRO_RATE_Z](#),
[MEASURE_IND_MAG_X](#),
[MEASURE_IND_MAG_Y](#), [MEASURE_IND_MAG_Z](#), [MEASURE_IND_PRESS_ALT](#), [MEASURE_IND_TAS](#),
[MEASURE_NUM_ROWS](#) }
- typedef Eigen::Matrix< [FloatType](#), [MEASURE_NUM_ROWS](#), 1 > [MeasureVectorType](#)

Public Member Functions

- [GliderVarioMeasurementVector](#) ()
- virtual [~GliderVarioMeasurementVector](#) ()
- [MeasureVectorType](#) const [getMeasureVector](#) () const

Public Attributes

- [FloatType](#) & [gpsLatitude](#) = [measureVector](#) [[MEASURE_IND_GPS_LAT](#)]
Latitude in Deg.
- [FloatType](#) & [gpsLongitude](#) = [measureVector](#) [[MEASURE_IND_GPS_LON](#)]
Longitude in Deg.
- [FloatType](#) & [gpsMSL](#) = [measureVector](#) [[MEASURE_IND_GPS_ALTMSL](#)]
Altitude MSL in m.
- [FloatType](#) & [gpsHeading](#) = [measureVector](#) [[MEASURE_IND_GPS_HEADING](#)]
Heading in Deg.
- [FloatType](#) & [gpsSpeed](#) = [measureVector](#) [[MEASURE_IND_GPS_SPEED](#)]
Speed in knots.
- [FloatType](#) & [accelX](#) = [measureVector](#) [[MEASURE_IND_ACC_X](#)]

- Acceleration along the X axis in m/s^2 .*
 - `FloatType & accelY = measureVector [MEASURE_IND_ACC_Y]`
- Acceleration along the Y axis in m/s^2 .*
 - `FloatType & accelZ = measureVector [MEASURE_IND_ACC_Z]`
- Acceleration along the Z axis in m/s^2 .*
 - `FloatType & gyroRateX = measureVector [MEASURE_IND_GYRO_RATE_X]`
- Turn rate around the X axis in Deg/s.*
 - `FloatType & gyroRateY = measureVector [MEASURE_IND_GYRO_RATE_Y]`
- Turn rate around the Y axis in Deg/s.*
 - `FloatType & gyroRateZ = measureVector [MEASURE_IND_GYRO_RATE_Z]`
- Turn rate around the Z axis in Deg/s.*
 - `FloatType & magX = measureVector [MEASURE_IND_MAG_X]`
- magnetic field strength along X axis in uT (absolute strength is irrelevant, only used to determine attitude)*
 - `FloatType & magY = measureVector [MEASURE_IND_MAG_Y]`
- magnetic field strength along Y axis in uT (absolute strength is irrelevant, only used to determine attitude)*
 - `FloatType & magZ = measureVector [MEASURE_IND_MAG_Z]`
- magnetic field strength along Z axis in uT (absolute strength is irrelevant, only used to determine attitude)*
 - `FloatType & pressAlt = measureVector [MEASURE_IND_PRESS_ALT]`
- pressure altitude in MSL*
 - `FloatType & trueAirSpeed = measureVector [MEASURE_IND_TAS]`
- True air speed (based on difference pressure and air density based on absolute pressure) in m/s.*

Protected Attributes

- `MeasureVectorType measureVector`
holder of the vector

6.3.1 Detailed Description

This is the measurement input vector into the Kalman filter. Not all measurements are the raw instrument readings. Particularly pressure readings are converted into altitude and speed before because the conversions are highly non-linear. Otherwise all units are converted to ISO base units. Absolute Magnetometer readings are irrelevant but their ratios are used to estimate the attitude.

Definition at line 41 of file `GliderVarioMeasurementVector.h`.

6.3.2 Member Typedef Documentation

- 6.3.2.1 `typedef Eigen::Matrix<FloatType, MEASURE_NUM_ROWS, 1> openEV::GliderVarioMeasurementVector::MeasureVectorType`↔

Definition at line 79 of file `GliderVarioMeasurementVector.h`.

6.3.3 Member Enumeration Documentation

- 6.3.3.1 `enum openEV::GliderVarioMeasurementVector::MeasureComponentIndex`

Enumerator

- `MEASURE_IND_GPS_LAT`** Latitude in Deg.
- `MEASURE_IND_GPS_LON`** Longitude in Deg.
- `MEASURE_IND_GPS_ALTMSL`** Altitude MSL in m.

MEASURE_IND_GPS_HEADING Heading in Deg.

MEASURE_IND_GPS_SPEED Speed in knots.

MEASURE_IND_ACC_X Acceleration along the X axis in m/s^2 .

MEASURE_IND_ACC_Y Acceleration along the Y axis in m/s^2 .

MEASURE_IND_ACC_Z Acceleration along the Z axis in m/s^2 .

MEASURE_IND_GYRO_RATE_X Turn rate around the X axis in Deg/s.

MEASURE_IND_GYRO_RATE_Y Turn rate around the Y axis in Deg/s.

MEASURE_IND_GYRO_RATE_Z Turn rate around the Z axis in Deg/s.

MEASURE_IND_MAG_X magnetic field strength along X axis in uT (absolute strength is irrelevant, only used to determine attitude)

MEASURE_IND_MAG_Y magnetic field strength along Y axis in uT (absolute strength is irrelevant, only used to determine attitude)

MEASURE_IND_MAG_Z magnetic field strength along Z axis in uT (absolute strength is irrelevant, only used to determine attitude)

MEASURE_IND_PRESS_ALT pressure altitude in MSL

MEASURE_IND_TAS True air speed (based on difference pressure and air density based on absolute pressure) in m/s.

MEASURE_NUM_ROWS

Definition at line 49 of file GliderVarioMeasurementVector.h.

6.3.4 Constructor & Destructor Documentation

6.3.4.1 `openEV::GliderVarioMeasurementVector::GliderVarioMeasurementVector ()` `[inline]`

Definition at line 43 of file GliderVarioMeasurementVector.h.

6.3.4.2 `openEV::GliderVarioMeasurementVector::~~GliderVarioMeasurementVector ()` `[virtual]`

Definition at line 31 of file GliderVarioMeasurementVector.cpp.

6.3.5 Member Function Documentation

6.3.5.1 `MeasureVectorType const openEV::GliderVarioMeasurementVector::getMeasureVector () const` `[inline]`

Definition at line 107 of file GliderVarioMeasurementVector.h.

6.3.6 Member Data Documentation

6.3.6.1 `FloatType& openEV::GliderVarioMeasurementVector::accelX = measureVector [MEASURE_IND_ACC_X]`

Acceleration along the X axis in m/s^2 .

Definition at line 89 of file GliderVarioMeasurementVector.h.

6.3.6.2 `FloatType& openEV::GliderVarioMeasurementVector::accelY = measureVector [MEASURE_IND_ACC_Y]`

Acceleration along the Y axis in m/s^2 .

Definition at line 90 of file GliderVarioMeasurementVector.h.

6.3.6.3 FloatType& openEV::GliderVarioMeasurementVector::accelZ = measureVector [MEASURE_IND_ACC_Z]

Acceleration along the Z axis in m/s^2 .

Definition at line 91 of file GliderVarioMeasurementVector.h.

6.3.6.4 FloatType& openEV::GliderVarioMeasurementVector::gpsHeading = measureVector [MEASURE_IND_GPS_HEADING]

Heading in Deg.

Definition at line 85 of file GliderVarioMeasurementVector.h.

6.3.6.5 FloatType& openEV::GliderVarioMeasurementVector::gpsLatitude = measureVector [MEASURE_IND_GPS_LAT]

Latitude in Deg.

Definition at line 82 of file GliderVarioMeasurementVector.h.

6.3.6.6 FloatType& openEV::GliderVarioMeasurementVector::gpsLongitude = measureVector [MEASURE_IND_GPS_LON]

Longitude in Deg.

Definition at line 83 of file GliderVarioMeasurementVector.h.

6.3.6.7 FloatType& openEV::GliderVarioMeasurementVector::gpsMSL = measureVector [MEASURE_IND_GPS_ALTMSL]

Altitude MSL in m.

Definition at line 84 of file GliderVarioMeasurementVector.h.

6.3.6.8 FloatType& openEV::GliderVarioMeasurementVector::gpsSpeed = measureVector [MEASURE_IND_GPS_SPEED]

Speed in knots.

Definition at line 86 of file GliderVarioMeasurementVector.h.

6.3.6.9 FloatType& openEV::GliderVarioMeasurementVector::gyroRateX = measureVector [MEASURE_IND_GYRO_RATE_X]

Turn rate around the X axis in Deg/s.

Definition at line 94 of file GliderVarioMeasurementVector.h.

6.3.6.10 FloatType& openEV::GliderVarioMeasurementVector::gyroRateY = measureVector [MEASURE_IND_GYRO_RATE_Y]

Turn rate around the Y axis in Deg/s.

Definition at line 95 of file GliderVarioMeasurementVector.h.

6.3.6.11 FloatType& openEV::GliderVarioMeasurementVector::gyroRateZ = measureVector [MEASURE_IND_GYRO_RATE_Z]

Turn rate around the Z axis in Deg/s.

Definition at line 96 of file GliderVarioMeasurementVector.h.

6.3.6.12 FloatType& openEV::GliderVarioMeasurementVector::magX = measureVector [MEASURE_IND_MAG_X]

magnetic field strength along X axis in uT (absolute strength is irrelevant, only used to determine attitude)

Definition at line 99 of file GliderVarioMeasurementVector.h.

6.3.6.13 FloatType& openEV::GliderVarioMeasurementVector::magY = measureVector [MEASURE_IND_MAG_Y]

magnetic field strength along Y axis in uT (absolute strength is irrelevant, only used to determine attitude)

Definition at line 100 of file GliderVarioMeasurementVector.h.

6.3.6.14 FloatType& openEV::GliderVarioMeasurementVector::magZ = measureVector [MEASURE_IND_MAG_Z]

magnetic field strength along Z axis in uT (absolute strength is irrelevant, only used to determine attitude)

Definition at line 101 of file GliderVarioMeasurementVector.h.

6.3.6.15 MeasureVectorType openEV::GliderVarioMeasurementVector::measureVector [protected]

holder of the vector

Definition at line 112 of file GliderVarioMeasurementVector.h.

6.3.6.16 FloatType& openEV::GliderVarioMeasurementVector::pressAlt = measureVector [MEASURE_IND_PRESS_ALT]

pressure altitude in MSL

Definition at line 104 of file GliderVarioMeasurementVector.h.

6.3.6.17 FloatType& openEV::GliderVarioMeasurementVector::trueAirSpeed = measureVector [MEASURE_IND_TAS]

True air speed (based on difference pressure and air density based on absolute pressure) in m/s.

Definition at line 105 of file GliderVarioMeasurementVector.h.

