openEVario

0.1

Generated by Doxygen 1.8.9.1

Mon Mar 21 2016 00:20:31

Contents

Chapter 1

Todo List

Member openEV::GliderVarioMeasurementMatrix::calcMeasurementMatrix (FloatType timeDiff, GliderVarioStatus const &lastStatus)

Normalized magnetometer values to direction

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:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

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
- class RotMatrixConversion

Typedefs

- typedef float FloatType
- typedef Eigen::Matrix< FloatType, 3, 1 > Vector3DType
- typedef Eigen::Matrix< FloatType, 3, 3 > RotationMatrix3DType

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 42 of file GliderVarioStatus.h.

5.1.1.2 typedef Eigen::Matrix<FloatType, 3, 3> openEV::RotationMatrix3DType

Definition at line 48 of file GliderVarioStatus.h.

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

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

Definition at line 47 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 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

in	*x*	component
in	*y*	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 >= 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	*tanVal*	tan value, i.e. the ratio of x and y. tan value *must* be $>=$ 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

angle[in]	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

angle[in]	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

angle[in]	in degrees. The angle MUST be >= 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

angle[in]	in degrees. The angle *must* >= 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, GliderVario←
  Status::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,Glider← VarioStatus::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*)

Todo Normalized magnetometer values to direction

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_I←ND_GPS_HEADING,
MEASURE_IND_GPS_SPEED, MEASURE_IND_ACC_X, MEASURE_IND_ACC_Y, MEASURE_IND_A←CC_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::GliderVarioMeasurement↩ Vector::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_PITCH,
STATUS_IND_ROLL, STATUS_IND_SPEED_GROUND_N, STATUS_IND_SPEED_GROUND_E, STAT↩
US_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_ROTATIO↩
N_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 & 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. Nothern 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 104 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 156 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_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 z 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 112 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 162 of file GliderVarioStatus.h.

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

Definition at line 166 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 <= pitch <= 90; If you fly a looping and turn past perpendicular you essentially roll 180 deg, and reverse direction 180 deg -Roll: -180 <= roll < 180; 180 deg counts as -180 -Yaw: 0<= yaw < 360; 360 deg counts as 0. Note that pitch must be normalized fist. 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 199 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 200 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 201 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 184 of file GliderVarioStatus.h.

6.4.6.5 FloatType& openEV::GliderVarioStatus::groundSpeedEast = statusVector[STATUS_IND_SPEED_GROUN↩ D_E]

Ground speed component East in m/s.

Definition at line 191 of file GliderVarioStatus.h.

6.4.6.6 FloatType& openEV::GliderVarioStatus::groundSpeedNorth = statusVector[STATUS_IND_SPEED_GROUN↩ D_N]

Ground speed component North in m/s.

Definition at line 190 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 209 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 210 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 211 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 193 of file GliderVarioStatus.h.

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

Latitude in deg North.

Definition at line 183 of file GliderVarioStatus.h.

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

Longitude in deg. East.

Definition at line 182 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 186 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 205 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 194 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 187 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 204 of file GliderVarioStatus.h.

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

Definition at line 218 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 215 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 192 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 195 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 213 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 212 of file GliderVarioStatus.h.

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

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

Definition at line 206 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::STATU↩ S_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::STAT↩ US_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	*timeDiff*	Time since last update in seconds.
in	*lastStatus*	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	*oldStatus*	Last known status
out	*newStatus*	New status by extrapolation after timeDiff seconds
in	*timeDiff*	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 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)

- RotationMatrix3DType & getMatrixGloToPlane ()

  The rotation matrix from the global(world) coordinate system to the plane coordinate system.
- RotationMatrix3DType & getMatrixPlaneToGlo ()

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

Protected Member Functions

- void calculateRotationMatrixGloToPlane ()
- void calculateRotationMatrixPlaneToGlo ()

Protected Attributes

- RotationMatrix3DType matrixGloToPlane

  The rotation matrix from the global(world) coordinate system to the plane coordinate system.
- RotationMatrix3DType 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 Constructor & Destructor Documentation

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

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

Definition at line 53 of file RotationMatrix.h.

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

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

Definition at line 68 of file RotationMatrix.h.

6.7.2.3 openEV::RotationMatrix::∼RotationMatrix ( ) `[virtual]`

Definition at line 33 of file RotationMatrix.cpp.

6.7.3 Member Function Documentation

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

Convert the plane vector into the world vector

Parameters

planeVector[in]	The vector in plane coordinates
worldVector[out]	The vector in world coordinates

Definition at line 123 of file RotationMatrix.h.

6.7.3.2 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

{align} M_{GNR} & = {pmatrix} & & - \ - & + & \ + & - & {pmatrix} {align}

Definition at line 56 of file RotationMatrix.cpp.

6.7.3.3 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 179 of file RotationMatrix.h.

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

Convert the world vector into the plane vector

Parameters

worldVector[in]	The vector in world coordinates
planeVector[out]	The vector in plane coordinates

Definition at line 134 of file RotationMatrix.h.

6.7.3.5 RotationMatrix3DType& openEV::RotationMatrix::getMatrixGloToPlane ( ) `[inline]`

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

Definition at line 140 of file RotationMatrix.h.

