

openEVario

0.1

Generated by Doxygen 1.8.9.1

Sun Mar 6 2016 01:50:48



# Contents

<b>1</b>	<b>Todo List</b>	<b>1</b>
<b>2</b>	<b>Namespace Index</b>	<b>3</b>
2.1	Namespace List . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Namespace Documentation</b>	<b>9</b>
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
<b>6</b>	<b>Class Documentation</b>	<b>11</b>
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	fastATan2 . . . . .	12
6.1.3.2	fastATan2Pos . . . . .	12
6.1.3.3	fastATanRaw . . . . .	13
6.1.3.4	fastCos . . . . .	13
6.1.3.5	fastSin . . . . .	13
6.1.3.6	fastSinPositive . . . . .	13

6.1.3.7	<a href="#">fastSinRaw</a>	14
6.1.4	<a href="#">Member Data Documentation</a>	14
6.1.4.1	<a href="#">atanTable</a>	14
6.1.4.2	<a href="#">degToRad</a>	14
6.1.4.3	<a href="#">radToDeg</a>	14
6.1.4.4	<a href="#">sineSamplesPerDegree</a>	14
6.1.4.5	<a href="#">sinusTable</a>	14
6.1.4.6	<a href="#">sizeATanTable</a>	15
6.1.4.7	<a href="#">sizeSineTable</a>	15
6.2	<a href="#">openEV::GliderVarioMeasurementMatrix Class Reference</a>	15
6.2.1	<a href="#">Detailed Description</a>	15
6.2.2	<a href="#">Member Typedef Documentation</a>	15
6.2.2.1	<a href="#">MeasureMatrixType</a>	15
6.2.3	<a href="#">Constructor &amp; Destructor Documentation</a>	16
6.2.3.1	<a href="#">GliderVarioMeasurementMatrix</a>	16
6.2.3.2	<a href="#">~GliderVarioMeasurementMatrix</a>	16
6.2.4	<a href="#">Member Function Documentation</a>	16
6.2.4.1	<a href="#">calcMeasurementMatrix</a>	16
6.2.4.2	<a href="#">getMeasureMatrix</a>	16
6.2.5	<a href="#">Member Data Documentation</a>	16
6.2.5.1	<a href="#">measurementMatrix</a>	16
6.3	<a href="#">openEV::GliderVarioMeasurementVector Class Reference</a>	16
6.3.1	<a href="#">Detailed Description</a>	17
6.3.2	<a href="#">Member Typedef Documentation</a>	18
6.3.2.1	<a href="#">MeasureVectorType</a>	18
6.3.3	<a href="#">Member Enumeration Documentation</a>	18
6.3.3.1	<a href="#">MeasureComponentIndex</a>	18
6.3.4	<a href="#">Constructor &amp; Destructor Documentation</a>	18
6.3.4.1	<a href="#">GliderVarioMeasurementVector</a>	18
6.3.4.2	<a href="#">~GliderVarioMeasurementVector</a>	18
6.3.5	<a href="#">Member Function Documentation</a>	18
6.3.5.1	<a href="#">getMeasureVector</a>	18
6.3.6	<a href="#">Member Data Documentation</a>	19
6.3.6.1	<a href="#">accelX</a>	19
6.3.6.2	<a href="#">accelY</a>	19
6.3.6.3	<a href="#">accelZ</a>	19
6.3.6.4	<a href="#">gpsHeading</a>	19
6.3.6.5	<a href="#">gpsLatitude</a>	19
6.3.6.6	<a href="#">gpsLongitude</a>	19
6.3.6.7	<a href="#">gpsMSL</a>	19

6.3.6.8	<a href="#">gpsSpeed</a>	19
6.3.6.9	<a href="#">gyroRateX</a>	20
6.3.6.10	<a href="#">gyroRateY</a>	20
6.3.6.11	<a href="#">gyroRateZ</a>	20
6.3.6.12	<a href="#">magX</a>	20
6.3.6.13	<a href="#">magY</a>	20
6.3.6.14	<a href="#">magZ</a>	20
6.3.6.15	<a href="#">measureVector</a>	20
6.3.6.16	<a href="#">pressAlt</a>	20
6.3.6.17	<a href="#">trueAirSpeed</a>	20
6.4	<a href="#">openEV::GliderVarioStatus Class Reference</a>	21
6.4.1	<a href="#">Detailed Description</a>	22
6.4.2	<a href="#">Member Typedef Documentation</a>	24
6.4.2.1	<a href="#">StatusVectorType</a>	24
6.4.3	<a href="#">Member Enumeration Documentation</a>	24
6.4.3.1	<a href="#">StatusComponentIndex</a>	24
6.4.4	<a href="#">Constructor &amp; Destructor Documentation</a>	25
6.4.4.1	<a href="#">GliderVarioStatus</a>	25
6.4.4.2	<a href="#">~GliderVarioStatus</a>	25
6.4.5	<a href="#">Member Function Documentation</a>	25
6.4.5.1	<a href="#">getStatusVector</a>	25
6.4.5.2	<a href="#">getStatusVector</a>	25
6.4.5.3	<a href="#">normalizeAngles</a>	25
6.4.6	<a href="#">Member Data Documentation</a>	25
6.4.6.1	<a href="#">accelX</a>	25
6.4.6.2	<a href="#">accelY</a>	25
6.4.6.3	<a href="#">accelZ</a>	25
6.4.6.4	<a href="#">altMSL</a>	25
6.4.6.5	<a href="#">groundSpeedEast</a>	26
6.4.6.6	<a href="#">groundSpeedNorth</a>	26
6.4.6.7	<a href="#">gyroBiasX</a>	26
6.4.6.8	<a href="#">gyroBiasY</a>	26
6.4.6.9	<a href="#">gyroBiasZ</a>	26
6.4.6.10	<a href="#">heading</a>	26
6.4.6.11	<a href="#">latitude</a>	26
6.4.6.12	<a href="#">longitude</a>	26
6.4.6.13	<a href="#">pitchAngle</a>	26
6.4.6.14	<a href="#">pitchRateY</a>	27
6.4.6.15	<a href="#">rateOfSink</a>	27
6.4.6.16	<a href="#">rollAngle</a>	27

6.4.6.17	rollRateX	27
6.4.6.18	statusVector	27
6.4.6.19	thermalSpeed	27
6.4.6.20	trueAirSpeed	27
6.4.6.21	verticalSpeed	27
6.4.6.22	windSpeedEast	27
6.4.6.23	windSpeedNorth	28
6.4.6.24	yawAngle	28
6.4.6.25	yawRateZ	28
6.5	openEV::GliderVarioTransitionMatrix Class Reference	28
6.5.1	Detailed Description	28
6.5.2	Member Typedef Documentation	29
6.5.2.1	TransitionMatrixType	29
6.5.3	Constructor & Destructor Documentation	29
6.5.3.1	GliderVarioTransitionMatrix	29
6.5.3.2	~GliderVarioTransitionMatrix	29
6.5.4	Member Function Documentation	29
6.5.4.1	calcTransitionMatrix	29
6.5.4.2	getTransitionMatrix	29
6.5.4.3	updateStatus	29
6.5.5	Member Data Documentation	29
6.5.5.1	transitionMatrix	30
6.6	openEV::MeasureMatrix Class Reference	30
6.6.1	Detailed Description	30
6.6.2	Constructor & Destructor Documentation	30
6.6.2.1	MeasureMatrix	30
6.6.2.2	~MeasureMatrix	30
6.7	openEV::RotationMatrix Class Reference	30
6.7.1	Detailed Description	31
6.7.2	Member Typedef Documentation	31
6.7.2.1	RotationMatrixType	31
6.7.3	Constructor & Destructor Documentation	32
6.7.3.1	RotationMatrix	32
6.7.3.2	RotationMatrix	32
6.7.3.3	~RotationMatrix	32
6.7.4	Member Function Documentation	32
6.7.4.1	calcPlaneVectorToWorldVector	32
6.7.4.2	calculateRotationMatrixGloToPlane	32
6.7.4.3	calculateRotationMatrixPlaneToGlo	32
6.7.4.4	calcWorldVectorToPlaneVector	33