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

- [src/GliderVarioMeasurementVector.h](#)
- [src/GliderVarioMeasurementVector.cpp](#)

6.4 openEV::GliderVarioStatus Class Reference

[GliderVarioStatus](#) manages the Kalman filter state x.

```
#include <GliderVarioStatus.h>
```

Public Types

- enum `StatusComponentIndex` {
`STATUS_IND_LONGITUDE`, `STATUS_IND_LATITUDE`, `STATUS_IND_ALT_MSL`, `STATUS_IND_YAW`,
`STATUS_IND_PITCH`, `STATUS_IND_ROLL`, `STATUS_IND_SPEED_GROUND_N`, `STATUS_IND_SPEED_GROUND_E`,
`STATUS_IND_TAS`, `STATUS_IND_HEADING`, `STATUS_IND_RATE_OF_SINK`, `STATUS_IND_VERTICAL_SPEED`,
`STATUS_IND_ACC_X`, `STATUS_IND_ACC_Y`, `STATUS_IND_ACC_Z`, `STATUS_IND_ROTATION_X`,
`STATUS_IND_ROTATION_Y`, `STATUS_IND_ROTATION_Z`, `STATUS_IND_GYRO_BIAS_X`, `STATUS_IND_GYRO_BIAS_Y`,
`STATUS_IND_GYRO_BIAS_Z`, `STATUS_IND_WIND_SPEED_N`, `STATUS_IND_WIND_SPEED_E`, `STATUS_IND_THERMAL_SPEED`,
`STATUS_NUM_ROWS` }

Index, i.e. positions of the status components in the status vector.

- typedef `Eigen::Matrix< FloatType, STATUS_NUM_ROWS, 1 >` `StatusVectorType`

Saves typing of the complex template type.

Public Member Functions

- `GliderVarioStatus` ()
- virtual `~GliderVarioStatus` ()
- `StatusVectorType` & `getStatusVector` ()
- `StatusVectorType` const & `getStatusVector` () const
- void `normalizeAngles` ()

Public Attributes

- `FloatType` & `longitude` = `statusVector[STATUS_IND_LONGITUDE]`
Longitude in deg. East.
- `FloatType` & `latitude` = `statusVector[STATUS_IND_LATITUDE]`
Latitude in deg North.
- `FloatType` & `altMSL` = `statusVector[STATUS_IND_ALT_MSL]`
Altitude in m over Mean Sea Level.
- `FloatType` & `yawAngle` = `statusVector[STATUS_IND_YAW]`
Yaw angle in deg. right turn from true North.
- `FloatType` & `pitchAngle` = `statusVector[STATUS_IND_PITCH]`
Pitch angle in deg. nose up. Pitch is applied after yaw.
- `FloatType` & `rollAngle` = `statusVector[STATUS_IND_ROLL]`
Roll angle in deg. right. Roll is applied after yaw and pitch.
- `FloatType` & `groundSpeedNorth` = `statusVector[STATUS_IND_SPEED_GROUND_N]`
Ground speed component North in m/s.
- `FloatType` & `groundSpeedEast` = `statusVector[STATUS_IND_SPEED_GROUND_E]`
Ground speed component East in m/s.
- `FloatType` & `trueAirSpeed` = `statusVector[STATUS_IND_TAS]`
True air speed in m/s relative to surrounding air.
- `FloatType` & `heading` = `statusVector[STATUS_IND_HEADING]`
Heading of the plane in deg. right turn from true north. This is the flight direction relative to the surrounding air.
- `FloatType` & `rateOfSink` = `statusVector[STATUS_IND_RATE_OF_SINK]`
Rate of sink in m/s relative to the surrounding air. Sink because the Z axis points downward.
- `FloatType` & `verticalSpeed` = `statusVector[STATUS_IND_VERTICAL_SPEED]`
Absolute vertical speed in m/s downward. Z axis is downward.

- `FloatType & accelX = statusVector[STATUS_IND_ACC_X]`
Acceleration in m/s^2 on the X axis of the plane.
- `FloatType & accelY = statusVector[STATUS_IND_ACC_Y]`
Acceleration in m/s^2 on the Y axis of the plane.
- `FloatType & accelZ = statusVector[STATUS_IND_ACC_Z]`
Acceleration in m/s^2 on the Z axis of the plane.
- `FloatType & rollRateX = statusVector[STATUS_IND_ROTATION_X]`
Roll rate in deg/s to the right around the X axis.
- `FloatType & pitchRateY = statusVector[STATUS_IND_ROTATION_Y]`
Pitch rate in deg/s nose up around the Y axis.
- `FloatType & yawRateZ = statusVector[STATUS_IND_ROTATION_Z]`
Yaw (turn) rate in deg/s around the Z axis.
- `FloatType & gyroBiasX = statusVector[STATUS_IND_GYRO_BIAS_X]`
Bias (0-offset) of the X axis gyro in deg/s.
- `FloatType & gyroBiasY = statusVector[STATUS_IND_GYRO_BIAS_Y]`
Bias (0-offset) of the Y axis gyro in deg/s.
- `FloatType & gyroBiasZ = statusVector[STATUS_IND_GYRO_BIAS_Z]`
Bias (0-offset) of the Z axis gyro in deg/s.
- `FloatType & windSpeedNorth = statusVector[STATUS_IND_WIND_SPEED_N]`
Wind speed North component in m/s.
- `FloatType & windSpeedEast = statusVector[STATUS_IND_WIND_SPEED_E]`
The direction is the direction from where the wind blows.
- `FloatType & thermalSpeed = statusVector[STATUS_IND_THERMAL_SPEED]`
The true reason for the whole exercise! :)

Protected Attributes

- `StatusVectorType statusVector`

6.4.1 Detailed Description

`GliderVarioStatus` manages the Kalman filter state x .

The class defines the Kalman filter status x as a vector of floats or doubles. Each component of the status vector is clearly identified by the index in the vector. The indexes are enumerated in the `StatusComponentIndex` enum. The components and index enumerators of the status vector are as follows:

Worldwide Position:

- Longitude `STATUS_IND_LONGITUDE`: **Longitude** in decimal degrees. Eastern hemisphere is positive, western hemisphere is negative.
- Latitude `STATUS_IND_LATITUDE`: **Latitude** in decimal degrees. Northern hemisphere is positive, southern hemisphere is negative.
- Altitude MSL `STATUS_IND_ALT_MSL`: **Altitude** above MSL in m(eter).

Attitude:

- Yaw angle `STATUS_IND_YAW`: **Yaw** angle in Degrees to the right of true North. Also known as **Heading**
- Pitch angle `STATUS_IND_PITCH`: **Pitch** angle in Degrees nose upward. 0 = horizontal flight. Also known as **Elevation**.
- Roll angle `STATUS_IND_ROLL`: **Roll** angle in degrees right. Left roll is negative. Also known as **Bank**.

Speeds and directions

- Ground speed STATUS_IND_SPEED_GROUND **Ground Speed** in m/s
- Direction over ground STATUS_IND_DIR_GROUND **Flight Direction over ground** in Degrees to the right to true North.
- True air speed STATUS_IND_TAS **True Air Speed** in m/s. Speed relative to the surrounding air
- Plane heading STATUS_IND_HEADING **True Heading of the plane**. I assume that the heading is equal to my movement vector in the air, i.e. I assume that I am not slipping.
- Plane rate of Climb STATUS_IND_RATE_OF_CLIMB **Rate of Climb** of the air plane relative to the air in m/s. Up is positive. This is kind of my stick thermals. STATUS_IND_VERTICAL_SPEED and Rate of climb are identical in stagnant air.
- Absolute vertical speed STATUS_IND_VERTICAL_SPEED **Absolute vertical speed** in m/s

Accelerations in reference to the body coordinate system

- Accel X axis STATUS_IND_ACC_X **Acceleration along X axis** in m/s^2
- Accel Y axis STATUS_IND_ACC_Y **Acceleration along Y axis** in m/s^2
- Accel Z axis STATUS_IND_ACC_Z **Acceleration along Y axis** in m/s^2

Turn rates in reference to the body coordinate system

- Rotation around X axis **Rotation around X axis** in degrees per second
- Rotation around Y axis **Rotation around Y axis** in degrees per second
- Rotation around Z axis **Rotation around Z axis** in degrees per second

Derived values which improve the responsiveness of the Kalman filter

- Gyro X bias STATUS_IND_GYRO_BIAS_X **Gyro X axis bias** Gyros tend to have a bias, i.e an offset of the 0-value. The bias is not constant but varies over time. Tracking it helps to make the filter more responsive
- Gyro Y bias STATUS_IND_GYRO_BIAS_Y **Gyro Y axis bias**
- Gyro Z bias STATUS_IND_GYRO_BIAS_Z **Gyro Z axis bias**
- Wind speed STATUS_IND_WIND_SPEED **Wind Speed** in m/s
- Wind direction STATUS_IND_WIND_DIR **Wind Direction** in Degrees, STATUS_IND_DIR_GROUND
- Thermal speed STATUS_IND_THERMAL_SPEED The real thermal updraft in m/s

Definition at line 105 of file GliderVarioStatus.h.

6.4.2 Member Typedef Documentation

6.4.2.1 `typedef Eigen::Matrix<FloatType,STATUS_NUM_ROWS,1> openEV::GliderVarioStatus::StatusVector`↔ Type

Saves typing of the complex template type.

Definition at line 157 of file GliderVarioStatus.h.

6.4.3 Member Enumeration Documentation

6.4.3.1 enum openEV::GliderVarioStatus::StatusComponentIndex

Index, i.e. positions of the status components in the status vector.

Enumeration of the components of the Kalman status vector x

Enumerator

STATUS_IND_LONGITUDE Position and attitude. Longitude in deg. East

STATUS_IND_LATITUDE Latitude in deg North.

STATUS_IND_ALT_MSL Altitude in m over Mean Sea Level.

STATUS_IND_YAW Yaw angle in deg. right turn from true North.

STATUS_IND_PITCH Pitch angle in deg. nose up. Pitch is applied after yaw.

STATUS_IND_ROLL Roll angle in deg. right. Roll is applied after yaw and pitch.

STATUS_IND_SPEED_GROUND_N Speeds and directions. Ground speed component North in m/s

STATUS_IND_SPEED_GROUND_E Ground speed component East in m/s.

STATUS_IND_TAS True air speed in m/s relative to surrounding air.

STATUS_IND_HEADING Heading of the plane in deg. right turn from true north. This is the flight direction relative to the surrounding air.

STATUS_IND_RATE_OF_SINK Rate of sink in m/s relative to the surrounding air. Sink because the y axis points downward.

STATUS_IND_VERTICAL_SPEED Absolute vertical speed in m/s downward. Z axis is direction down.

STATUS_IND_ACC_X Acceleration in m/s^2 on the X axis of the plane. Accelerations in reference to the body coordinate system. Accelerations are on the axis of the *plane*. If the plane is pitched up an acceleration on the X axis would speed the plane upward, not forward.

STATUS_IND_ACC_Y Acceleration in m/s^2 on the Y axis of the plane.

STATUS_IND_ACC_Z Acceleration in m/s^2 on the Z axis of the plane.

STATUS_IND_ROTATION_X Turn rates in reference to the body coordinate system. Roll rate in deg/s to the right around the X axis

STATUS_IND_ROTATION_Y Pitch rate in deg/s nose up around the Y axis.

STATUS_IND_ROTATION_Z Yaw (turn) rate in deg/s around the Z axis.

STATUS_IND_GYRO_BIAS_X Derived values which improve the responsiveness of the Kalman filter. Some are also the true goals of the filter. Bias (0-offset) of the X axis gyro in deg/s

STATUS_IND_GYRO_BIAS_Y Bias (0-offset) of the Y axis gyro in deg/s.