6.7.3.6 RotationMatrix3DType& openEV::RotationMatrix::getMatrixPlaneToGlo ( ) `[inline]`

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

Definition at line 145 of file RotationMatrix.h.

6.7.3.7 FloatType openEV::RotationMatrix::getPitch ( ) `[inline]`

Definition at line 104 of file RotationMatrix.h.

6.7.3.8 FloatType openEV::RotationMatrix::getRoll ( ) [inline]

Definition at line 115 of file RotationMatrix.h.

6.7.3.9 FloatType openEV::RotationMatrix::getYaw ( ) [inline]

Definition at line 93 of file RotationMatrix.h.

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

set pitch angle . Invalidates the matrix.

Definition at line 97 of file RotationMatrix.h.

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

set roll angle . Invalidates the matrix.

Definition at line 108 of file RotationMatrix.h.

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

set yaw angle . Invalidates the matrix.

Definition at line 86 of file RotationMatrix.h.

6.7.4 Member Data Documentation

6.7.4.1 RotationMatrix3DType openEV::RotationMatrix::matrixGloToPlane [protected]

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

Definition at line 153 of file RotationMatrix.h.

6.7.4.2 bool openEV::RotationMatrix::matrixGloToPlaneIsValid [protected]

Definition at line 157 of file RotationMatrix.h.

6.7.4.3 RotationMatrix3DType openEV::RotationMatrix::matrixPlaneToGlo [protected]

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

Definition at line 155 of file RotationMatrix.h.

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

Definition at line 158 of file RotationMatrix.h.

6.7.4.5 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 164 of file RotationMatrix.h.

6.7.4.6 FloatType openEV::RotationMatrix::roll `[protected]`

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

Definition at line 166 of file RotationMatrix.h.

6.7.4.7 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 162 of file RotationMatrix.h.

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

- src/RotationMatrix.h
- src/RotationMatrix.cpp

6.8 openEV::RotMatrixConversion Class Reference

`#include <RotMatrixConversion.h>`

Public Member Functions

- RotMatrixConversion ()
- virtual ∼RotMatrixConversion ()

Static Public Member Functions

- static void vectors2RotMatrix (RotationMatrix3DType &rotMatrix, Vector3DType const &v1, Vector3DType const &v2)

Static Private Member Functions

- static int axisDominantV3Single (Vector3DType const &vec)
- static void orthoV3V3 (Vector3DType &p, Vector3DType const &v)
- static void axisAngleNormalizedToMat3Ex (RotationMatrix3DType ∗mat, Vector3DType const &axis, const FloatType angle_sin, const FloatType angle_cos)

6.8.1 Detailed Description

The algorithm to construct a rotation matrix from two vectors comes from: `http://stackoverflow.↩ com/questions/23166898/efficient-way-to-calculate-a-3x3-rotation-matrix-from-the-rotatio`

The algorithm to decompose a rotation matrix into the Euler angles comes from `http://nghiaho.↩ com/?page_id=846`

Definition at line 43 of file RotMatrixConversion.h.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 openEV::RotMatrixConversion::RotMatrixConversion ( ) `[inline]`

Definition at line 45 of file RotMatrixConversion.h.

6.8.2.2 openEV::RotMatrixConversion::∼RotMatrixConversion ( ) [virtual]

Definition at line 30 of file RotMatrixConversion.cpp.

6.8.3 Member Function Documentation

6.8.3.1 void openEV::RotMatrixConversion::axisAngleNormalizedToMat3Ex (RotationMatrix3DType ∗ *mat,* Vector3DType const & *axis,* const FloatType *angle_sin,* const FloatType *angle_cos*) [static], [private]

Does the basic calculation of the rotation matrix.

Parameters

mat[out]	Due to a gcc 5.3 bug I cannot pass mat as a reference but I have to pass the pointer.
axis	[in]
angle_sin[in]	
angle_cos[in]	

Definition at line 115 of file RotMatrixConversion.cpp.

6.8.3.2 int openEV::RotMatrixConversion::axisDominantV3Single (Vector3DType const & *vec*) [static], [private]

Determines the dominant axis in the vector, i.e. the dimension with the greatest length

Parameters

in	*vec*	

Returns

0: x-axis, 1: y-axis, 2: y-axis

Definition at line 81 of file RotMatrixConversion.cpp.

6.8.3.3 void openEV::RotMatrixConversion::orthoV3V3 (Vector3DType & *p,* Vector3DType const & *v*) [static], [private]

Calculate the orthogonal

Parameters

out	*p*	The orthogonal vector to v
	v[in]	Input.

Definition at line 91 of file RotMatrixConversion.cpp.

6.8.3.4 void openEV::RotMatrixConversion::vectors2RotMatrix (RotationMatrix3DType & *rotMatrix,* Vector3DType const & *v1,* Vector3DType const & *v2*) [static]

Calculate a rotation matrix from 2 normalized vectors. v1 and v2 must be unit length.

Parameters

rotMatrix[out]	The rotation matrix to map v1 to v2
v1[in]	the original vector
v2[in]	the resulting vector after being multiplied with rotMatrix.

Definition at line 43 of file RotMatrixConversion.cpp.