6.7.4.5	<a href="#">getMatrixGloToPlane</a>	34
6.7.4.6	<a href="#">getMatrixPlaneToGlo</a>	34
6.7.4.7	<a href="#">getPitch</a>	34
6.7.4.8	<a href="#">getRoll</a>	34
6.7.4.9	<a href="#">getYaw</a>	34
6.7.4.10	<a href="#">setPitch</a>	34
6.7.4.11	<a href="#">setRoll</a>	34
6.7.4.12	<a href="#">setYaw</a>	34
6.7.5	<a href="#">Member Data Documentation</a>	34
6.7.5.1	<a href="#">matrixGloToPlane</a>	34
6.7.5.2	<a href="#">matrixGloToPlanelValid</a>	35
6.7.5.3	<a href="#">matrixPlaneToGlo</a>	35
6.7.5.4	<a href="#">matrixPlaneToGloIsValid</a>	35
6.7.5.5	<a href="#">pitch</a>	35
6.7.5.6	<a href="#">roll</a>	35
6.7.5.7	<a href="#">yaw</a>	35
<b>7</b>	<b><a href="#">File Documentation</a></b>	<b>37</b>
7.1	<a href="#">src/FastMath.cpp File Reference</a>	37
7.2	<a href="#">src/FastMath.h File Reference</a>	37
7.2.1	<a href="#">Macro Definition Documentation</a>	38
7.2.1.1	<a href="#">M_PI</a>	38
7.3	<a href="#">src/FastMath_test.cpp File Reference</a>	39
7.4	<a href="#">src/FastMathSineTable.cpp File Reference</a>	40
7.5	<a href="#">src/genSineTables.cpp File Reference</a>	40
7.5.1	<a href="#">Function Documentation</a>	41
7.5.1.1	<a href="#">main</a>	41
7.5.1.2	<a href="#">printSineTable</a>	41
7.5.1.3	<a href="#">usage</a>	42
7.6	<a href="#">src/GliderVarioMeasurementMatrix.cpp File Reference</a>	42
7.7	<a href="#">src/GliderVarioMeasurementMatrix.h File Reference</a>	42
7.8	<a href="#">src/GliderVarioMeasurementMatrix_test.cpp File Reference</a>	43
7.9	<a href="#">src/GliderVarioMeasurementVector.cpp File Reference</a>	44
7.10	<a href="#">src/GliderVarioMeasurementVector.h File Reference</a>	45
7.11	<a href="#">src/GliderVarioMeasurementVector_test.cpp File Reference</a>	46
7.12	<a href="#">src/GliderVarioStatus.cpp File Reference</a>	47
7.12.1	<a href="#">Function Documentation</a>	47
7.12.1.1	<a href="#">operator&lt;&lt;</a>	47
7.13	<a href="#">src/GliderVarioStatus.h File Reference</a>	48
7.13.1	<a href="#">Function Documentation</a>	48

7.13.1.1 operator<<	48
7.14 src/GliderVarioStatus_test.cpp File Reference	49
7.15 src/GliderVarioTransitionMatrix.cpp File Reference	49
7.16 src/GliderVarioTransitionMatrix.h File Reference	50
7.17 src/GliderVarioTransitionMatrix_test.cpp File Reference	51
7.18 src/MeasureMatrix.cpp File Reference	51
7.19 src/MeasureMatrix.h File Reference	52
7.20 src/MeasureMatrix_test.cpp File Reference	52
7.21 src/openEVario.cpp File Reference	53
7.21.1 Function Documentation	53
7.21.1.1 main	53
7.21.2 Variable Documentation	54
7.21.2.1 randomGenerator	54
7.21.2.2 x	54
7.22 src/RotationMatrix.cpp File Reference	54
7.23 src/RotationMatrix.h File Reference	55
<b>Index</b>	<b>57</b>



## 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:

<a href="#">openEV::FastMath</a>	??
<a href="#">openEV::GliderVarioMeasurementMatrix</a>	??
<a href="#">openEV::GliderVarioMeasurementVector</a>	??
<a href="#">openEV::GliderVarioStatus</a>	
<a href="#">GliderVarioStatus</a> manages the Kalman filter state x	??
<a href="#">openEV::GliderVarioTransitionMatrix</a>	??
<a href="#">openEV::MeasureMatrix</a>	??
<a href="#">openEV::RotationMatrix</a>	??



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

src/ <a href="#">FastMath.cpp</a>	??
src/ <a href="#">FastMath.h</a>	??
src/ <a href="#">FastMath_test.cpp</a>	??
src/ <a href="#">FastMathSineTable.cpp</a>	??
src/ <a href="#">genSineTables.cpp</a>	??
src/ <a href="#">GliderVarioMeasurementMatrix.cpp</a>	??
src/ <a href="#">GliderVarioMeasurementMatrix.h</a>	??
src/ <a href="#">GliderVarioMeasurementMatrix_test.cpp</a>	??
src/ <a href="#">GliderVarioMeasurementVector.cpp</a>	??
src/ <a href="#">GliderVarioMeasurementVector.h</a>	??
src/ <a href="#">GliderVarioMeasurementVector_test.cpp</a>	??
src/ <a href="#">GliderVarioStatus.cpp</a>	??
src/ <a href="#">GliderVarioStatus.h</a>	??
src/ <a href="#">GliderVarioStatus_test.cpp</a>	??
src/ <a href="#">GliderVarioTransitionMatrix.cpp</a>	??
src/ <a href="#">GliderVarioTransitionMatrix.h</a>	??
src/ <a href="#">GliderVarioTransitionMatrix_test.cpp</a>	??
src/ <a href="#">MeasureMatrix.cpp</a>	??
src/ <a href="#">MeasureMatrix.h</a>	??
src/ <a href="#">MeasureMatrix_test.cpp</a>	??
src/ <a href="#">openEVario.cpp</a>	??
src/ <a href="#">RotationMatrix.cpp</a>	??
src/ <a href="#">RotationMatrix.h</a>	??





## 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](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](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 [FloatType](#) [fastATan2](#) ([FloatType](#) y, [FloatType](#) x)

#### 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 unsigned [sizeATanTable](#) = 256  
*the arc tan table is defined for the 1st 45 degrees in 256 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](#) [fastATan2Pos](#) ([FloatType](#) y, [FloatType](#) x)
- static [FloatType](#) [fastATanRaw](#) ([FloatType](#) tanVal)
- static [FloatType](#) [fastSinPositive](#) ([FloatType](#) angle)

## Static Protected Attributes

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

### 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::fastATan2 ( FloatType y, FloatType x )` `[inline]`, `[static]`

Calculates the arc tan angle for the x and y component in Cartesian coordinates. Based on the signs of x and y the function returns angles from the entire circle The returned angle is in degrees from 0 to 360 degrees.

##### Parameters

<code>in</code>	<code>x</code>	component
<code>in</code>	<code>y</code>	component

##### Returns

Angle in degrees 0-360 deg.

Definition at line 102 of file FastMath.h.

#### 6.1.3.2 `static FloatType openEV::FastMath::fastATan2Pos ( FloatType y, FloatType x )` `[inline]`, `[static]`, `[protected]`

Calculates the arc tangent from the x and y component of Cartesian coordinates within the first quadrant, i.e. x and y must  $\geq 0$

## Parameters

in	x	x-component of a point in Cartesian coordinates
in	y	x-component of a point in Cartesian coordinates

## Returns

the arc tangent of the ratio of x and y

Definition at line 157 of file FastMath.h.

**6.1.3.3** `static FloatType openEV::FastMath::fastATanRaw ( FloatType tanVal )` `[inline]`, `[static]`, `[protected]`

Calculate the arc tangent value of a value between 0 and  $< 1$ . This function interpolates the pre-calculated values from the table atanTable. Due to the range of the input values only the first octant can be calculated. Everything must be mirrored from this partial range.