STATUS_IND_GYRO_BIAS_Z Bias (0-offset) of the Z axis gyro in deg/s.

STATUS_IND_WIND_SPEED_N Wind speed North component in m/s.

STATUS_IND_WIND_SPEED_E The direction is the direction *from where* the wind blows. Wind speed East component in m/s

STATUS_IND_THERMAL_SPEED The true reason for the whole exercise! :)

STATUS_NUM_ROWS The number of rows in the vector.

Definition at line 113 of file GliderVarioStatus.h.

6.4.4 Constructor & Destructor Documentation

6.4.4.1 openEV::GliderVarioStatus::GliderVarioStatus ()

Definition at line 33 of file GliderVarioStatus.cpp.

6.4.4.2 `openEV::GliderVarioStatus::~~GliderVarioStatus () [virtual]`

Definition at line 39 of file `GliderVarioStatus.cpp`.

6.4.5 Member Function Documentation

6.4.5.1 `StatusVectorType& openEV::GliderVarioStatus::getStatusVector () [inline]`

Definition at line 163 of file `GliderVarioStatus.h`.

6.4.5.2 `StatusVectorType const& openEV::GliderVarioStatus::getStatusVector () const [inline]`

Definition at line 167 of file `GliderVarioStatus.h`.

6.4.5.3 `void openEV::GliderVarioStatus::normalizeAngles ()`

Updating the status may lead to wrap-around of angles. Here are the limits: -Pitch: $90 \leq \text{pitch} \leq 90$; If you fly a looping and turn past perpendicular you essentially roll 180 deg, and reverse direction 180 deg -Roll: $-180 \leq \text{roll} < 180$; 180 deg counts as -180 -Yaw: $0 \leq \text{yaw} < 360$; 360 deg counts as 0. Note that pitch must be normalized first. It may flip roll and yaw around. Yaw and roll are independent from the other angles.

Definition at line 44 of file `GliderVarioStatus.cpp`.

6.4.6 Member Data Documentation

6.4.6.1 `FloatType& openEV::GliderVarioStatus::accelX = statusVector[STATUS_IND_ACC_X]`

Acceleration in m/s^2 on the X axis of the plane.

Definition at line 200 of file `GliderVarioStatus.h`.

6.4.6.2 `FloatType& openEV::GliderVarioStatus::accelY = statusVector[STATUS_IND_ACC_Y]`

Acceleration in m/s^2 on the Y axis of the plane.

Definition at line 201 of file `GliderVarioStatus.h`.

6.4.6.3 `FloatType& openEV::GliderVarioStatus::accelZ = statusVector[STATUS_IND_ACC_Z]`

Acceleration in m/s^2 on the Z axis of the plane.

Definition at line 202 of file `GliderVarioStatus.h`.

6.4.6.4 `FloatType& openEV::GliderVarioStatus::altMSL = statusVector[STATUS_IND_ALT_MSL]`

Altitude in m over Mean Sea Level.

Definition at line 185 of file `GliderVarioStatus.h`.

6.4.6.5 `FloatType& openEV::GliderVarioStatus::groundSpeedEast = statusVector[STATUS_IND_SPEED_GROUND_E]`

Ground speed component East in m/s.

Definition at line 192 of file `GliderVarioStatus.h`.

6.4.6.6 `FloatType& openEV::GliderVarioStatus::groundSpeedNorth = statusVector[STATUS_IND_SPEED_GROUND_N]`

Ground speed component North in m/s.

Definition at line 191 of file GliderVarioStatus.h.

6.4.6.7 `FloatType& openEV::GliderVarioStatus::gyroBiasX = statusVector[STATUS_IND_GYRO_BIAS_X]`

Bias (0-offset) of the X axis gyro in deg/s.

Definition at line 210 of file GliderVarioStatus.h.

6.4.6.8 `FloatType& openEV::GliderVarioStatus::gyroBiasY = statusVector[STATUS_IND_GYRO_BIAS_Y]`

Bias (0-offset) of the Y axis gyro in deg/s.

Definition at line 211 of file GliderVarioStatus.h.

6.4.6.9 `FloatType& openEV::GliderVarioStatus::gyroBiasZ = statusVector[STATUS_IND_GYRO_BIAS_Z]`

Bias (0-offset) of the Z axis gyro in deg/s.

Definition at line 212 of file GliderVarioStatus.h.

6.4.6.10 `FloatType& openEV::GliderVarioStatus::heading = statusVector[STATUS_IND_HEADING]`

Heading of the plane in deg. right turn from true north. This is the flight direction relative to the surrounding air.

Definition at line 194 of file GliderVarioStatus.h.

6.4.6.11 `FloatType& openEV::GliderVarioStatus::latitude = statusVector[STATUS_IND_LATITUDE]`

Latitude in deg North.

Definition at line 184 of file GliderVarioStatus.h.

6.4.6.12 `FloatType& openEV::GliderVarioStatus::longitude = statusVector[STATUS_IND_LONGITUDE]`

Longitude in deg. East.

Definition at line 183 of file GliderVarioStatus.h.

6.4.6.13 `FloatType& openEV::GliderVarioStatus::pitchAngle = statusVector[STATUS_IND_PITCH]`

Pitch angle in deg. nose up. Pitch is applied after yaw.

Definition at line 187 of file GliderVarioStatus.h.

6.4.6.14 `FloatType& openEV::GliderVarioStatus::pitchRateY = statusVector[STATUS_IND_ROTATION_Y]`

Pitch rate in deg/s nose up around the Y axis.

Definition at line 206 of file GliderVarioStatus.h.

6.4.6.15 FloatType& openEV::GliderVarioStatus::rateOfSink = statusVector[STATUS_IND_RATE_OF_SINK]

Rate of sink in m/s relative to the surrounding air. Sink because the Z axis points downward.

Definition at line 195 of file GliderVarioStatus.h.

6.4.6.16 FloatType& openEV::GliderVarioStatus::rollAngle = statusVector[STATUS_IND_ROLL]

Roll angle in deg. right. Roll is applied after yaw and pitch.

Definition at line 188 of file GliderVarioStatus.h.

6.4.6.17 FloatType& openEV::GliderVarioStatus::rollRateX = statusVector[STATUS_IND_ROTATION_X]

Roll rate in deg/s to the right around the X axis.

Definition at line 205 of file GliderVarioStatus.h.

6.4.6.18 StatusVectorType openEV::GliderVarioStatus::statusVector [protected]

Definition at line 219 of file GliderVarioStatus.h.

6.4.6.19 FloatType& openEV::GliderVarioStatus::thermalSpeed = statusVector[STATUS_IND_THERMAL_SPEED]

The true reason for the whole exercise! :)

Definition at line 216 of file GliderVarioStatus.h.

6.4.6.20 FloatType& openEV::GliderVarioStatus::trueAirSpeed = statusVector[STATUS_IND_TAS]

True air speed in m/s relative to surrounding air.

Definition at line 193 of file GliderVarioStatus.h.

6.4.6.21 FloatType& openEV::GliderVarioStatus::verticalSpeed = statusVector[STATUS_IND_VERTICAL_SPEED]

Absolute vertical speed in m/s downward. Z axis is downward.

Definition at line 196 of file GliderVarioStatus.h.

6.4.6.22 FloatType& openEV::GliderVarioStatus::windSpeedEast = statusVector[STATUS_IND_WIND_SPEED_E]

The direction is the direction *from where* the wind blows.

Wind speed East component in m/s

Definition at line 214 of file GliderVarioStatus.h.

6.4.6.23 FloatType& openEV::GliderVarioStatus::windSpeedNorth = statusVector[STATUS_IND_WIND_SPEED_N]

Wind speed North component in m/s.

Definition at line 213 of file GliderVarioStatus.h.

6.4.6.24 FloatType& openEV::GliderVarioStatus::yawAngle = statusVector[STATUS_IND_YAW]

Yaw angle in deg. right turn from true North.

Definition at line 186 of file GliderVarioStatus.h.

6.4.6.25 FloatType& openEV::GliderVarioStatus::yawRateZ = statusVector[STATUS_IND_ROTATION_Z]

Yaw (turn) rate in deg/s around the Z axis.

Definition at line 207 of file GliderVarioStatus.h.

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

- [src/GliderVarioStatus.h](#)
- [src/GliderVarioStatus.cpp](#)

6.5 openEV::GliderVarioTransitionMatrix Class Reference

```
#include <GliderVarioTransitionMatrix.h>
```

Public Types

- typedef Eigen::Matrix< [FloatType](#), [GliderVarioStatus::STATUS_NUM_ROWS](#), [GliderVarioStatus::STATUS_NUM_ROWS](#) > [TransitionMatrixType](#)

Public Member Functions

- [GliderVarioTransitionMatrix](#) ()
- virtual [~GliderVarioTransitionMatrix](#) ()
- [TransitionMatrixType](#) & [getTransitionMatrix](#) ()
- void [calcTransitionMatrix](#) ([FloatType](#) timeDiff, [GliderVarioStatus](#) const &lastStatus)
- void [updateStatus](#) ([GliderVarioStatus](#) const &oldStatus, [GliderVarioStatus](#) &newStatus, [FloatType](#) timeDiff)

Protected Attributes

- [TransitionMatrixType](#) [transitionMatrix](#)

6.5.1 Detailed Description

This is the transition matrix implementation of the Kalman filter. The transition matrix is re-calculated before every update step because it depends on the elapsed interval, and on the current attitude (i.e. heading pitch and roll affect the TAS vs speed and course over ground).

Definition at line 50 of file GliderVarioTransitionMatrix.h.

6.5.2 Member Typedef Documentation

6.5.2.1 typedef Eigen::Matrix<FloatType,GliderVarioStatus::STATUS_NUM_ROWS,GliderVarioStatus::STATUS_NUM_ROWS> openEV::GliderVarioTransitionMatrix::TransitionMatrixType

Definition at line 53 of file GliderVarioTransitionMatrix.h.

6.5.3 Constructor & Destructor Documentation

6.5.3.1 `openEV::GliderVarioTransitionMatrix::GliderVarioTransitionMatrix ()` `[inline]`

Definition at line 56 of file `GliderVarioTransitionMatrix.h`.

6.5.3.2 `openEV::GliderVarioTransitionMatrix::~~GliderVarioTransitionMatrix ()` `[virtual]`

Definition at line 40 of file `GliderVarioTransitionMatrix.cpp`.

6.5.4 Member Function Documentation

6.5.4.1 `void openEV::GliderVarioTransitionMatrix::calcTransitionMatrix (FloatType timeDiff, GliderVarioStatus const & lastStatus)`

Recalculates the transition matrix. Only active coefficients are recalculated. All other coefficients are supposed to be 0 as they were set at construction time.

Parameters

in	<i>timeDiff</i>	Time since last update in seconds.
in	<i>lastStatus</i>	Most recent status vector. Used to convert world into local coordinates.

Todo Calculation of Rate of Sink: Refine the vario compensation by considering the decrease of drag based on the polar.

Definition at line 46 of file `GliderVarioTransitionMatrix.cpp`.

6.5.4.2 `TransitionMatrixType& openEV::GliderVarioTransitionMatrix::getTransitionMatrix ()` `[inline]`

Definition at line 64 of file `GliderVarioTransitionMatrix.h`.