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

- src/RotMatrixConversion.h
- src/RotMatrixConversion.cpp

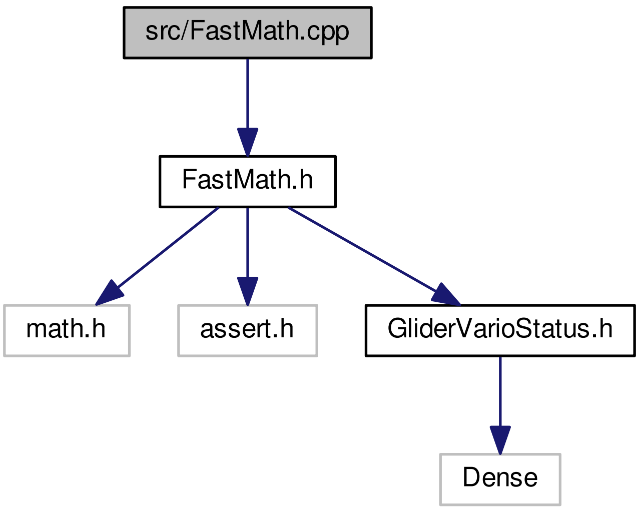
Chapter 7

File Documentation

7.1 src/FastMath.cpp File Reference

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

Include dependency graph for FastMath.cpp:



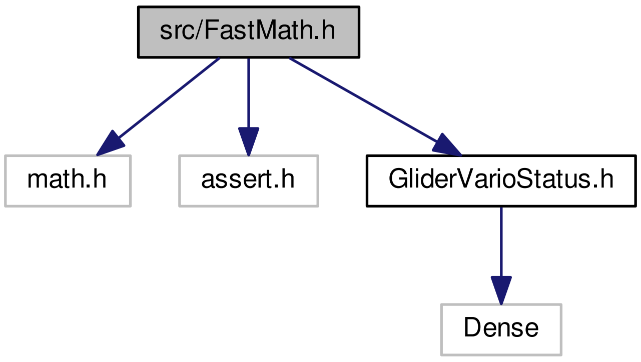
Namespaces

- openEV

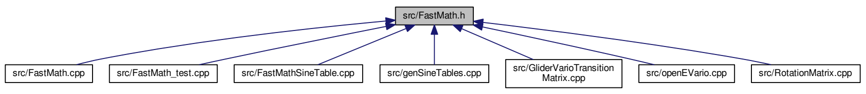
7.2 src/FastMath.h File Reference

```
#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

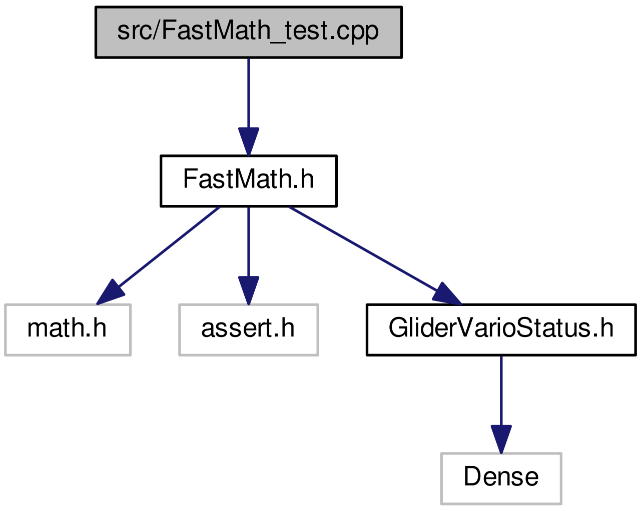
7.2.1.1 #define M_PI 3.14159265358979323846 /∗ pi ∗/

Definition at line 38 of file FastMath.h.

7.3 src/FastMath_test.cpp File Reference

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

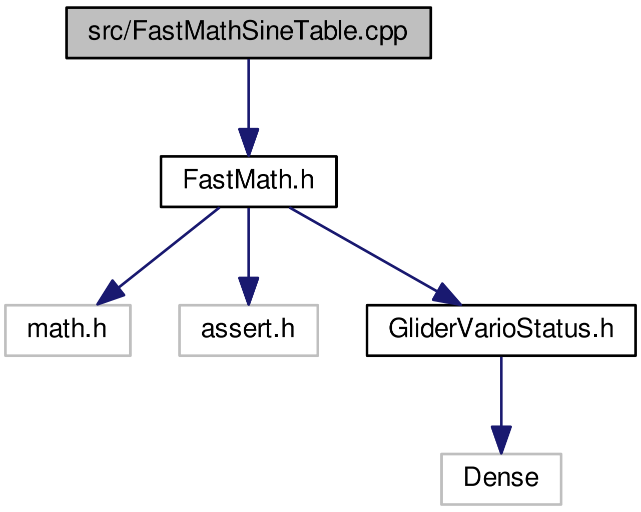
Include dependency graph for FastMath_test.cpp:



7.4 src/FastMathSineTable.cpp File Reference

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

Include dependency graph for FastMathSineTable.cpp:



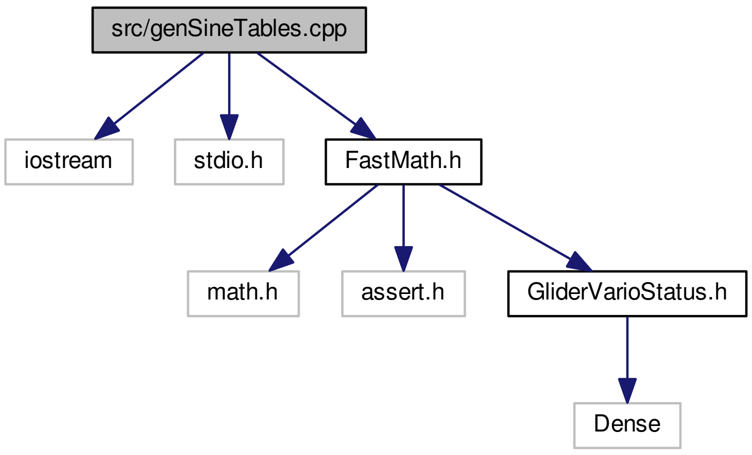
Namespaces

- 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.

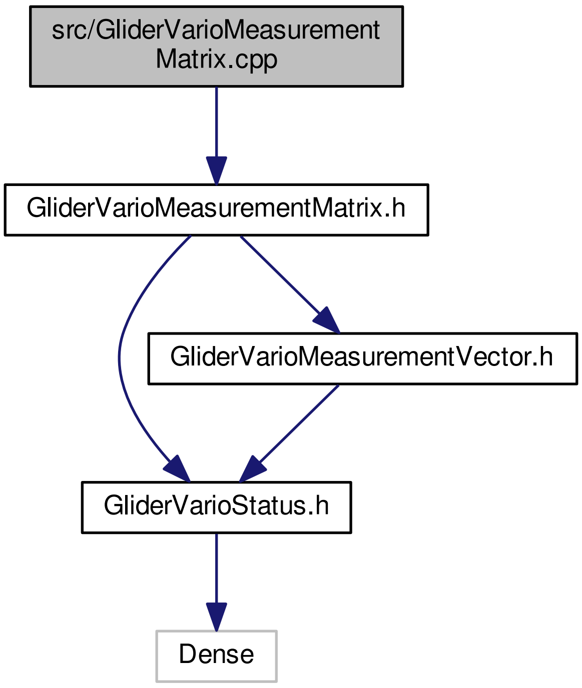
7.5.1.3 static void usage ( ) [static]

Definition at line 130 of file genSineTables.cpp.

7.6 src/GliderVarioMeasurementMatrix.cpp File Reference

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

Include dependency graph for GliderVarioMeasurementMatrix.cpp:


src/GliderVarioMeasurement
Matrix.cpp
GliderVarioMeasurementMatrix.h
GliderVarioMeasurementVector.h
GliderVarioStatus.h
Dense

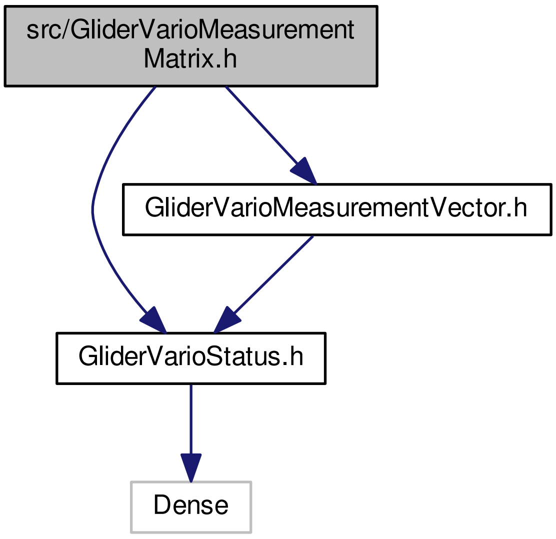
Namespaces

- openEV

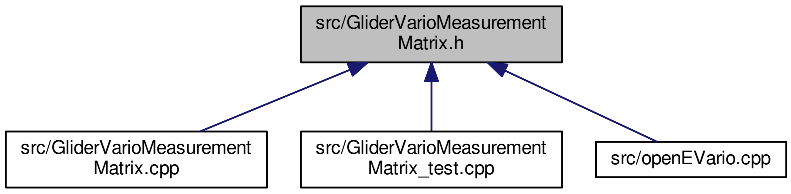
7.7 src/GliderVarioMeasurementMatrix.h File Reference

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

Include dependency graph for GliderVarioMeasurementMatrix.h:


src/GliderVarioMeasurement Matrix.h
GliderVarioMeasurementVector.h
GliderVarioStatus.h
Dense

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


src/GliderVarioMeasurement Matrix.h
src/GliderVarioMeasurement Matrix.cpp
src/GliderVarioMeasurement Matrix_test.cpp
src/openEVario.cpp

Classes

- class openEV::GliderVarioMeasurementMatrix

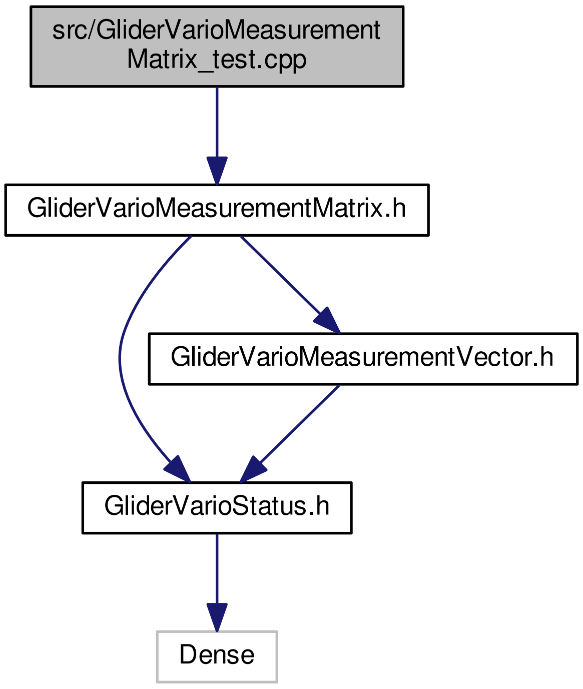
Namespaces

- openEV

7.8 src/GliderVarioMeasurementMatrix_test.cpp File Reference