## Parameters

in	<i>tanVal</i>	tan value, i.e. the ratio of x and y. tan value <i>must</i> be $\geq 0$ and $< 1$ . This function is only defined in the 1st 45 degrees.
----	---------------	--

## Returns

the arc tan value in degrees.

Definition at line 187 of file FastMath.h.

**6.1.3.4** `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 89 of file FastMath.h.

**6.1.3.5** `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 74 of file FastMath.h.

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

## Parameters

<i>angle[in]</i>	in degrees. The angle MUST be $\geq 0$ .
------------------	--

## Returns

The sine value of the angle

Definition at line 204 of file FastMath.h.

**6.1.3.7 static FloatType openEV::FastMath::fastSinRaw ( FloatType *angle* )** [inline], [static], [protected]

## Parameters

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

## Returns

The sine value of the angle

Definition at line 138 of file FastMath.h.

## 6.1.4 Member Data Documentation

**6.1.4.1 const double openEV::FastMath::atanTable** [static], [protected]

The table of pre-computed arc sine values from 0 to 45 deg. Anything else is derived from this range. Here 2 larger than the number of increments: including 0, all 256 steps in between, and 1

Definition at line 131 of file FastMath.h.

**6.1.4.2 constexpr double openEV::FastMath::degToRad = M\_PI / 180.0** [static]

Definition at line 62 of file FastMath.h.

**6.1.4.3 constexpr double openEV::FastMath::radToDeg = 180.0 / M\_PI** [static]

Definition at line 61 of file FastMath.h.

**6.1.4.4 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.

**6.1.4.5 const double openEV::FastMath::sinusTable** [static], [protected]

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

Generated by [genSineTables.cpp](#).

Definition at line 127 of file FastMath.h.

6.1.4.6 constexpr unsigned openEV::FastMath::sizeATanTable = 256 [static]

the arc tan table is defined for the 1st 45 degrees in 256 steps.

Definition at line 60 of file FastMath.h.

6.1.4.7 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<sup>2</sup>.*
- [FloatType](#) & [accelY](#) = [measureVector](#) [MEASURE\_IND\_ACC\_Y]  
*Acceleration along the Y axis in m/s<sup>2</sup>.*
- [FloatType](#) & [accelZ](#) = [measureVector](#) [MEASURE\_IND\_ACC\_Z]  
*Acceleration along the Z axis in m/s<sup>2</sup>.*
- [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  $\text{m/s}^2$ .  
**MEASURE\_IND\_ACC\_Y** Acceleration along the Y axis in  $\text{m/s}^2$ .  
**MEASURE\_IND\_ACC\_Z** Acceleration along the Z axis in  $\text{m/s}^2$ .  
**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  $\text{m/s}^2$ .

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  $\text{m/s}^2$ .

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  $\text{m/s}^2$ .

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  $\text{m/s}^2$
- Accel Y axis STATUS\_IND\_ACC\_Y **Acceleration along Y axis** in  $\text{m/s}^2$
- Accel Z axis STATUS\_IND\_ACC\_Z **Acceleration along Y axis** in  $\text{m/s}^2$

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  $\text{m/s}^2$  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  $\text{m/s}^2$  on the Y axis of the plane.

**STATUS\_IND\_ACC\_Z** Acceleration in  $\text{m/s}^2$  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  $\text{m/s}^2$  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  $\text{m/s}^2$  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  $\text{m/s}^2$  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](https://de.wikipedia.org/wiki/Eulersche_Winkel#Drehfolgen_in_der_Fahrzeugtechnik)

$M_{\{G \rightarrow N\}} = \{p_{matrix}\} \cdot \{r_{matrix}\} \cdot \{a_{matrix}\}$

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](#)



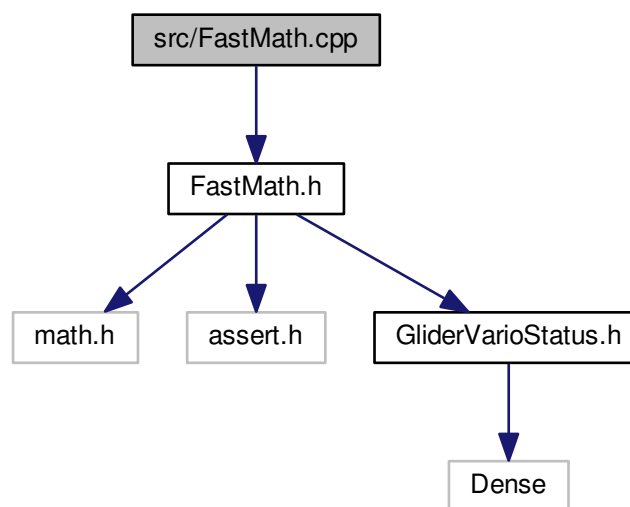
## Chapter 7

# File Documentation

### 7.1 src/FastMath.cpp File Reference

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

Include dependency graph for FastMath.cpp:



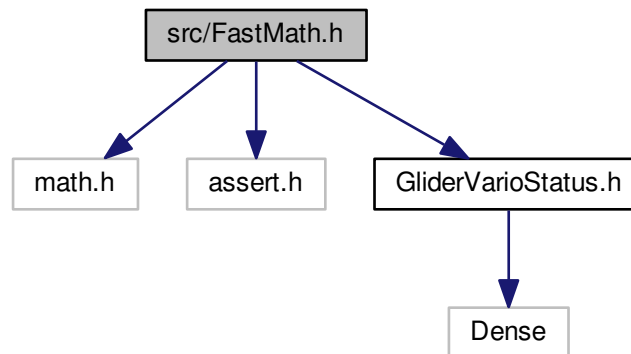
### Namespaces

- [openEV](#)