6.5.4.3 `void openEV::GliderVarioTransitionMatrix::updateStatus (GliderVarioStatus const & oldStatus, GliderVarioStatus & newStatus, FloatType timeDiff)` `[inline]`

Extrapolates the newStatus from the oldStatus after timeDiff seconds. internally recalculates the transition matrix.

Parameters

in	<i>oldStatus</i>	Last known status
out	<i>newStatus</i>	New status by extrapolation after timeDiff seconds
in	<i>timeDiff</i>	The time difference in seconds

Definition at line 88 of file `GliderVarioTransitionMatrix.h`.

6.5.5 Member Data Documentation

6.5.5.1 `TransitionMatrixType openEV::GliderVarioTransitionMatrix::transitionMatrix` `[protected]`

Definition at line 103 of file `GliderVarioTransitionMatrix.h`.

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

- [src/GliderVarioTransitionMatrix.h](#)
- [src/GliderVarioTransitionMatrix.cpp](#)

6.6 openEV::MeasureMatrix Class Reference

```
#include <MeasureMatrix.h>
```

Public Member Functions

- [MeasureMatrix](#) ()
- virtual [~MeasureMatrix](#) ()

6.6.1 Detailed Description

Definition at line 32 of file MeasureMatrix.h.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 openEV::MeasureMatrix::MeasureMatrix ()

Definition at line 31 of file MeasureMatrix.cpp.

6.6.2.2 openEV::MeasureMatrix::~~MeasureMatrix () [virtual]

Definition at line 37 of file MeasureMatrix.cpp.

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

- src/[MeasureMatrix.h](#)
- src/[MeasureMatrix.cpp](#)

6.7 openEV::RotationMatrix Class Reference

```
#include <RotationMatrix.h>
```

Public Types

- typedef Eigen::Matrix< [FloatType](#), 3, 3 > [RotationMatrixType](#)

Public Member Functions

- [RotationMatrix](#) ()
Default constructor. Initialized all angles to 0. The rotation matrix is the Identity matrix.
- [RotationMatrix](#) ([FloatType](#) yaw, [FloatType](#) pitch, [FloatType](#) roll)
Constructor with initial angles. The matrix is not yet defined.
- virtual [~RotationMatrix](#) ()
- void [setYaw](#) ([FloatType](#) yaw)
set yaw angle . Invalidates the matrix.
- [FloatType](#) [getYaw](#) ()
- void [setPitch](#) ([FloatType](#) pitch)
set pitch angle . Invalidates the matrix.
- [FloatType](#) [getPitch](#) ()
- void [setRoll](#) ([FloatType](#) roll)

- set roll angle . Invalidates the matrix.*
- [FloatType getRoll \(\)](#)
- void [calcPlaneVectorToWorldVector](#) (const [Vector3DType](#) &planeVector, [Vector3DType](#) &worldVector)
- void [calcWorldVectorToPlaneVector](#) (const [Vector3DType](#) &worldVector, [Vector3DType](#) &planeVector)
- [RotationMatrixType](#) & [getMatrixGloToPlane](#) ()
The rotation matrix from the global(world) coordinate system to the plane coordinate system.
- [RotationMatrixType](#) & [getMatrixPlaneToGlo](#) ()
The rotation matrix from the global(world) coordinate system to the plane coordinate system.

Protected Member Functions

- void [calculateRotationMatrixGloToPlane](#) ()
- void [calculateRotationMatrixPlaneToGlo](#) ()

Protected Attributes

- [RotationMatrixType matrixGloToPlane](#)
The rotation matrix from the global(world) coordinate system to the plane coordinate system.
- [RotationMatrixType matrixPlaneToGlo](#)
The rotation matrix from the global(world) coordinate system to the plane coordinate system.
- bool [matrixGloToPlaneIsValid](#)
- bool [matrixPlaneToGloIsValid](#)
- [FloatType yaw](#)
*Yaw angle in deg. in the norm DIN 9300. Also called **Heading**. Turning right hand around the z axis, i.e. in navigation direction.*
- [FloatType pitch](#)
*Pitch angle in deg. in the norm DIN 9300. Also called **Elevation**. Turning nose up around the y axis is positive.*
- [FloatType roll](#)
*Roll angle in deg. in the norm DIN 9300. Also called **Bank angle**.*

6.7.1 Detailed Description

Definition at line 47 of file RotationMatrix.h.

6.7.2 Member Typedef Documentation

6.7.2.1 `typedef Eigen::Matrix<FloatType, 3, 3> openEV::RotationMatrix::RotationMatrixType`

Definition at line 50 of file RotationMatrix.h.

6.7.3 Constructor & Destructor Documentation

6.7.3.1 `openEV::RotationMatrix::RotationMatrix () [inline]`

Default constructor. Initialized all angles to 0. The rotation matrix is the Identity matrix.

Definition at line 54 of file RotationMatrix.h.

6.7.3.2 `openEV::RotationMatrix::RotationMatrix (FloatType yaw, FloatType pitch, FloatType roll) [inline]`

Constructor with initial angles. The matrix is not yet defined.

Definition at line 69 of file RotationMatrix.h.

6.7.3.3 openEV::RotationMatrix::~~RotationMatrix () [virtual]

Definition at line 33 of file RotationMatrix.cpp.

6.7.4 Member Function Documentation

6.7.4.1 void openEV::RotationMatrix::calcPlaneVectorToWorldVector (const Vector3DType & planeVector, Vector3DType & worldVector) [inline]

Convert the plane vector into the world vector

Parameters

<i>planeVector[in]</i>	The vector in plane coordinates
<i>worldVector[out]</i>	The vector in world coordinates

Definition at line 124 of file RotationMatrix.h.

6.7.4.2 void openEV::RotationMatrix::calculateRotationMatrixGloToPlane () [protected]

Calculates the rotation matrix global to plane is calculated.

Calculates the rotation matrix. The matrix from world coordinates to plane coordinates is calculated only.

Again the the angle definitions:

- Yaw angle = Heading
- Pitch angle = Elevation
- Rollwinkel = Bank angle

Implementing the matrix according to the German Wikipedia https://de.wikipedia.org/wiki/Eulersche_Winkel#Drehfolgen_in_der_Fahrzeugtechnik

$M_{\{G \rightarrow N\}} = \{pmatrix\} \cdot \begin{pmatrix} \cos \alpha & 0 & 0 \\ 0 & \cos \beta & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos \gamma & \sin \gamma & 0 \\ -\sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Definition at line 56 of file RotationMatrix.cpp.

6.7.4.3 void openEV::RotationMatrix::calculateRotationMatrixPlaneToGlo () [inline], [protected]

Calculate the rotation matrix plane to global. Do this by transposing the global to plane matrix.

Definition at line 180 of file RotationMatrix.h.

6.7.4.4 void openEV::RotationMatrix::calcWorldVectorToPlaneVector (const Vector3DType & worldVector, Vector3DType & planeVector) [inline]

Convert the world vector into the plane vector

Parameters

<i>worldVector[in]</i>	The vector in world coordinates
<i>planeVector[out]</i>	The vector in plane coordinates

Definition at line 135 of file RotationMatrix.h.

6.7.4.5 RotationMatrixType& openEV::RotationMatrix::getMatrixGloToPlane () [inline]

The rotation matrix from the global(world) coordinate system to the plane coordinate system.

Definition at line 141 of file RotationMatrix.h.

6.7.4.6 **RotationMatrixType& openEV::RotationMatrix::getMatrixPlaneToGlo ()** `[inline]`

The rotation matrix from the global(world) coordinate system to the plane coordinate system.

Definition at line 146 of file RotationMatrix.h.

6.7.4.7 **FloatType openEV::RotationMatrix::getPitch ()** `[inline]`

Definition at line 105 of file RotationMatrix.h.

6.7.4.8 **FloatType openEV::RotationMatrix::getRoll ()** `[inline]`

Definition at line 116 of file RotationMatrix.h.

6.7.4.9 **FloatType openEV::RotationMatrix::getYaw ()** `[inline]`

Definition at line 94 of file RotationMatrix.h.

6.7.4.10 **void openEV::RotationMatrix::setPitch (FloatType *pitch*)** `[inline]`

set pitch angle . Invalidates the matrix.

Definition at line 98 of file RotationMatrix.h.

6.7.4.11 **void openEV::RotationMatrix::setRoll (FloatType *roll*)** `[inline]`

set roll angle . Invalidates the matrix.

Definition at line 109 of file RotationMatrix.h.

6.7.4.12 **void openEV::RotationMatrix::setYaw (FloatType *yaw*)** `[inline]`

set yaw angle . Invalidates the matrix.

Definition at line 87 of file RotationMatrix.h.

6.7.5 Member Data Documentation

6.7.5.1 **RotationMatrixType openEV::RotationMatrix::matrixGloToPlane** `[protected]`

The rotation matrix from the global(world) coordinate system to the plane coordinate system.

Definition at line 154 of file RotationMatrix.h.

6.7.5.2 **bool openEV::RotationMatrix::matrixGloToPlanelValid** `[protected]`

Definition at line 158 of file RotationMatrix.h.

6.7.5.3 RotationMatrixType openEV::RotationMatrix::matrixPlaneToGlo [protected]

The rotation matrix from the global(world) coordinate system to the plane coordinate system.

Definition at line 156 of file RotationMatrix.h.

6.7.5.4 bool openEV::RotationMatrix::matrixPlaneToGloIsValid [protected]

Definition at line 159 of file RotationMatrix.h.

6.7.5.5 FloatType openEV::RotationMatrix::pitch [protected]

Pitch angle in deg. in the norm DIN 9300. Also called **Elevation**. Turning nose up around the y axis is positive.

Definition at line 165 of file RotationMatrix.h.

6.7.5.6 FloatType openEV::RotationMatrix::roll [protected]

Roll angle in deg. in the norm DIN 9300. Also called **Bank angle**.

Definition at line 167 of file RotationMatrix.h.

6.7.5.7 FloatType openEV::RotationMatrix::yaw [protected]

Yaw angle in deg. in the norm DIN 9300. Also called **Heading**. Turning right hand around the z axis, i.e. in navigation direction.

Definition at line 163 of file RotationMatrix.h.