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

Include dependency graph for GliderVarioMeasurementMatrix_test.cpp:


src/GliderVarioMeasurement Matrix_test.cpp
GliderVarioMeasurementMatrix.h
GliderVarioMeasurementVector.h
GliderVarioStatus.h
Dense

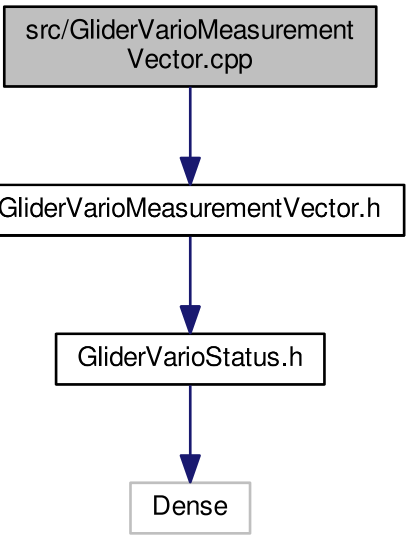
Namespaces

- openEV

7.9 src/GliderVarioMeasurementVector.cpp File Reference

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

Include dependency graph for GliderVarioMeasurementVector.cpp:


src/GliderVarioMeasurement Vector.cpp
GliderVarioMeasurementVector.h
GliderVarioStatus.h
Dense

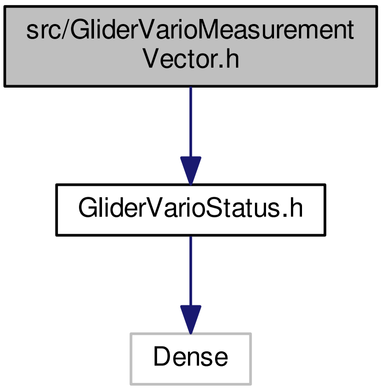
Namespaces

- openEV

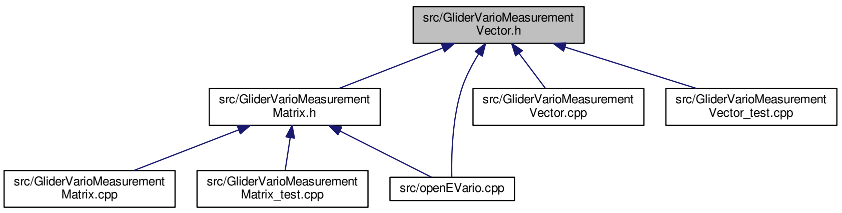
7.10 src/GliderVarioMeasurementVector.h File Reference

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

Include dependency graph for GliderVarioMeasurementVector.h:



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


src/GliderVarioMeasurement
Vector.h
src/GliderVarioMeasurement
Matrix.h
src/GliderVarioMeasurement
Vector.cpp
src/GliderVarioMeasurement
Vector_test.cpp
src/GliderVarioMeasurement
Matrix.cpp
src/GliderVarioMeasurement
Matrix_test.cpp
src/openEVario.cpp

Classes

- class openEV::GliderVarioMeasurementVector

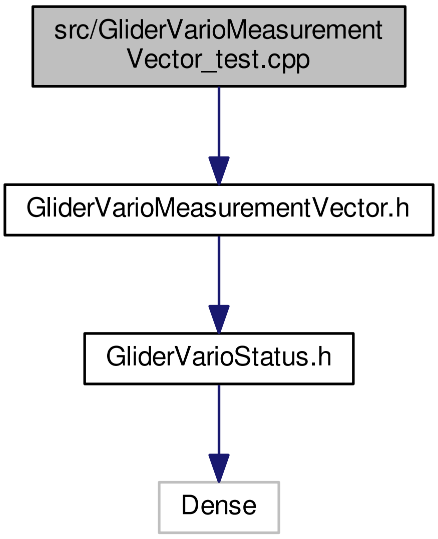
Namespaces

- openEV

7.11 src/GliderVarioMeasurementVector_test.cpp File Reference

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

Include dependency graph for GliderVarioMeasurementVector_test.cpp:



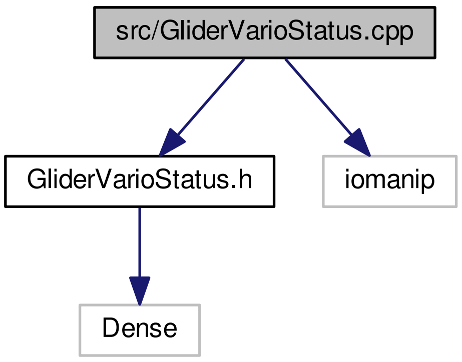
Namespaces

- 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

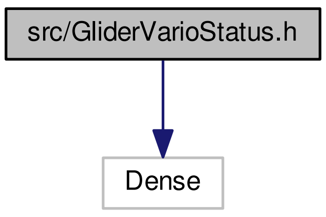
7.12.1.1 std::ostream& operator<< (std::ostream & *o*, openEV::GliderVarioStatus & *s*)

Definition at line 140 of file GliderVarioStatus.cpp.

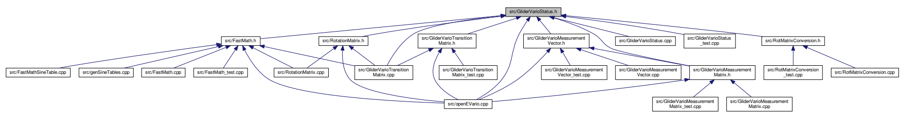
7.13 src/GliderVarioStatus.h File Reference

`#include <Dense>`

Include dependency graph for GliderVarioStatus.h:



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


src/GliderVarioStatus.h
src/FastMath.h
src/RotationMatrix.h
src/GliderVarioTransition Matrix.h
src/GliderVarioMeasurement Vector.h
src/GliderVarioStatus.cpp
src/GliderVarioStatus _test.cpp
src/RotMatrixConversion.h
src/FastMathSineTable.cpp
src/genSineTables.cpp
src/FastMath.cpp
src/FastMath_test.cpp
src/RotationMatrix.cpp
src/GliderVarioTransition Matrix.cpp
src/GliderVarioTransition Matrix_test.cpp
src/GliderVarioMeasurement Vector_test.cpp
src/GliderVarioMeasurement Vector.cpp
src/GliderVarioMeasurement Matrix.h
src/RotMatrixConversion _test.cpp
src/RotMatrixConversion.cpp
src/openEVario.cpp
src/GliderVarioMeasurement Matrix_test.cpp
src/GliderVarioMeasurement Matrix.cpp

Classes

- class openEV::GliderVarioStatus

    GliderVarioStatus manages the Kalman filter state x.

Namespaces

- openEV

Typedefs

- typedef float openEV::FloatType
- typedef Eigen::Matrix< FloatType, 3, 1 > openEV::Vector3DType
- typedef Eigen::Matrix< FloatType, 3, 3 > openEV::RotationMatrix3DType

Functions

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

7.13.1 Function Documentation

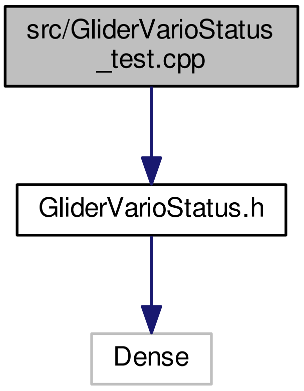
7.13.1.1 std::ostream& operator<< (std::ostream & *o*, openEV::GliderVarioStatus & *s*)

Definition at line 140 of file GliderVarioStatus.cpp.

7.14 src/GliderVarioStatus_test.cpp File Reference

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

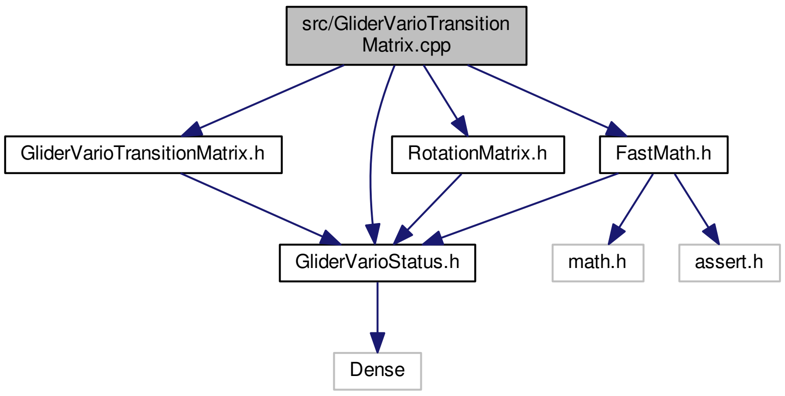
Include dependency graph for GliderVarioStatus_test.cpp:



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:


src/GliderVarioTransitionMatrix.cpp
GliderVarioTransitionMatrix.h
RotationMatrix.h
FastMath.h
GliderVarioStatus.h
math.h
assert.h
Dense

Namespaces

- openEV

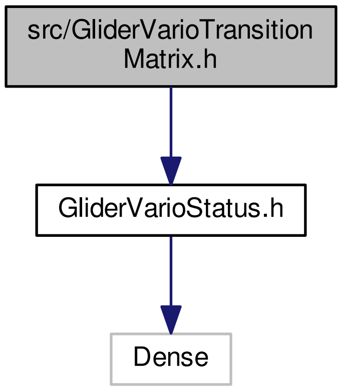
Variables

- FloatType constexpr openEV::lenLatitude = 111132.0

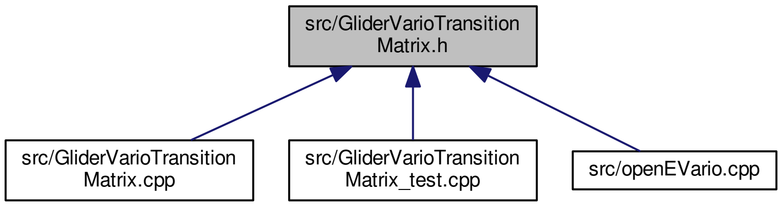
7.16 src/GliderVarioTransitionMatrix.h File Reference

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