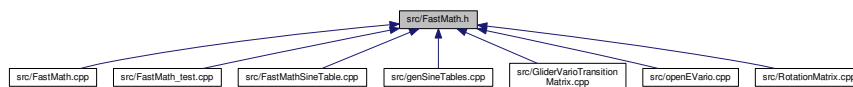
### 7.2 src/FastMath.h File Reference

```
#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

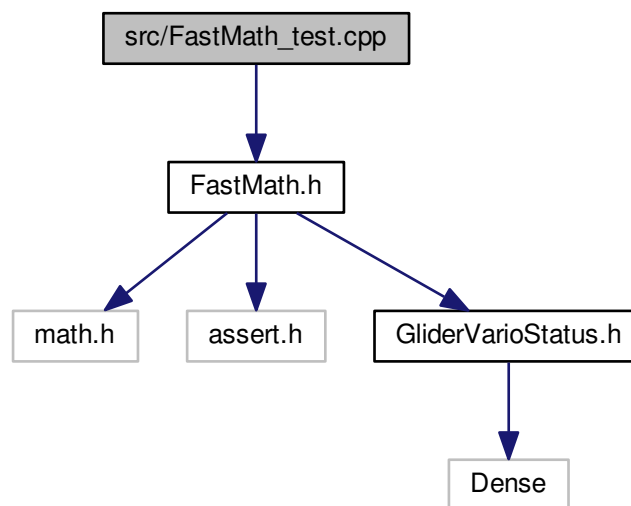
#### 7.2.1.1 `#define M_PI 3.14159265358979323846 /* pi */`

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

### 7.3 src/FastMath\_test.cpp File Reference

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

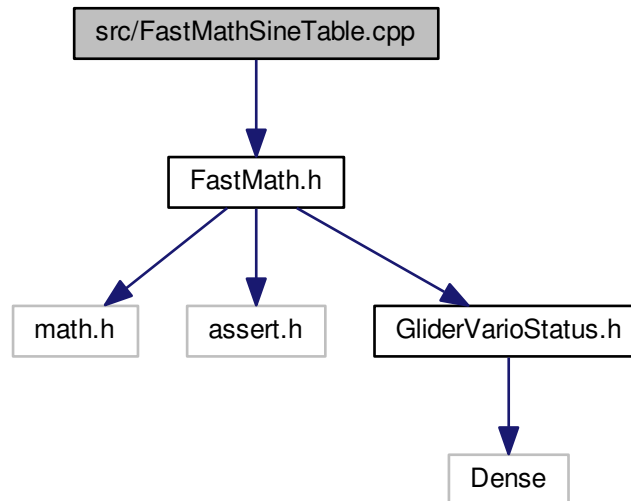
Include dependency graph for FastMath\_test.cpp:



## 7.4 src/FastMathSineTable.cpp File Reference

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

Include dependency graph for FastMathSineTable.cpp:



## Namespaces

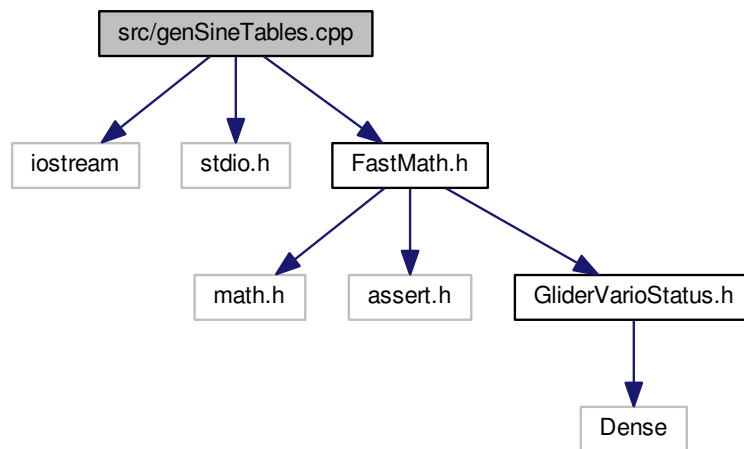
- [openEV](#)

## 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 134 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`.

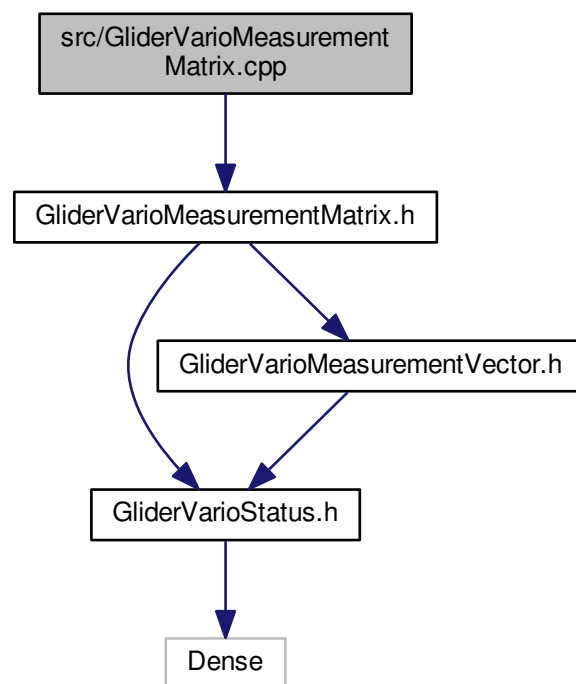
### 7.5.1.3 static void usage ( ) [static]

Definition at line 130 of file genSineTables.cpp.

## 7.6 src/GliderVarioMeasurementMatrix.cpp File Reference

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

Include dependency graph for GliderVarioMeasurementMatrix.cpp:



### Namespaces

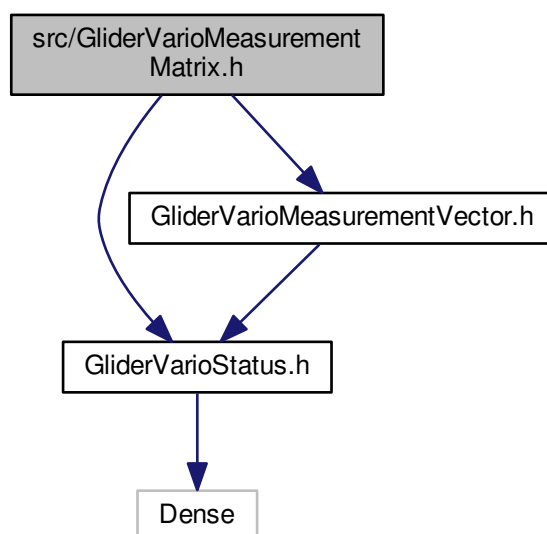
- [openEV](#)

## 7.7 src/GliderVarioMeasurementMatrix.h File Reference

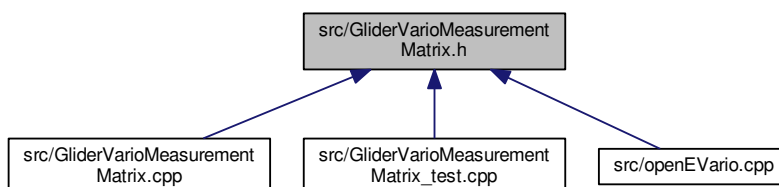
```
#include "GliderVarioStatus.h"
```

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

Include dependency graph for GliderVarioMeasurementMatrix.h:



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



## Classes

- class [openEV::GliderVarioMeasurementMatrix](#)

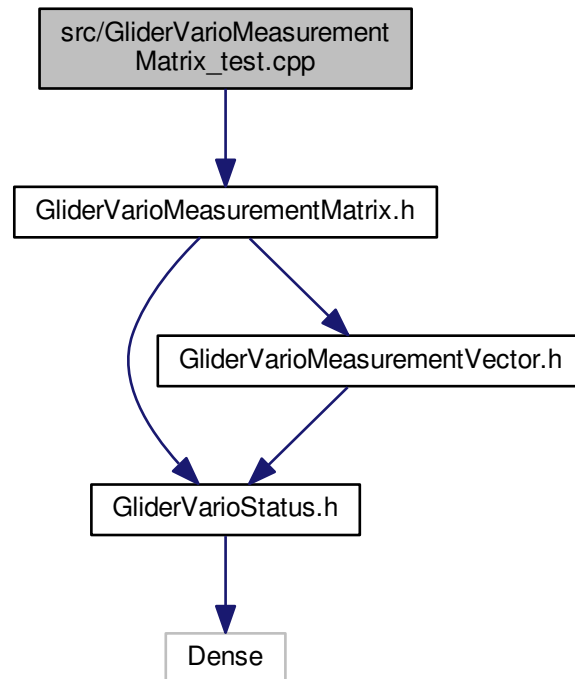
## Namespaces

- [openEV](#)

## 7.8 src/GliderVarioMeasurementMatrix\_test.cpp File Reference

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

Include dependency graph for GliderVarioMeasurementMatrix\_test.cpp:



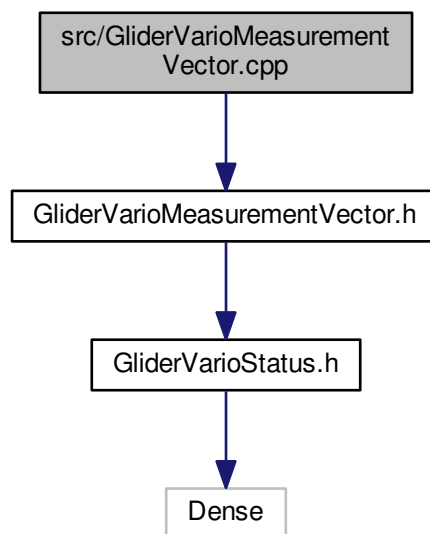
## Namespaces

- [openEV](#)

## 7.9 src/GliderVarioMeasurementVector.cpp File Reference

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

Include dependency graph for GliderVarioMeasurementVector.cpp:



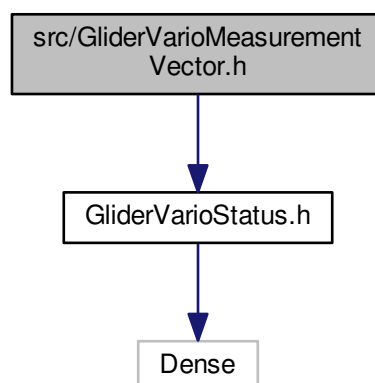
### Namespaces

- [openEV](#)

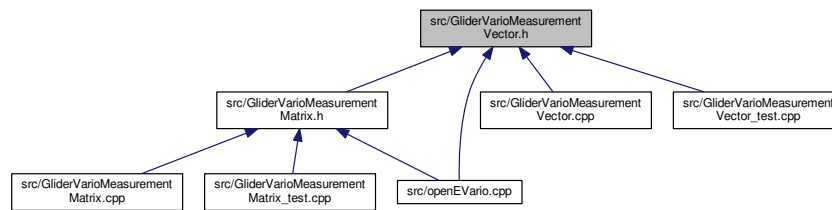
## 7.10 src/GliderVarioMeasurementVector.h File Reference

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

Include dependency graph for GliderVarioMeasurementVector.h:



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



## Classes

- class [openEV::GliderVarioMeasurementVector](#)

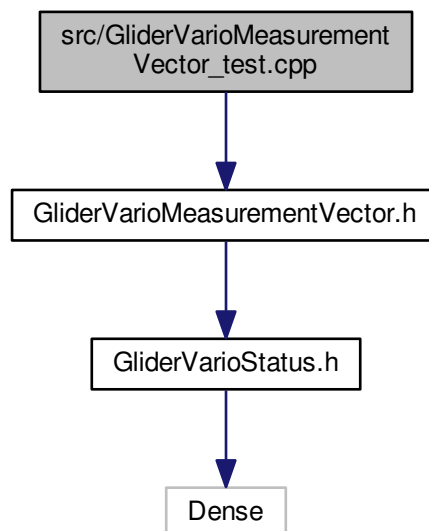
## Namespaces

- [openEV](#)

## 7.11 src/GliderVarioMeasurementVector\_test.cpp File Reference

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

Include dependency graph for `GliderVarioMeasurementVector_test.cpp`:



## Namespaces

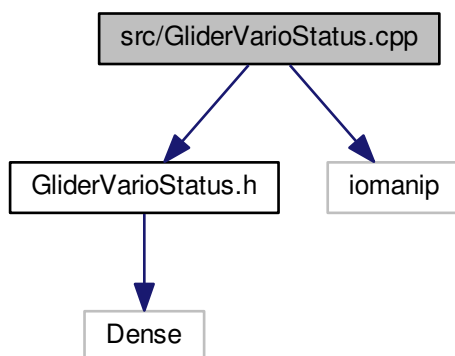
- [openEV](#)

## 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`.

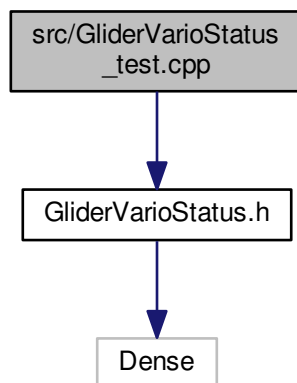




## 7.14 src/GliderVarioStatus\_test.cpp File Reference

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

Include dependency graph for GliderVarioStatus\_test.cpp:



## 7.15 src/GliderVarioTransitionMatrix.cpp File Reference

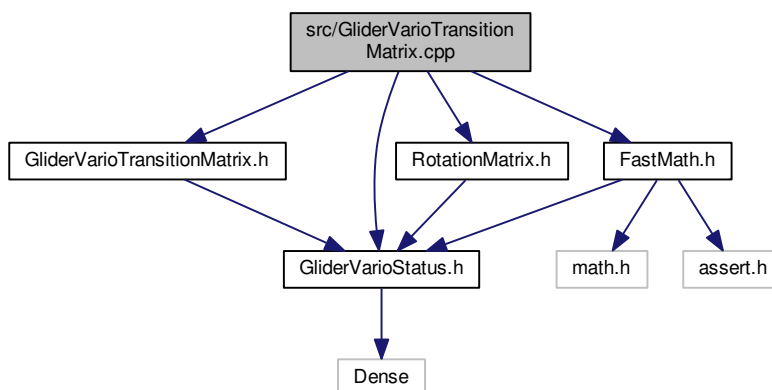
```
#include <GliderVarioTransitionMatrix.h>
```

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

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

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

Include dependency graph for GliderVarioTransitionMatrix.cpp:



### Namespaces

- [openEV](#)

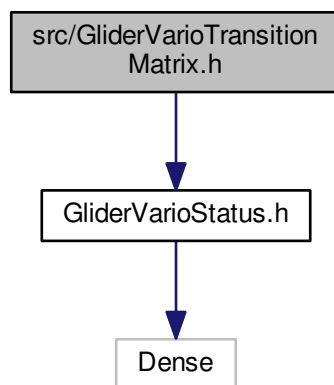
## Variables

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

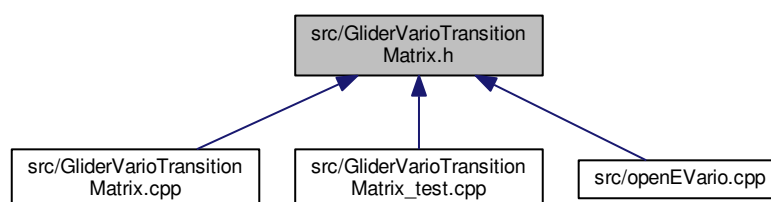
## 7.16 src/GliderVarioTransitionMatrix.h File Reference