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

- [src/RotationMatrix.h](#)
- [src/RotationMatrix.cpp](#)

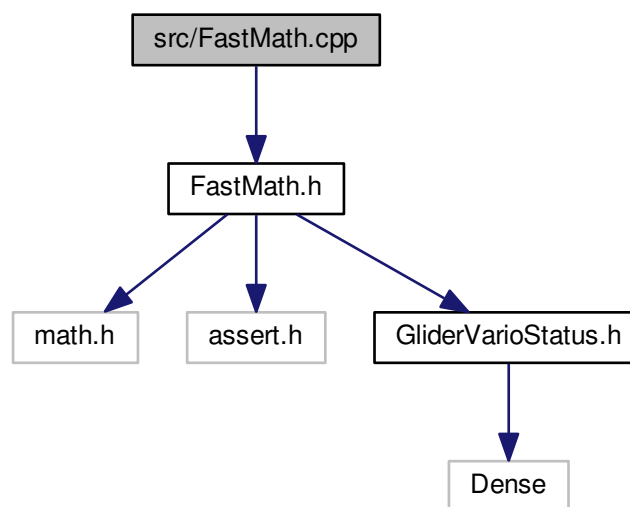
Chapter 7

File Documentation

7.1 src/FastMath.cpp File Reference

```
#include "FastMath.h"
```

Include dependency graph for FastMath.cpp:



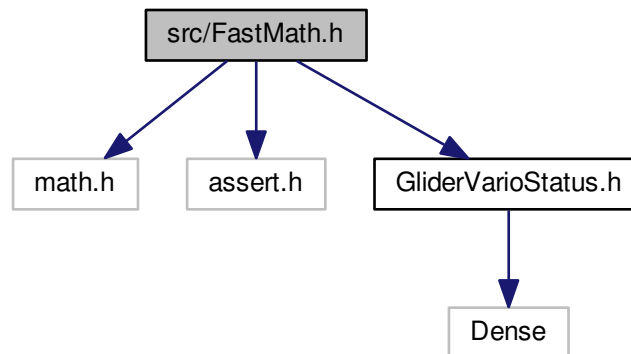
Namespaces

- [openEV](#)

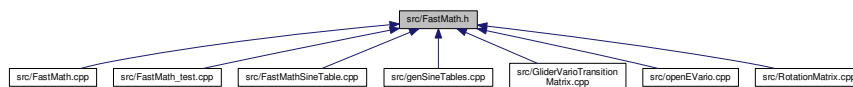
7.2 src/FastMath.h File Reference

```
#include <math.h>
#include <assert.h>
#include "GliderVarioStatus.h"
```

Include dependency graph for FastMath.h:



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



Classes

- class `openEV::FastMath`

Namespaces

- `openEV`

Macros

- `#define M_PI 3.14159265358979323846 /* pi */`

7.2.1 Macro Definition Documentation

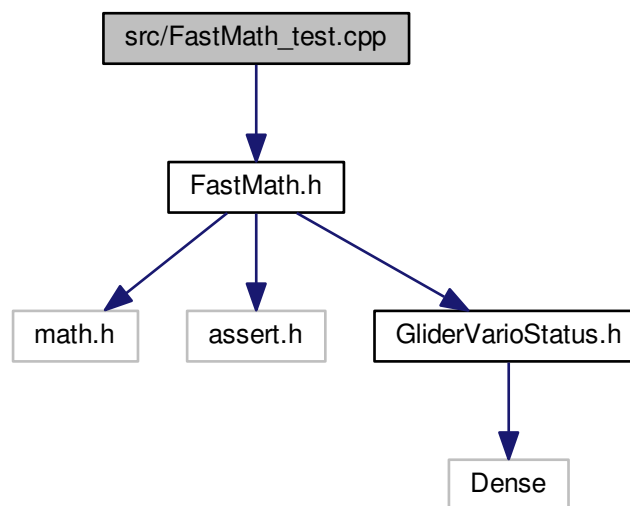
7.2.1.1 `#define M_PI 3.14159265358979323846 /* pi */`

Definition at line 38 of file `FastMath.h`.

7.3 src/FastMath_test.cpp File Reference

```
#include "FastMath.h"
```

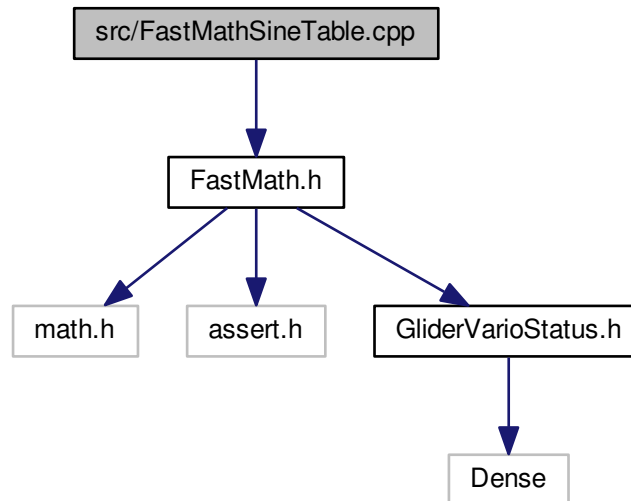
Include dependency graph for FastMath_test.cpp:



7.4 src/FastMathSineTable.cpp File Reference

```
#include "FastMath.h"
```

Include dependency graph for FastMathSineTable.cpp:



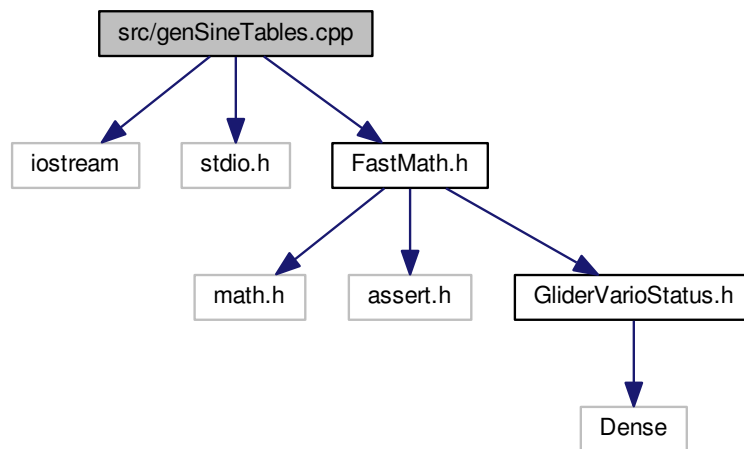
Namespaces

- [openEV](#)

7.5 src/genSineTables.cpp File Reference

```
#include <iostream>
#include <stdio.h>
#include "FastMath.h"
```

Include dependency graph for genSineTables.cpp:



Functions

- static int [printSineTable](#) (const char *fileName)
Generate constant sinus tables for FastMath [genSineTables.cpp](#).
- static void [usage](#) ()
- int [main](#) (int argc, const char **argv)

7.5.1 Function Documentation

7.5.1.1 int main (int argc, const char ** argv)

Definition at line 115 of file `genSineTables.cpp`.

7.5.1.2 static int printSineTable (const char * fileName) [static]

Generate constant sinus tables for FastMath [genSineTables.cpp](#).

Created on: Dec 27, 2015 Author: hor

This file is part of openEVario, an electronic variometer for glider planes Copyright (C) 2016 Kai Horstmann

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA. Print the sine table into a c++ source file "fileName".

Definition at line 32 of file `genSineTables.cpp`.

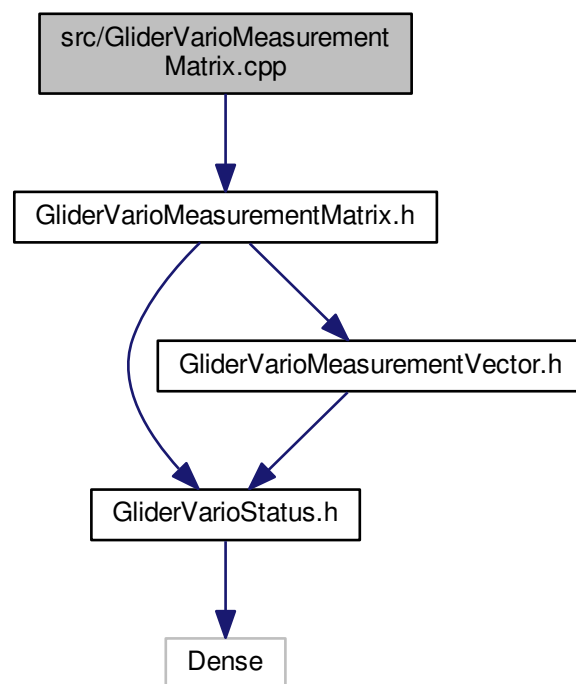
7.5.1.3 static void usage () [static]

Definition at line 111 of file genSineTables.cpp.

7.6 src/GliderVarioMeasurementMatrix.cpp File Reference

```
#include "GliderVarioMeasurementMatrix.h"
```

Include dependency graph for GliderVarioMeasurementMatrix.cpp:



Namespaces

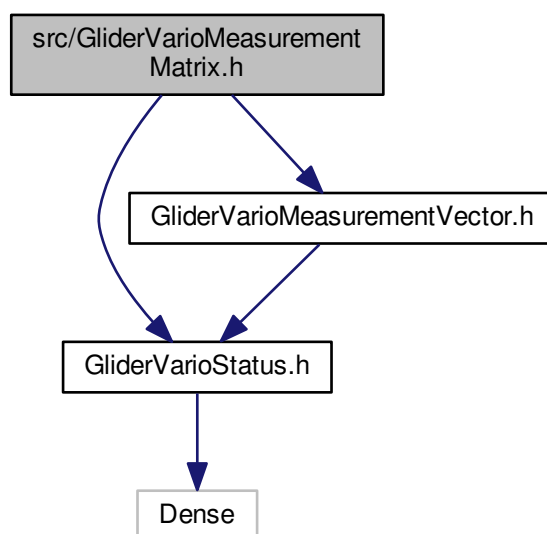
- [openEV](#)

7.7 src/GliderVarioMeasurementMatrix.h File Reference

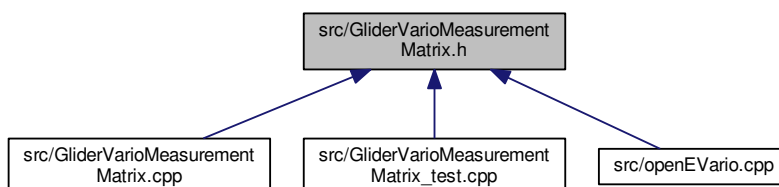
```
#include "GliderVarioStatus.h"
```

```
#include "GliderVarioMeasurementVector.h"
```


Include dependency graph for GliderVarioMeasurementMatrix.h:



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



Classes

- class [openEV::GliderVarioMeasurementMatrix](#)

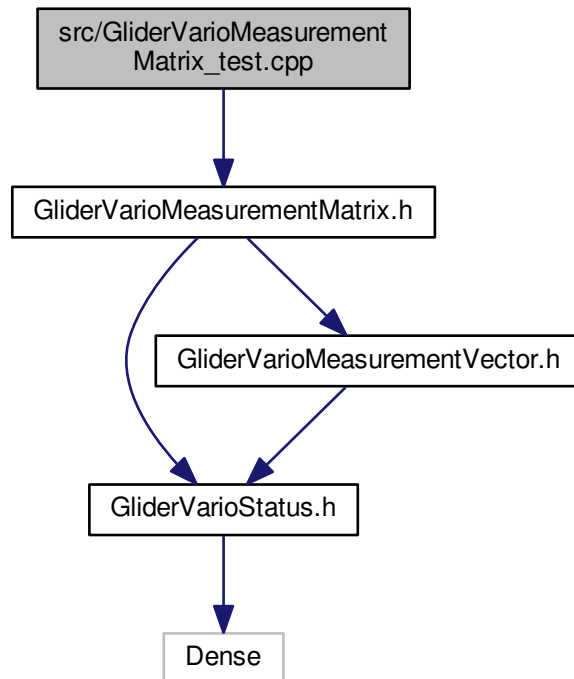
Namespaces

- [openEV](#)