Include dependency graph for GliderVarioTransitionMatrix.h:



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


src/GliderVarioTransition Matrix.h
src/GliderVarioTransition Matrix.cpp
src/GliderVarioTransition Matrix_test.cpp
src/openEVario.cpp

Classes

- class openEV::GliderVarioTransitionMatrix

Namespaces

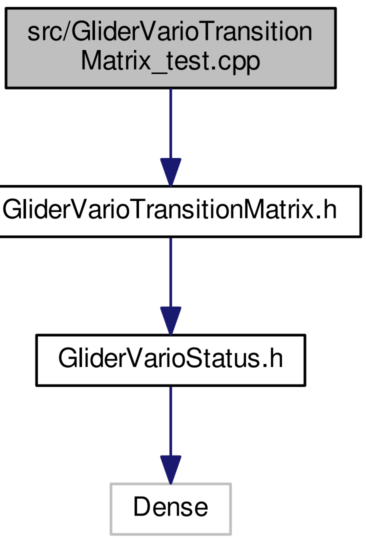
- openEV

Variables

- static FloatType constexpr openEV::GRAVITY = 9.81

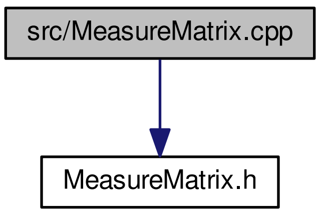
7.17 src/GliderVarioTransitionMatrix_test.cpp File Reference

`#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:

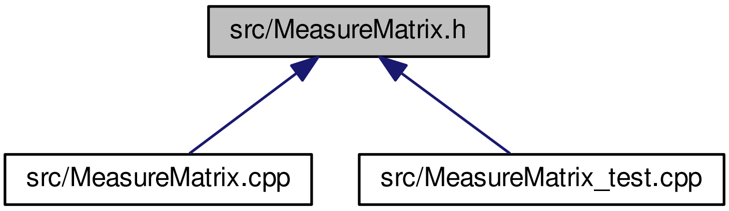


Namespaces

- openEV

7.19 src/MeasureMatrix.h File Reference

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



Classes

- class openEV::MeasureMatrix

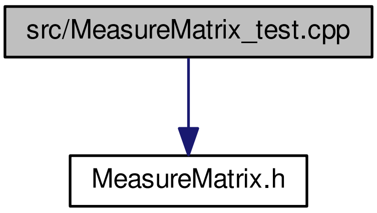
Namespaces

- openEV

7.20 src/MeasureMatrix_test.cpp File Reference

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

Include dependency graph for MeasureMatrix_test.cpp:



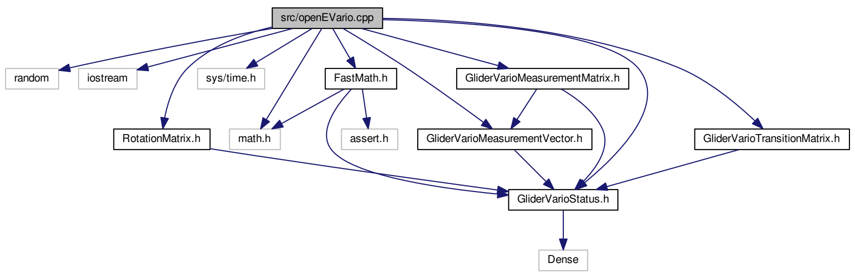
Namespaces

- 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 intialization. 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 intialization. Demonization if required. Entry into the main processing loop.

Parameters

argc	
argv	

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.

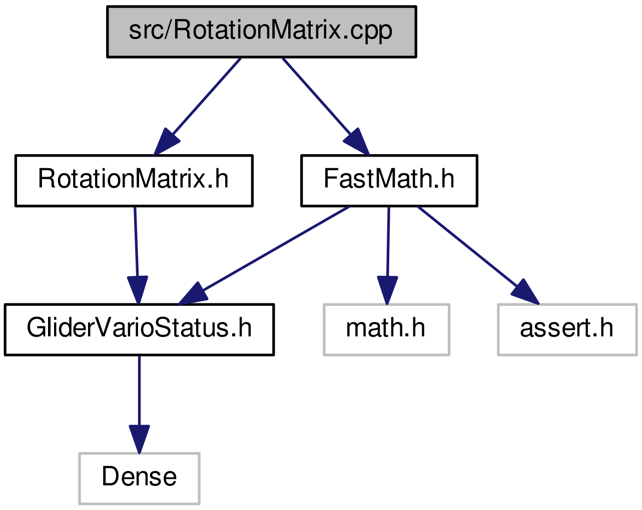
7.21.2.2 FloatType x = 0

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:



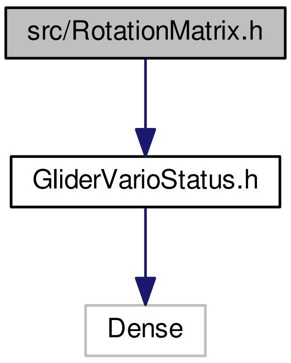
Namespaces

- openEV

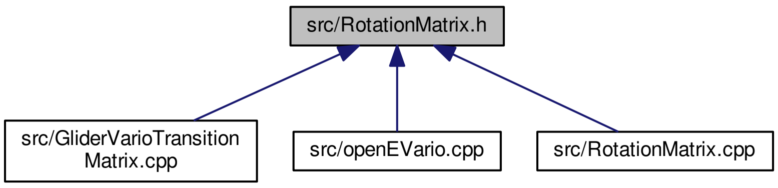
7.23 src/RotationMatrix.h File Reference

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

Include dependency graph for RotationMatrix.h:



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



Classes

- class openEV::RotationMatrix

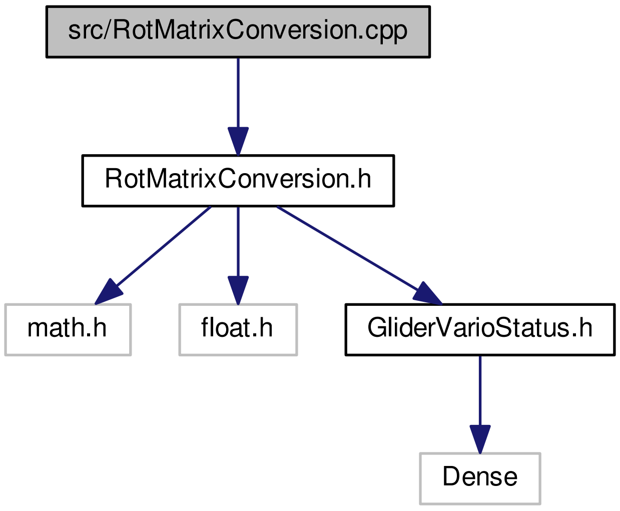
Namespaces

- openEV

7.24 src/RotMatrixConversion.cpp File Reference