```
#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](#)

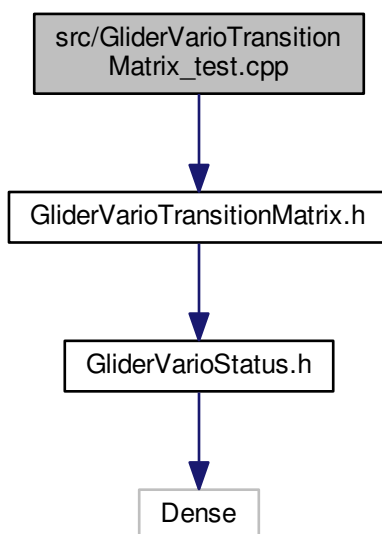
## Variables

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

## 7.17 src/GliderVarioTransitionMatrix\_test.cpp File Reference

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

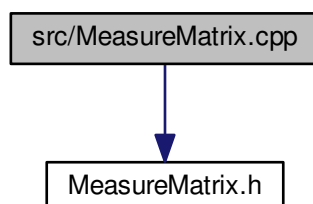
Include dependency graph for GliderVarioTransitionMatrix\_test.cpp:



## 7.18 src/MeasureMatrix.cpp File Reference

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

Include dependency graph for MeasureMatrix.cpp:

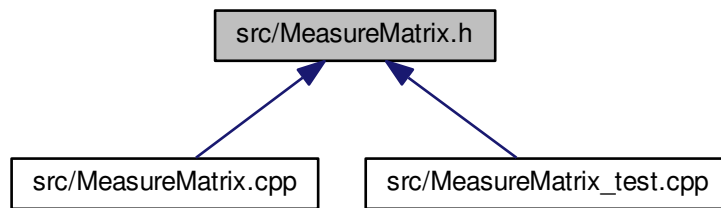


## Namespaces

- [openEV](#)

## 7.19 src/MeasureMatrix.h File Reference

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



## Classes

- class [openEV::MeasureMatrix](#)

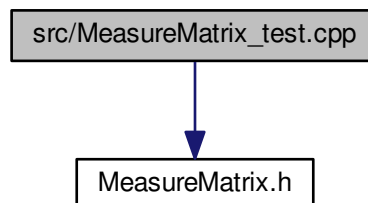
## Namespaces

- [openEV](#)

## 7.20 src/MeasureMatrix\_test.cpp File Reference

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

Include dependency graph for `MeasureMatrix_test.cpp`:



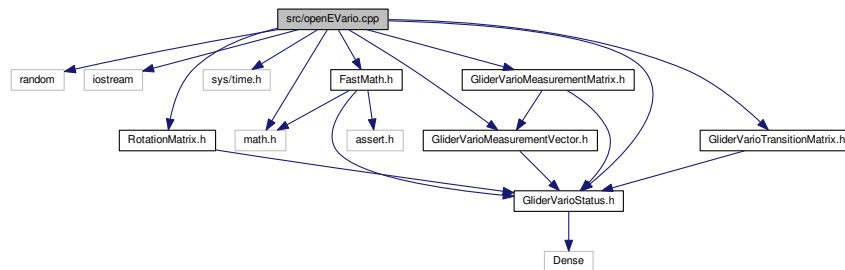
## Namespaces

- [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

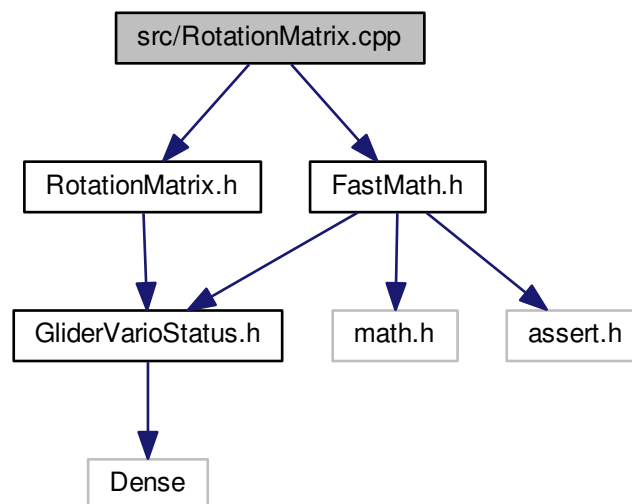
Definition at line 45 of file openEVario.cpp.

## 7.22 src/RotationMatrix.cpp File Reference

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

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

Include dependency graph for RotationMatrix.cpp:



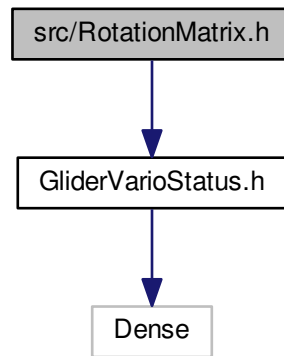
## Namespaces

- [openEV](#)