7.8 src/GliderVarioMeasurementMatrix_test.cpp File Reference

```
#include "GliderVarioMeasurementMatrix.h"
```

Include dependency graph for GliderVarioMeasurementMatrix_test.cpp:



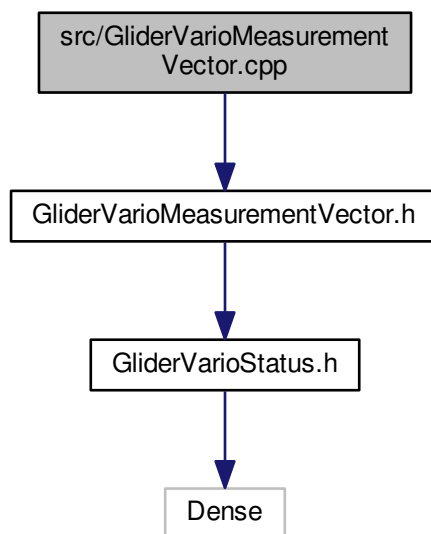
Namespaces

- [openEV](#)

7.9 src/GliderVarioMeasurementVector.cpp File Reference

```
#include "GliderVarioMeasurementVector.h"
```

Include dependency graph for GliderVarioMeasurementVector.cpp:



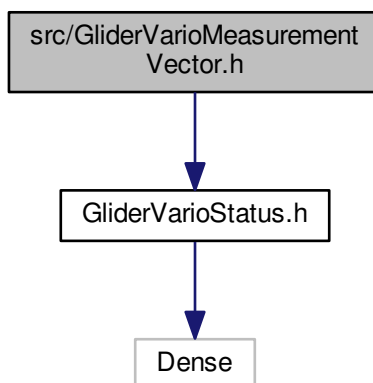
Namespaces

- [openEV](#)

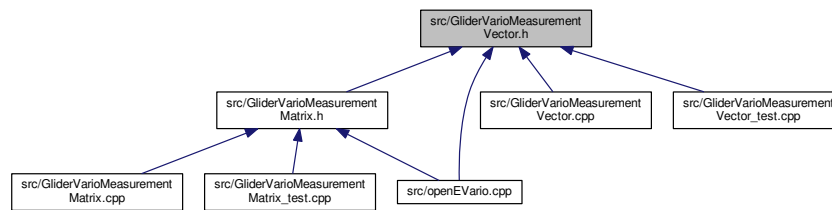
7.10 src/GliderVarioMeasurementVector.h File Reference

```
#include "GliderVarioStatus.h"
```

Include dependency graph for GliderVarioMeasurementVector.h:



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



Classes

- class `openEV::GliderVarioMeasurementVector`

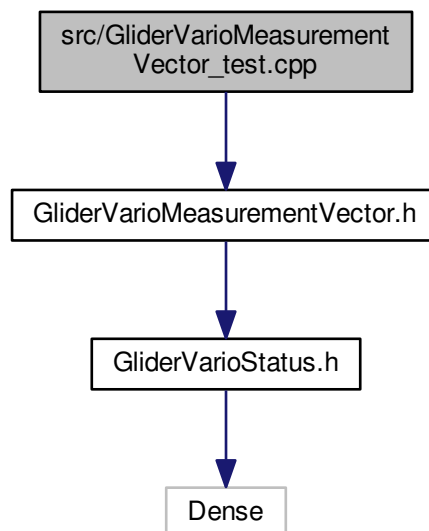
Namespaces

- openEV

7.11 src/GliderVarioMeasurementVector_test.cpp File Reference

```
#include "GliderVarioMeasurementVector.h"
```

Include dependency graph for GliderVarioMeasurementVector_test.cpp:



Namespaces

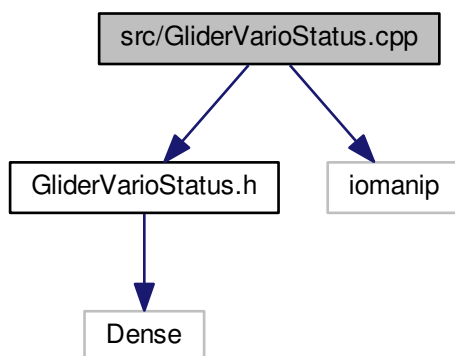
- openEV

7.12 src/GliderVarioStatus.cpp File Reference

```
#include "GliderVarioStatus.h"
```

```
#include <iomanip>
```

Include dependency graph for GliderVarioStatus.cpp:



Namespaces

- [openEV](#)

Functions

- `std::ostream & operator<< (std::ostream &o, openEV::GliderVarioStatus &s)`

7.12.1 Function Documentation

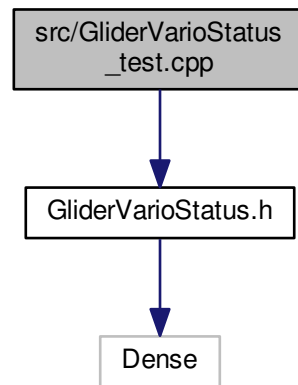
7.12.1.1 `std::ostream& operator<< (std::ostream & o, openEV::GliderVarioStatus & s)`

Definition at line 140 of file `GliderVarioStatus.cpp`.

7.14 src/GliderVarioStatus_test.cpp File Reference

```
#include "GliderVarioStatus.h"
```

Include dependency graph for GliderVarioStatus_test.cpp:



7.15 src/GliderVarioTransitionMatrix.cpp File Reference

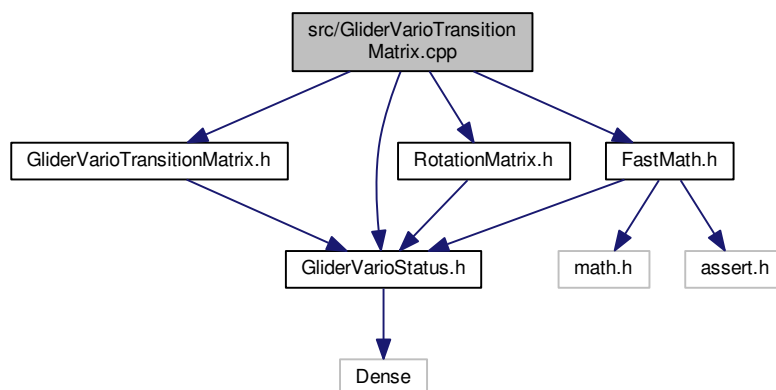
```
#include <GliderVarioTransitionMatrix.h>
```

```
#include "GliderVarioStatus.h"
```

```
#include "RotationMatrix.h"
```

```
#include "FastMath.h"
```

Include dependency graph for GliderVarioTransitionMatrix.cpp:



Namespaces

- [openEV](#)

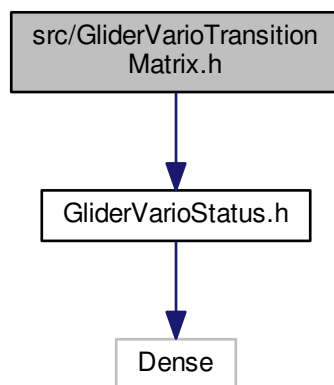
Variables

- FloatType constexpr [openEV::lenLatitude](#) = 111132.0

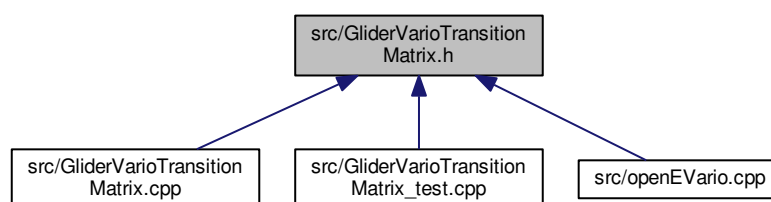
7.16 src/GliderVarioTransitionMatrix.h File Reference

```
#include "GliderVarioStatus.h"
```

Include dependency graph for GliderVarioTransitionMatrix.h:



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



Classes

- class [openEV::GliderVarioTransitionMatrix](#)

Namespaces

- [openEV](#)

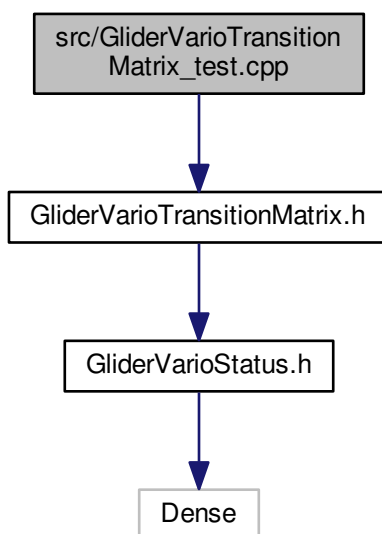
Variables

- static FloatType constexpr `openEV::GRAVITY` = 9.81

7.17 src/GliderVarioTransitionMatrix_test.cpp File Reference

```
#include "GliderVarioTransitionMatrix.h"
```

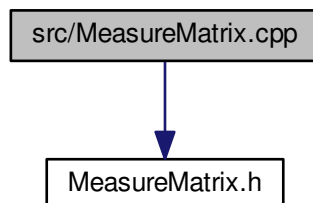
Include dependency graph for GliderVarioTransitionMatrix_test.cpp:



7.18 src/MeasureMatrix.cpp File Reference

```
#include "MeasureMatrix.h"
```

Include dependency graph for MeasureMatrix.cpp:

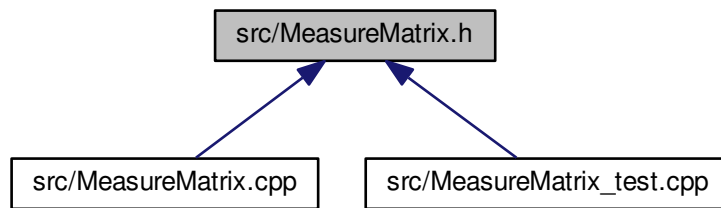


Namespaces

- [openEV](#)

7.19 src/MeasureMatrix.h File Reference

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



Classes

- class [openEV::MeasureMatrix](#)

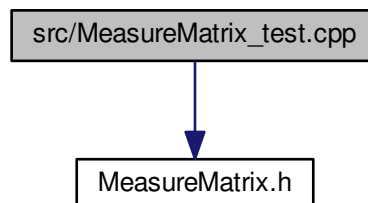
Namespaces

- [openEV](#)

7.20 src/MeasureMatrix_test.cpp File Reference

```
#include "MeasureMatrix.h"
```

Include dependency graph for `MeasureMatrix_test.cpp`:



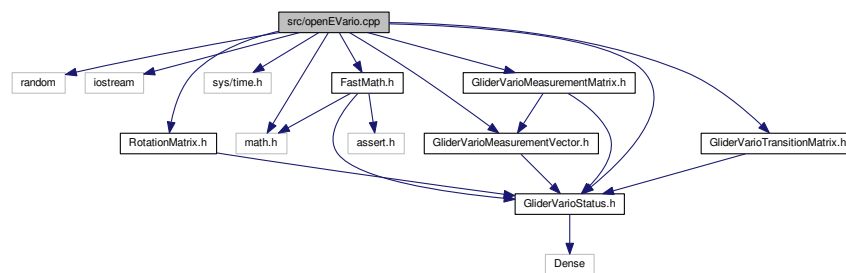
Namespaces

- [openEV](#)

7.21 src/openEVario.cpp File Reference

```
#include <random>
#include <iostream>
#include <math.h>
#include <sys/time.h>
#include "GliderVarioStatus.h"
#include "GliderVarioTransitionMatrix.h"
#include "RotationMatrix.h"
#include "FastMath.h"
#include "GliderVarioMeasurementVector.h"
#include "GliderVarioMeasurementMatrix.h"
```

Include dependency graph for openEVario.cpp:



Functions

- `int main (int argc, char *argv[])`
The one and only [main\(\)](#) function Startup and initialization. Demonization if required. Entry into the main processing loop.