```
#include "RotMatrixConversion.h"
```

Include dependency graph for RotMatrixConversion.cpp:



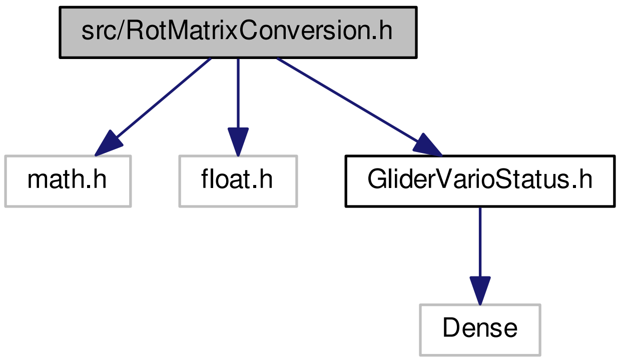
Namespaces

- openEV

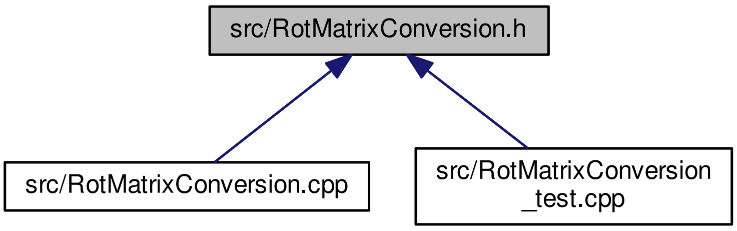
7.25 src/RotMatrixConversion.h File Reference

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

Include dependency graph for RotMatrixConversion.h:



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


src/RotMatrixConversion.h
src/RotMatrixConversion.cpp
src/RotMatrixConversion_test.cpp

Classes

- class openEV::RotMatrixConversion

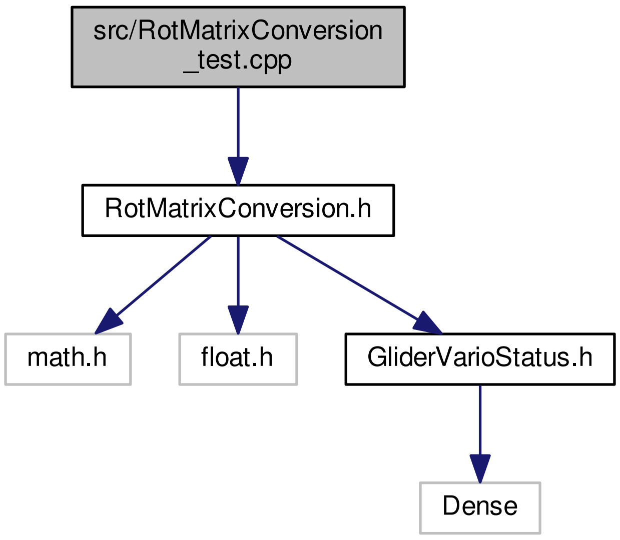
Namespaces

- openEV

7.26 src/RotMatrixConversion_test.cpp File Reference

```
#include "RotMatrixConversion.h"
```

Include dependency graph for RotMatrixConversion_test.cpp:



Namespaces

- openEV

Index