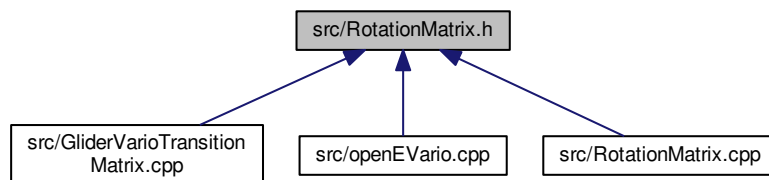
## 7.23 src/RotationMatrix.h File Reference

```
#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, [16](#)
- ~GliderVarioMeasurementVector
  - openEV::GliderVarioMeasurementVector, [18](#)
- ~GliderVarioStatus
  - openEV::GliderVarioStatus, [25](#)
- ~GliderVarioTransitionMatrix
  - openEV::GliderVarioTransitionMatrix, [29](#)
- ~MeasureMatrix
  - openEV::MeasureMatrix, [30](#)
- ~RotationMatrix
  - openEV::RotationMatrix, [32](#)
- accelX
  - openEV::GliderVarioMeasurementVector, [19](#)
  - openEV::GliderVarioStatus, [25](#)
- accelY
  - openEV::GliderVarioMeasurementVector, [19](#)
  - openEV::GliderVarioStatus, [25](#)
- accelZ
  - openEV::GliderVarioMeasurementVector, [19](#)
  - openEV::GliderVarioStatus, [25](#)
- altMSL
  - openEV::GliderVarioStatus, [25](#)
- atanTable
  - openEV::FastMath, [14](#)
- calcMeasurementMatrix
  - openEV::GliderVarioMeasurementMatrix, [16](#)
- calcPlaneVectorToWorldVector
  - openEV::RotationMatrix, [32](#)
- calcTransitionMatrix
  - openEV::GliderVarioTransitionMatrix, [29](#)
- calcWorldVectorToPlaneVector
  - openEV::RotationMatrix, [32](#)
- calculateRotationMatrixGloToPlane
  - openEV::RotationMatrix, [32](#)
- calculateRotationMatrixPlaneToGlo
  - openEV::RotationMatrix, [32](#)
- degToRad
  - openEV::FastMath, [14](#)
- fastATan2
  - openEV::FastMath, [12](#)
- fastATan2Pos
  - openEV::FastMath, [12](#)
- fastATanRaw
  - openEV::FastMath, [13](#)
- fastCos
  - openEV::FastMath, [13](#)
- FastMath
  - openEV::FastMath, [12](#)
- FastMath.h
  - M\_PI, [38](#)
- fastSin
  - openEV::FastMath, [13](#)
- fastSinPositive
  - openEV::FastMath, [13](#)
- fastSinRaw
  - openEV::FastMath, [14](#)
- FloatType
  - openEV, [9](#)
- GRAVITY
  - openEV, [10](#)
- genSineTables.cpp
  - main, [41](#)
  - printSineTable, [41](#)
  - usage, [41](#)
- getMatrixGloToPlane
  - openEV::RotationMatrix, [34](#)
- getMatrixPlaneToGlo
  - openEV::RotationMatrix, [34](#)
- getMeasureMatrix
  - openEV::GliderVarioMeasurementMatrix, [16](#)
- getMeasureVector
  - openEV::GliderVarioMeasurementVector, [18](#)
- getPitch
  - openEV::RotationMatrix, [34](#)
- getRoll
  - openEV::RotationMatrix, [34](#)
- getStatusVector
  - openEV::GliderVarioStatus, [25](#)
- getTransitionMatrix
  - openEV::GliderVarioTransitionMatrix, [29](#)
- getYaw
  - openEV::RotationMatrix, [34](#)
- GliderVarioMeasurementMatrix
  - openEV::GliderVarioMeasurementMatrix, [16](#)
- GliderVarioMeasurementVector
  - openEV::GliderVarioMeasurementVector, [18](#)
- GliderVarioStatus
  - openEV::GliderVarioStatus, [25](#)
- GliderVarioStatus.cpp
  - operator<<, [47](#)
- GliderVarioStatus.h
  - operator<<, [48](#)

- GliderVarioTransitionMatrix
  - openEV::GliderVarioTransitionMatrix, [29](#)
- gpsHeading
  - openEV::GliderVarioMeasurementVector, [19](#)
- gpsLatitude
  - openEV::GliderVarioMeasurementVector, [19](#)
- gpsLongitude
  - openEV::GliderVarioMeasurementVector, [19](#)
- gpsMSL
  - openEV::GliderVarioMeasurementVector, [19](#)
- gpsSpeed
  - openEV::GliderVarioMeasurementVector, [19](#)
- groundSpeedEast
  - openEV::GliderVarioStatus, [25](#)
- groundSpeedNorth
  - openEV::GliderVarioStatus, [26](#)
- gyroBiasX
  - openEV::GliderVarioStatus, [26](#)
- gyroBiasY
  - openEV::GliderVarioStatus, [26](#)
- gyroBiasZ
  - openEV::GliderVarioStatus, [26](#)
- gyroRateX
  - openEV::GliderVarioMeasurementVector, [19](#)
- gyroRateY
  - openEV::GliderVarioMeasurementVector, [20](#)
- gyroRateZ
  - openEV::GliderVarioMeasurementVector, [20](#)
- heading
  - openEV::GliderVarioStatus, [26](#)
- latitude
  - openEV::GliderVarioStatus, [26](#)
- lenLatitude
  - openEV, [10](#)
- longitude
  - openEV::GliderVarioStatus, [26](#)
- M\_PI
  - FastMath.h, [38](#)
- MEASURE\_IND\_ACC\_X
  - openEV::GliderVarioMeasurementVector, [18](#)
- MEASURE\_IND\_ACC\_Y
  - openEV::GliderVarioMeasurementVector, [18](#)
- MEASURE\_IND\_ACC\_Z
  - openEV::GliderVarioMeasurementVector, [18](#)
- MEASURE\_IND\_GPS\_ALTMSL
  - openEV::GliderVarioMeasurementVector, [18](#)
- MEASURE\_IND\_GPS\_HEADING
  - openEV::GliderVarioMeasurementVector, [18](#)
- MEASURE\_IND\_GPS\_LAT
  - openEV::GliderVarioMeasurementVector, [18](#)
- MEASURE\_IND\_GPS\_LON
  - openEV::GliderVarioMeasurementVector, [18](#)
- MEASURE\_IND\_GPS\_SPEED
  - openEV::GliderVarioMeasurementVector, [18](#)
- MEASURE\_IND\_GYRO\_RATE\_X
  - openEV::GliderVarioMeasurementVector, [18](#)
- MEASURE\_IND\_GYRO\_RATE\_Y
  - openEV::GliderVarioMeasurementVector, [18](#)
- MEASURE\_IND\_GYRO\_RATE\_Z
  - openEV::GliderVarioMeasurementVector, [18](#)
- MEASURE\_IND\_MAG\_X
  - openEV::GliderVarioMeasurementVector, [18](#)
- MEASURE\_IND\_MAG\_Y
  - openEV::GliderVarioMeasurementVector, [18](#)
- MEASURE\_IND\_MAG\_Z
  - openEV::GliderVarioMeasurementVector, [18](#)
- MEASURE\_IND\_PRESS\_ALT
  - openEV::GliderVarioMeasurementVector, [18](#)
- MEASURE\_IND\_TAS
  - openEV::GliderVarioMeasurementVector, [18](#)
- MEASURE\_NUM\_ROWS
  - openEV::GliderVarioMeasurementVector, [18](#)
- magX
  - openEV::GliderVarioMeasurementVector, [20](#)
- magY
  - openEV::GliderVarioMeasurementVector, [20](#)
- magZ
  - openEV::GliderVarioMeasurementVector, [20](#)
- main
  - genSineTables.cpp, [41](#)
  - openEVario.cpp, [53](#)
- matrixGloToPlane
  - openEV::RotationMatrix, [34](#)
- matrixGloToPlanelValid
  - openEV::RotationMatrix, [35](#)
- matrixPlaneToGlo
  - openEV::RotationMatrix, [35](#)
- matrixPlaneToGloValid
  - openEV::RotationMatrix, [35](#)
- MeasureComponentIndex
  - openEV::GliderVarioMeasurementVector, [18](#)
- MeasureMatrix
  - openEV::MeasureMatrix, [30](#)
- MeasureMatrixType
  - openEV::GliderVarioMeasurementMatrix, [15](#)
- measureVector
  - openEV::GliderVarioMeasurementVector, [20](#)
- MeasureVectorType
  - openEV::GliderVarioMeasurementVector, [18](#)
- measurementMatrix
  - openEV::GliderVarioMeasurementMatrix, [16](#)
- normalizeAngles
  - openEV::GliderVarioStatus, [25](#)
- openEV, [9](#)
  - FloatType, [9](#)
  - GRAVITY, [10](#)
  - lenLatitude, [10](#)
  - Vector3DType, [9](#)
- openEV::FastMath, [11](#)
  - ~FastMath, [12](#)
  - atanTable, [14](#)
  - degToRad, [14](#)
  - fastATan2, [12](#)

- fastATan2Pos, [12](#)
- fastATanRaw, [13](#)
- fastCos, [13](#)
- FastMath, [12](#)
- fastSin, [13](#)
- fastSinPositive, [13](#)
- fastSinRaw, [14](#)
- radToDeg, [14](#)
- sineSamplesPerDegree, [14](#)
- sinusTable, [14](#)
- sizeATanTable, [14](#)
- sizeSineTable, [15](#)
- openEV::GliderVarioMeasurementMatrix, [15](#)
  - ~GliderVarioMeasurementMatrix, [16](#)
  - calcMeasurementMatrix, [16](#)
  - getMeasureMatrix, [16](#)
  - GliderVarioMeasurementMatrix, [16](#)
  - MeasureMatrixType, [15](#)
  - measurementMatrix, [16](#)
- openEV::GliderVarioMeasurementVector, [16](#)
  - ~GliderVarioMeasurementVector, [18](#)
  - accelX, [19](#)
  - accelY, [19](#)
  - accelZ, [19](#)
  - getMeasureVector, [18](#)
  - GliderVarioMeasurementVector, [18](#)
  - gpsHeading, [19](#)
  - gpsLatitude, [19](#)
  - gpsLongitude, [19](#)
  - gpsMSL, [19](#)
  - gpsSpeed, [19](#)
  - gyroRateX, [19](#)
  - gyroRateY, [20](#)
  - gyroRateZ, [20](#)
  - MEASURE\_IND\_ACC\_X, [18](#)
  - MEASURE\_IND\_ACC\_Y, [18](#)
  - MEASURE\_IND\_ACC\_Z, [18](#)
  - MEASURE\_IND\_GPS\_ALTMSL, [18](#)
  - MEASURE\_IND\_GPS\_HEADING, [18](#)
  - MEASURE\_IND\_GPS\_LAT, [18](#)
  - MEASURE\_IND\_GPS\_LON, [18](#)
  - MEASURE\_IND\_GPS\_SPEED, [18](#)
  - MEASURE\_IND\_GYRO\_RATE\_X, [18](#)
  - MEASURE\_IND\_GYRO\_RATE\_Y, [18](#)
  - MEASURE\_IND\_GYRO\_RATE\_Z, [18](#)