Variables

- `mt19937 randomGenerator`
- `FloatType x = 0`

7.21.1 Function Documentation

7.21.1.1 `int main (int argc, char * argv[])`

The one and only [main\(\)](#) function Startup and initialization. Demonization if required. Entry into the main processing loop.

Parameters

<i>argc</i>	
<i>argv</i>	

Returns

TODO remove all the test code, and replace it by real application code.

Definition at line 56 of file openEVario.cpp.

7.21.2 Variable Documentation

7.21.2.1 mt19937 randomGenerator

Definition at line 43 of file openEVario.cpp.

7.21.2.2 FloatType x = 0

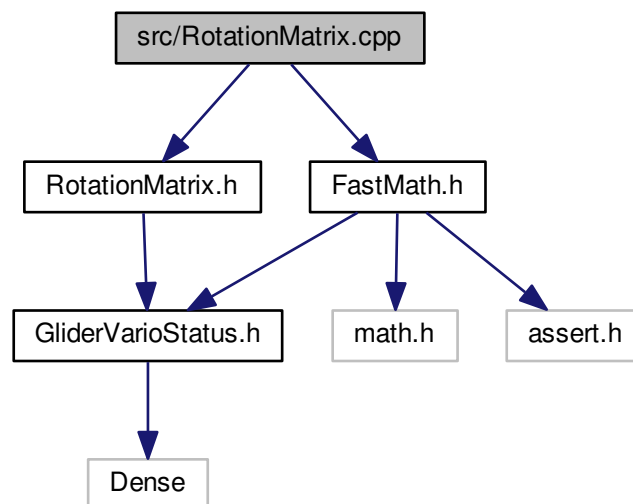
Definition at line 45 of file openEVario.cpp.

7.22 src/RotationMatrix.cpp File Reference

```
#include "RotationMatrix.h"
```

```
#include "FastMath.h"
```

Include dependency graph for RotationMatrix.cpp:



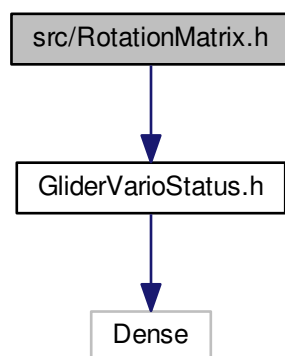
Namespaces

- [openEV](#)

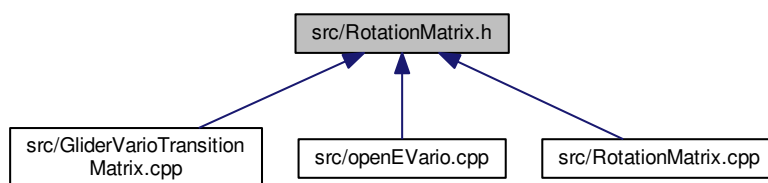
7.23 src/RotationMatrix.h File Reference

```
#include "GliderVarioStatus.h"
```

Include dependency graph for RotationMatrix.h:



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



Classes

- class [openEV::RotationMatrix](#)

Namespaces

- [openEV](#)

Index

- ~FastMath
 - openEV::FastMath, [12](#)
- ~GliderVarioMeasurementMatrix
 - openEV::GliderVarioMeasurementMatrix, [14](#)
- ~GliderVarioMeasurementVector
 - openEV::GliderVarioMeasurementVector, [17](#)
- ~GliderVarioStatus
 - openEV::GliderVarioStatus, [23](#)
- ~GliderVarioTransitionMatrix
 - openEV::GliderVarioTransitionMatrix, [28](#)
- ~MeasureMatrix
 - openEV::MeasureMatrix, [29](#)
- ~RotationMatrix
 - openEV::RotationMatrix, [30](#)
- accelX
 - openEV::GliderVarioMeasurementVector, [17](#)
 - openEV::GliderVarioStatus, [24](#)
- accelY
 - openEV::GliderVarioMeasurementVector, [17](#)
 - openEV::GliderVarioStatus, [24](#)
- accelZ
 - openEV::GliderVarioMeasurementVector, [17](#)
 - openEV::GliderVarioStatus, [24](#)
- altMSL
 - openEV::GliderVarioStatus, [24](#)
- calcMeasurementMatrix
 - openEV::GliderVarioMeasurementMatrix, [14](#)
- calcPlaneVectorToWorldVector
 - openEV::RotationMatrix, [31](#)
- calcTransitionMatrix
 - openEV::GliderVarioTransitionMatrix, [28](#)
- calcWorldVectorToPlaneVector
 - openEV::RotationMatrix, [31](#)
- calculateRotationMatrixGloToPlane
 - openEV::RotationMatrix, [31](#)
- calculateRotationMatrixPlaneToGlo
 - openEV::RotationMatrix, [31](#)
- degToRad
 - openEV::FastMath, [13](#)
- fastCos
 - openEV::FastMath, [12](#)
- FastMath
 - openEV::FastMath, [12](#)
- FastMath.h
 - M_PI, [36](#)
- fastSin
 - openEV::FastMath, [12](#)
- fastSinPositive
 - openEV::FastMath, [12](#)
- fastSinRaw
 - openEV::FastMath, [13](#)
- FloatType
 - openEV, [9](#)
- GRAVITY
 - openEV, [10](#)
- genSineTables.cpp
 - main, [39](#)
 - printSineTable, [39](#)
 - usage, [39](#)
- getMatrixGloToPlane
 - openEV::RotationMatrix, [31](#)
- getMatrixPlaneToGlo
 - openEV::RotationMatrix, [32](#)
- getMeasureMatrix
 - openEV::GliderVarioMeasurementMatrix, [14](#)
- getMeasureVector
 - openEV::GliderVarioMeasurementVector, [17](#)
- getPitch
 - openEV::RotationMatrix, [32](#)
- getRoll
 - openEV::RotationMatrix, [32](#)
- getStatusVector
 - openEV::GliderVarioStatus, [24](#)
- getTransitionMatrix
 - openEV::GliderVarioTransitionMatrix, [28](#)
- getYaw
 - openEV::RotationMatrix, [32](#)
- GliderVarioMeasurementMatrix
 - openEV::GliderVarioMeasurementMatrix, [14](#)
- GliderVarioMeasurementVector
 - openEV::GliderVarioMeasurementVector, [17](#)
- GliderVarioStatus
 - openEV::GliderVarioStatus, [23](#)
- GliderVarioStatus.cpp
 - operator<<, [45](#)
- GliderVarioStatus.h
 - operator<<, [46](#)
- GliderVarioTransitionMatrix
 - openEV::GliderVarioTransitionMatrix, [28](#)
- gpsHeading
 - openEV::GliderVarioMeasurementVector, [18](#)
- gpsLatitude
 - openEV::GliderVarioMeasurementVector, [18](#)
- gpsLongitude
 - openEV::GliderVarioMeasurementVector, [18](#)

- gpsMSL
 - openEV::GliderVarioMeasurementVector, 18
- gpsSpeed
 - openEV::GliderVarioMeasurementVector, 18
- groundSpeedEast
 - openEV::GliderVarioStatus, 24
- groundSpeedNorth
 - openEV::GliderVarioStatus, 24
- gyroBiasX
 - openEV::GliderVarioStatus, 25
- gyroBiasY
 - openEV::GliderVarioStatus, 25
- gyroBiasZ
 - openEV::GliderVarioStatus, 25
- gyroRateX
 - openEV::GliderVarioMeasurementVector, 18
- gyroRateY
 - openEV::GliderVarioMeasurementVector, 18
- gyroRateZ
 - openEV::GliderVarioMeasurementVector, 18
- heading
 - openEV::GliderVarioStatus, 25
- latitude
 - openEV::GliderVarioStatus, 25
- lenLatitude
 - openEV, 10
- longitude
 - openEV::GliderVarioStatus, 25
- M_PI
 - FastMath.h, 36
- MEASURE_IND_ACC_X
 - openEV::GliderVarioMeasurementVector, 17
- MEASURE_IND_ACC_Y
 - openEV::GliderVarioMeasurementVector, 17
- MEASURE_IND_ACC_Z
 - openEV::GliderVarioMeasurementVector, 17
- MEASURE_IND_GPS_ALTMSL
 - openEV::GliderVarioMeasurementVector, 16
- MEASURE_IND_GPS_HEADING
 - openEV::GliderVarioMeasurementVector, 16
- MEASURE_IND_GPS_LAT
 - openEV::GliderVarioMeasurementVector, 16
- MEASURE_IND_GPS_LON
 - openEV::GliderVarioMeasurementVector, 16
- MEASURE_IND_GPS_SPEED
 - openEV::GliderVarioMeasurementVector, 17
- MEASURE_IND_GYRO_RATE_X
 - openEV::GliderVarioMeasurementVector, 17
- MEASURE_IND_GYRO_RATE_Y
 - openEV::GliderVarioMeasurementVector, 17
- MEASURE_IND_GYRO_RATE_Z
 - openEV::GliderVarioMeasurementVector, 17
- MEASURE_IND_MAG_X
 - openEV::GliderVarioMeasurementVector, 17
- MEASURE_IND_MAG_Y
 - openEV::GliderVarioMeasurementVector, 17
- MEASURE_IND_MAG_Z
 - openEV::GliderVarioMeasurementVector, 17
- MEASURE_IND_PRESS_ALT
 - openEV::GliderVarioMeasurementVector, 17
- MEASURE_IND_TAS
 - openEV::GliderVarioMeasurementVector, 17
- MEASURE_NUM_ROWS
 - openEV::GliderVarioMeasurementVector, 17
- magX
 - openEV::GliderVarioMeasurementVector, 19
- magY
 - openEV::GliderVarioMeasurementVector, 19
- magZ
 - openEV::GliderVarioMeasurementVector, 19
- main
 - genSineTables.cpp, 39
 - openEVario.cpp, 51
- matrixGloToPlane
 - openEV::RotationMatrix, 32
- matrixGloToPlanelsValid
 - openEV::RotationMatrix, 32
- matrixPlaneToGlo
 - openEV::RotationMatrix, 32
- matrixPlaneToGloIsValid
 - openEV::RotationMatrix, 33
- MeasureComponentIndex
 - openEV::GliderVarioMeasurementVector, 16
- MeasureMatrix
 - openEV::MeasureMatrix, 29
- MeasureMatrixType
 - openEV::GliderVarioMeasurementMatrix, 14
- measureVector
 - openEV::GliderVarioMeasurementVector, 19
- MeasureVectorType
 - openEV::GliderVarioMeasurementVector, 16
- measurementMatrix
 - openEV::GliderVarioMeasurementMatrix, 15
- normalizeAngles
 - openEV::GliderVarioStatus, 24
- openEV, 9
 - FloatType, 9
 - GRAVITY, 10
 - lenLatitude, 10
 - Vector3DType, 9
- openEV::FastMath, 11
 - ~FastMath, 12
 - degToRad, 13
 - fastCos, 12
 - FastMath, 12
 - fastSin, 12
 - fastSinPositive, 12
 - fastSinRaw, 13
 - radToDeg, 13
 - sineSamplesPerDegree, 13
 - sinusTable, 13
 - sizeSineTable, 13
- openEV::GliderVarioMeasurementMatrix, 14