  - MEASURE\_IND\_MAG\_X, [18](#)
  - MEASURE\_IND\_MAG\_Y, [18](#)
  - MEASURE\_IND\_MAG\_Z, [18](#)
  - MEASURE\_IND\_PRESS\_ALT, [18](#)
  - MEASURE\_IND\_TAS, [18](#)
  - MEASURE\_NUM\_ROWS, [18](#)
  - magX, [20](#)
  - magY, [20](#)
  - magZ, [20](#)
  - MeasureComponentIndex, [18](#)
  - measureVector, [20](#)
  - MeasureVectorType, [18](#)
  - pressAlt, [20](#)
  - trueAirSpeed, [20](#)
- openEV::GliderVarioStatus, [21](#)
  - ~GliderVarioStatus, [25](#)
  - accelX, [25](#)
  - accelY, [25](#)
  - accelZ, [25](#)
  - altMSL, [25](#)
  - getStatusVector, [25](#)
  - GliderVarioStatus, [25](#)
  - groundSpeedEast, [25](#)
  - groundSpeedNorth, [26](#)
  - gyroBiasX, [26](#)
  - gyroBiasY, [26](#)
  - gyroBiasZ, [26](#)
  - heading, [26](#)
  - latitude, [26](#)
  - longitude, [26](#)
  - normalizeAngles, [25](#)
  - pitchAngle, [26](#)
  - pitchRateY, [26](#)
  - rateOfSink, [27](#)
  - rollAngle, [27](#)
  - rollRateX, [27](#)
  - STATUS\_IND\_ACC\_X, [24](#)
  - STATUS\_IND\_ACC\_Y, [24](#)
  - STATUS\_IND\_ACC\_Z, [24](#)
  - STATUS\_IND\_ALT\_MSL, [24](#)
  - STATUS\_IND\_GYRO\_BIAS\_X, [24](#)
  - STATUS\_IND\_GYRO\_BIAS\_Y, [24](#)
  - STATUS\_IND\_GYRO\_BIAS\_Z, [24](#)
  - STATUS\_IND\_HEADING, [24](#)
  - STATUS\_IND\_LATITUDE, [24](#)
  - STATUS\_IND\_LONGITUDE, [24](#)
  - STATUS\_IND\_PITCH, [24](#)
  - STATUS\_IND\_RATE\_OF\_SINK, [24](#)
  - STATUS\_IND\_ROLL, [24](#)
  - STATUS\_IND\_ROTATION\_X, [24](#)
  - STATUS\_IND\_ROTATION\_Y, [24](#)
  - STATUS\_IND\_ROTATION\_Z, [24](#)
  - STATUS\_IND\_SPEED\_GROUND\_E, [24](#)
  - STATUS\_IND\_SPEED\_GROUND\_N, [24](#)
  - STATUS\_IND\_TAS, [24](#)
  - STATUS\_IND\_THERMAL\_SPEED, [24](#)
  - STATUS\_IND\_VERTICAL\_SPEED, [24](#)
  - STATUS\_IND\_WIND\_SPEED\_E, [24](#)
  - STATUS\_IND\_WIND\_SPEED\_N, [24](#)
  - STATUS\_IND\_YAW, [24](#)
  - STATUS\_NUM\_ROWS, [24](#)
  - StatusComponentIndex, [24](#)
  - statusVector, [27](#)
  - StatusVectorType, [24](#)
  - thermalSpeed, [27](#)
  - trueAirSpeed, [27](#)
  - verticalSpeed, [27](#)
  - windSpeedEast, [27](#)
  - windSpeedNorth, [27](#)
  - yawAngle, [28](#)
  - yawRateZ, [28](#)

- openEV::GliderVarioTransitionMatrix, 28
  - ~GliderVarioTransitionMatrix, 29
  - calcTransitionMatrix, 29
  - getTransitionMatrix, 29
  - GliderVarioTransitionMatrix, 29
  - transitionMatrix, 29
  - TransitionMatrixType, 29
  - updateStatus, 29
- openEV::MeasureMatrix, 30
  - ~MeasureMatrix, 30
  - MeasureMatrix, 30
- openEV::RotationMatrix, 30
  - ~RotationMatrix, 32
  - calcPlaneVectorToWorldVector, 32
  - calcWorldVectorToPlaneVector, 32
  - calculateRotationMatrixGloToPlane, 32
  - calculateRotationMatrixPlaneToGlo, 32
  - getMatrixGloToPlane, 34
  - getMatrixPlaneToGlo, 34
  - getPitch, 34
  - getRoll, 34
  - getYaw, 34
  - matrixGloToPlane, 34
  - matrixGloToPlaneIsValid, 35
  - matrixPlaneToGlo, 35
  - matrixPlaneToGloIsValid, 35
  - pitch, 35
  - roll, 35
  - RotationMatrix, 32
  - RotationMatrixType, 31
  - setPitch, 34
  - setRoll, 34
  - setYaw, 34
  - yaw, 35
- openEVario.cpp
  - main, 53
  - randomGenerator, 54
  - x, 54
- operator<<
  - GliderVarioStatus.cpp, 47
  - GliderVarioStatus.h, 48
- pitch
  - openEV::RotationMatrix, 35
- pitchAngle
  - openEV::GliderVarioStatus, 26
- pitchRateY
  - openEV::GliderVarioStatus, 26
- pressAlt
  - openEV::GliderVarioMeasurementVector, 20
- printSineTable
  - genSineTables.cpp, 41
- radToDeg
  - openEV::FastMath, 14
- randomGenerator
  - openEVario.cpp, 54
- rateOfSink
  - openEV::GliderVarioStatus, 27
- roll
  - openEV::RotationMatrix, 35
- rollAngle
  - openEV::GliderVarioStatus, 27
- rollRateX
  - openEV::GliderVarioStatus, 27
- RotationMatrix
  - openEV::RotationMatrix, 32
- RotationMatrixType
  - openEV::RotationMatrix, 31
- STATUS\_IND\_ACC\_X
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_ACC\_Y
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_ACC\_Z
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_ALT\_MSL
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_GYRO\_BIAS\_X
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_GYRO\_BIAS\_Y
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_GYRO\_BIAS\_Z
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_HEADING
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_LATITUDE
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_LONGITUDE
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_PITCH
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_RATE\_OF\_SINK
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_ROLL
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_ROTATION\_X
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_ROTATION\_Y
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_ROTATION\_Z
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_SPEED\_GROUND\_E
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_SPEED\_GROUND\_N
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_TAS
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_THERMAL\_SPEED
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_VERTICAL\_SPEED
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_WIND\_SPEED\_E
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_WIND\_SPEED\_N
  - openEV::GliderVarioStatus, 24
- STATUS\_IND\_YAW
  - openEV::GliderVarioStatus, 24

- STATUS\_NUM\_ROWS
  - openEV::GliderVarioStatus, [24](#)
- setPitch
  - openEV::RotationMatrix, [34](#)
- setRoll
  - openEV::RotationMatrix, [34](#)
- setYaw
  - openEV::RotationMatrix, [34](#)
- sineSamplesPerDegree
  - openEV::FastMath, [14](#)
- sinusTable
  - openEV::FastMath, [14](#)
- sizeATanTable
  - openEV::FastMath, [14](#)
- sizeSineTable
  - openEV::FastMath, [15](#)
- src/FastMath.cpp, [37](#)
- src/FastMath.h, [37](#)
- src/FastMath\_test.cpp, [39](#)
- src/FastMathSineTable.cpp, [40](#)
- src/GliderVarioMeasurementMatrix.cpp, [42](#)
- src/GliderVarioMeasurementMatrix.h, [42](#)
- src/GliderVarioMeasurementMatrix\_test.cpp, [43](#)
- src/GliderVarioMeasurementVector.cpp, [44](#)
- src/GliderVarioMeasurementVector.h, [45](#)
- src/GliderVarioMeasurementVector\_test.cpp, [46](#)
- src/GliderVarioStatus.cpp, [47](#)
- src/GliderVarioStatus.h, [48](#)
- src/GliderVarioStatus\_test.cpp, [49](#)
- src/GliderVarioTransitionMatrix.cpp, [49](#)
- src/GliderVarioTransitionMatrix.h, [50](#)
- src/GliderVarioTransitionMatrix\_test.cpp, [51](#)
- src/MeasureMatrix.cpp, [51](#)
- src/MeasureMatrix.h, [52](#)
- src/MeasureMatrix\_test.cpp, [52](#)
- src/RotationMatrix.cpp, [54](#)
- src/RotationMatrix.h, [55](#)
- src/genSineTables.cpp, [40](#)
- src/openEVario.cpp, [53](#)
- StatusComponentIndex
  - openEV::GliderVarioStatus, [24](#)
- statusVector
  - openEV::GliderVarioStatus, [27](#)
- StatusVectorType
  - openEV::GliderVarioStatus, [24](#)
- thermalSpeed
  - openEV::GliderVarioStatus, [27](#)
- transitionMatrix
  - openEV::GliderVarioTransitionMatrix, [29](#)
- TransitionMatrixType
  - openEV::GliderVarioTransitionMatrix, [29](#)
- trueAirSpeed
  - openEV::GliderVarioMeasurementVector, [20](#)
  - openEV::GliderVarioStatus, [27](#)
- updateStatus
  - openEV::GliderVarioTransitionMatrix, [29](#)
- usage
  - genSineTables.cpp, [41](#)
- Vector3DType
  - openEV, [9](#)
- verticalSpeed
  - openEV::GliderVarioStatus, [27](#)
- windSpeedEast
  - openEV::GliderVarioStatus, [27](#)
- windSpeedNorth
  - openEV::GliderVarioStatus, [27](#)
- x
  - openEVario.cpp, [54](#)
- yaw
  - openEV::RotationMatrix, [35](#)
- yawAngle
  - openEV::GliderVarioStatus, [28](#)
- yawRateZ
  - openEV::GliderVarioStatus, [28](#)