- ~GliderVarioMeasurementMatrix, 14
 - calcMeasurementMatrix, 14
 - getMeasureMatrix, 14
 - GliderVarioMeasurementMatrix, 14
 - MeasureMatrixType, 14
 - measurementMatrix, 15
- openEV::GliderVarioMeasurementVector, 15
 - ~GliderVarioMeasurementVector, 17
 - accelX, 17
 - accelY, 17
 - accelZ, 17
 - getMeasureVector, 17
 - GliderVarioMeasurementVector, 17
 - gpsHeading, 18
 - gpsLatitude, 18
 - gpsLongitude, 18
 - gpsMSL, 18
 - gpsSpeed, 18
 - gyroRateX, 18
 - gyroRateY, 18
 - gyroRateZ, 18
 - MEASURE_IND_ACC_X, 17
 - MEASURE_IND_ACC_Y, 17
 - MEASURE_IND_ACC_Z, 17
 - MEASURE_IND_GPS_ALTMSL, 16
 - MEASURE_IND_GPS_HEADING, 16
 - MEASURE_IND_GPS_LAT, 16
 - MEASURE_IND_GPS_LON, 16
 - MEASURE_IND_GPS_SPEED, 17
 - MEASURE_IND_GYRO_RATE_X, 17
 - MEASURE_IND_GYRO_RATE_Y, 17
 - MEASURE_IND_GYRO_RATE_Z, 17
 - MEASURE_IND_MAG_X, 17
 - MEASURE_IND_MAG_Y, 17
 - MEASURE_IND_MAG_Z, 17
 - MEASURE_IND_PRESS_ALT, 17
 - MEASURE_IND_TAS, 17
 - MEASURE_NUM_ROWS, 17
 - magX, 19
 - magY, 19
 - magZ, 19
 - MeasureComponentIndex, 16
 - measureVector, 19
 - MeasureVectorType, 16
 - pressAlt, 19
 - trueAirSpeed, 19
- openEV::GliderVarioStatus, 19
 - ~GliderVarioStatus, 23
 - accelX, 24
 - accelY, 24
 - accelZ, 24
 - altMSL, 24
 - getStatusVector, 24
 - GliderVarioStatus, 23
 - groundSpeedEast, 24
 - groundSpeedNorth, 24
 - gyroBiasX, 25
 - gyroBiasY, 25
 - gyroBiasZ, 25
 - heading, 25
 - latitude, 25
 - longitude, 25
 - normalizeAngles, 24
 - pitchAngle, 25
 - pitchRateY, 25
 - rateOfSink, 25
 - rollAngle, 26
 - rollRateX, 26
 - STATUS_IND_ACC_X, 23
 - STATUS_IND_ACC_Y, 23
 - STATUS_IND_ACC_Z, 23
 - STATUS_IND_ALT_MSL, 23
 - STATUS_IND_GYRO_BIAS_X, 23
 - STATUS_IND_GYRO_BIAS_Y, 23
 - STATUS_IND_GYRO_BIAS_Z, 23
 - STATUS_IND_HEADING, 23
 - STATUS_IND_LATITUDE, 23
 - STATUS_IND_LONGITUDE, 23
 - STATUS_IND_PITCH, 23
 - STATUS_IND_RATE_OF_SINK, 23
 - STATUS_IND_ROLL, 23
 - STATUS_IND_ROTATION_X, 23
 - STATUS_IND_ROTATION_Y, 23
 - STATUS_IND_ROTATION_Z, 23
 - STATUS_IND_SPEED_GROUND_E, 23
 - STATUS_IND_SPEED_GROUND_N, 23
 - STATUS_IND_TAS, 23
 - STATUS_IND_THERMAL_SPEED, 23
 - STATUS_IND_VERTICAL_SPEED, 23
 - STATUS_IND_WIND_SPEED_E, 23
 - STATUS_IND_WIND_SPEED_N, 23
 - STATUS_IND_YAW, 23
 - STATUS_NUM_ROWS, 23
 - StatusComponentIndex, 23
 - statusVector, 26
 - StatusVectorType, 22
 - thermalSpeed, 26
 - trueAirSpeed, 26
 - verticalSpeed, 26
 - windSpeedEast, 26
 - windSpeedNorth, 26
 - yawAngle, 26
 - yawRateZ, 27
- openEV::GliderVarioTransitionMatrix, 27
 - ~GliderVarioTransitionMatrix, 28
 - calcTransitionMatrix, 28
 - getTransitionMatrix, 28
 - GliderVarioTransitionMatrix, 28
 - transitionMatrix, 28
 - TransitionMatrixType, 27
 - updateStatus, 28
- openEV::MeasureMatrix, 29
 - ~MeasureMatrix, 29
 - MeasureMatrix, 29
- openEV::RotationMatrix, 29
 - ~RotationMatrix, 30

- calcPlaneVectorToWorldVector, [31](#)
- calcWorldVectorToPlaneVector, [31](#)
- calculateRotationMatrixGloToPlane, [31](#)
- calculateRotationMatrixPlaneToGlo, [31](#)
- getMatrixGloToPlane, [31](#)
- getMatrixPlaneToGlo, [32](#)
- getPitch, [32](#)
- getRoll, [32](#)
- getYaw, [32](#)
- matrixGloToPlane, [32](#)
- matrixGloToPlaneIsValid, [32](#)
- matrixPlaneToGlo, [32](#)
- matrixPlaneToGloIsValid, [33](#)
- pitch, [33](#)
- roll, [33](#)
- RotationMatrix, [30](#)
- RotationMatrixType, [30](#)
- setPitch, [32](#)
- setRoll, [32](#)
- setYaw, [32](#)
- yaw, [33](#)
- openEVario.cpp
 - main, [51](#)
 - randomGenerator, [52](#)
 - x, [52](#)
- operator<<
 - GliderVarioStatus.cpp, [45](#)
 - GliderVarioStatus.h, [46](#)
- pitch
 - openEV::RotationMatrix, [33](#)
- pitchAngle
 - openEV::GliderVarioStatus, [25](#)
- pitchRateY
 - openEV::GliderVarioStatus, [25](#)
- pressAlt
 - openEV::GliderVarioMeasurementVector, [19](#)
- printSineTable
 - genSineTables.cpp, [39](#)
- radToDeg
 - openEV::FastMath, [13](#)
- randomGenerator
 - openEVario.cpp, [52](#)
- rateOfSink
 - openEV::GliderVarioStatus, [25](#)
- roll
 - openEV::RotationMatrix, [33](#)
- rollAngle
 - openEV::GliderVarioStatus, [26](#)
- rollRateX
 - openEV::GliderVarioStatus, [26](#)
- RotationMatrix
 - openEV::RotationMatrix, [30](#)
- RotationMatrixType
 - openEV::RotationMatrix, [30](#)
- STATUS_IND_ACC_X
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_ACC_Y
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_ACC_Z
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_ALT_MSL
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_GYRO_BIAS_X
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_GYRO_BIAS_Y
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_GYRO_BIAS_Z
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_HEADING
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_LATITUDE
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_LONGITUDE
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_PITCH
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_RATE_OF_SINK
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_ROLL
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_ROTATION_X
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_ROTATION_Y
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_ROTATION_Z
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_SPEED_GROUND_E
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_SPEED_GROUND_N
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_TAS
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_THERMAL_SPEED
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_VERTICAL_SPEED
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_WIND_SPEED_E
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_WIND_SPEED_N
 - openEV::GliderVarioStatus, [23](#)
- STATUS_IND_YAW
 - openEV::GliderVarioStatus, [23](#)
- STATUS_NUM_ROWS
 - openEV::GliderVarioStatus, [23](#)
- setPitch
 - openEV::RotationMatrix, [32](#)
- setRoll
 - openEV::RotationMatrix, [32](#)
- setYaw
 - openEV::RotationMatrix, [32](#)
- sineSamplesPerDegree
 - openEV::FastMath, [13](#)
- sinusTable
 - openEV::FastMath, [13](#)

- sizeSineTable
 - openEV::FastMath, [13](#)
- src/FastMath.cpp, [35](#)
- src/FastMath.h, [35](#)
- src/FastMath_test.cpp, [37](#)
- src/FastMathSineTable.cpp, [38](#)
- src/GliderVarioMeasurementMatrix.cpp, [40](#)
- src/GliderVarioMeasurementMatrix.h, [40](#)
- src/GliderVarioMeasurementMatrix_test.cpp, [41](#)
- src/GliderVarioMeasurementVector.cpp, [42](#)
- src/GliderVarioMeasurementVector.h, [43](#)
- src/GliderVarioMeasurementVector_test.cpp, [44](#)
- src/GliderVarioStatus.cpp, [45](#)
- src/GliderVarioStatus.h, [46](#)
- src/GliderVarioStatus_test.cpp, [47](#)
- src/GliderVarioTransitionMatrix.cpp, [47](#)
- src/GliderVarioTransitionMatrix.h, [48](#)
- src/GliderVarioTransitionMatrix_test.cpp, [49](#)
- src/MeasureMatrix.cpp, [49](#)
- src/MeasureMatrix.h, [50](#)
- src/MeasureMatrix_test.cpp, [50](#)
- src/RotationMatrix.cpp, [52](#)
- src/RotationMatrix.h, [53](#)
- src/genSineTables.cpp, [38](#)
- src/openEVario.cpp, [51](#)
- StatusComponentIndex
 - openEV::GliderVarioStatus, [23](#)
- statusVector
 - openEV::GliderVarioStatus, [26](#)
- StatusVectorType
 - openEV::GliderVarioStatus, [22](#)
- thermalSpeed
 - openEV::GliderVarioStatus, [26](#)
- transitionMatrix
 - openEV::GliderVarioTransitionMatrix, [28](#)
- TransitionMatrixType
 - openEV::GliderVarioTransitionMatrix, [27](#)
- trueAirSpeed
 - openEV::GliderVarioMeasurementVector, [19](#)
 - openEV::GliderVarioStatus, [26](#)
- updateStatus
 - openEV::GliderVarioTransitionMatrix, [28](#)
- usage
 - genSineTables.cpp, [39](#)
- Vector3DType
 - openEV, [9](#)
- verticalSpeed
 - openEV::GliderVarioStatus, [26](#)
- windSpeedEast
 - openEV::GliderVarioStatus, [26](#)
- windSpeedNorth
 - openEV::GliderVarioStatus, [26](#)
- x
 - openEVario.cpp, [52](#)
- yaw
 - openEV::RotationMatrix, [33](#)
- yawAngle
 - openEV::GliderVarioStatus, [26](#)
- yawRateZ
 - openEV::GliderVarioStatus, [27](#)