InvenSense Inc.
1197 Borregas Ave., Sunnyvale, CA 94089 U.S.A.
Tel: +1 (408) 988-7339 Fax: +1 (408) 988-8104
Website: www.invensense.com

Doc : SW-MA-AND-REL-5.1.4
Doc Rev : 1.0
Date : 11/26/2012

MotionApps v5.1.4 APIs Specification

A printed copy of this document is
NOT UNDER REVISION CONTROL
unless it is dated and stamped in red ink as,
“REVISION CONTROLLED COPY.”

This information furnished by InvenSense is believed to be accurate and reliable. However, no responsibility is assumed by InvenSense for its use, or for any infringements or patents or other rights of third parties that may result from its use. Specifications are subject to change without notice. Certain intellectual property owned by InvenSense and described in this document is patent protected. No license is granted by Implication or otherwise under any patent or patent rights of InvenSense. This is an unpublished work protected under the United States copyright laws. This work contains proprietary and confidential information of InvenSense Inc. Use, disclosure or reproduction without the express written authorization of InvenSense Inc. is prohibited. Trademarks that are registered trademarks are the property of their respective companies.

This publication supersedes and replaces all information previously supplied. InvenSense sensors should not be used or sold for the development, storing, production and utilization of any conventional or mass-destructive weapons or any other weapons or life threatening applications, as well as to be used in any other life critical applications such as medical, transportation, aerospace, nuclear, undersea, power, disaster and crime prevention equipment.

Chapter 1

Purpose and Scope

This document is a guide to all of the functions available in the InvenSense MotionApps Platform Library (MPL), and corresponds with MotionApps Release v5.1.4.

The MPL contains the code for controlling the InvenSense devices, including activating and managing built in motion processing features. All of the source code is in ANSI C and can be compiled in C or C++ environments.

All functions available in the MPL are described in this document, including all parameters involved in the function calls. The functions are divided into modules as follows:

Module	Name	Description
Data Builder	Builds Sensor Data Structures	Builds the sensor structures and calls functions that need to use them.
HAL Outputs	HAL Outputs	Creates and holds information that a Android HAL layer might want.
Message Layer	Message Layer	Holds Messages
ML Math Func	Math Functions	Support Math Functions.
MPL	MPU Start	Handles init, start, and version properties .
Result_Holder	Result Holder	Holds various output results.
Start_Manager	Start Manager	Sends start events.
Storage_Manager	Store Variables	Stores Internal States.

For more information on how to use these functions in a specific application, refer to InvenSense Application Notes.

Chapter 2

About this document

This document is automatically generated from the source files using Doxygen's output format in the LaTeX. Heading, footer, and general document format are customized from the standard header template provided by Doxygen. The document is subdivided in the various sections, each describing the main source Modules composing the MPL and implementing specific features.

Every section starts with a brief description and an overview of the functions composing the module. Each of those functions is also fully documented in the analogous "Function Documentation" section. Clicking on the function prototype will lead to the portion of text full documentating it.

This **MotionApps Functional Specification** is best viewed in a PDF viewer, as it provides text hyperlinks and bookmarks on the left-hand side for ease of browsing. There is an Alphabatical Index of the modules and their functions available at the bottom of this document.

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Module Documentation

4.1 data_builder

Motion Library - Data Builder Constructs and Creates the data for MPL.

Files

- file data_builder.c

  Data Builder.

Functions

- void inv_accel_was_turned_off ()

  This should be called when the accel has been turned off.
- void inv_apply_calibration (struct inv_single_sensor_t ∗sensor, const long ∗bias)

  Takes raw data stored in the sensor, removes bias, and converts it to calibrated data in the body frame.
- inv_error_t inv_build_accel (const long ∗accel, int status, inv_time_t timestamp)

  Record new accel data for use when inv_execute_on_data() is called.
- inv_error_t inv_build_compass (const long ∗compass, int status, inv_time_t timestamp)

  Record new compass data for use when inv_execute_on_data() is called.
- inv_error_t inv_build_gyro (const short ∗gyro, inv_time_t timestamp)

  Record new gyro data and calls inv_execute_on_data() if previous sample has not been processed.

- inv_error_t inv_build_quat (const long ∗quat, int status, inv_time_t timestamp)

  quaternion data
- inv_error_t inv_build_temp (const long temp, inv_time_t timestamp)

  Record new temperature data for use when inv_execute_on_data() is called.
- void inv_compass_was_turned_off ()

  This should be called when the compass has been turned off.
- void inv_disable_compass_soft_iron_matrix (void)

  This subroutine disables the the soft iron transformation process.
- void inv_enable_compass_soft_iron_matrix (void)

  This subroutine enables the the soft iron transformation process.
- inv_error_t inv_execute_on_data (void)

  After at least one of inv_build_gyro(), inv_build_accel(), or inv_build_compass() has been called, this function should be called.
- int inv_get_accel_accuracy (void)

  Returns accuracy of accel.
- void inv_get_accel_bias (long ∗bias, long ∗temp)

  Get Accel Bias.
- int inv_get_accel_on ()

  Helper function stating whether the acceleromter is on or off.
- long inv_get_accel_sensitivity (void)

  Accel sensitivity.
- void inv_get_accel_set (long ∗data, int8_t ∗accuracy, inv_time_t ∗timestamp)

  Gets a whole set of accel data including data, accuracy and timestamp.
- void inv_get_compass_bias (long ∗bias)

  Returns the current bias for the compass.
- int inv_get_compass_on ()

  Helper function stating whether the compass is on or off.
- long inv_get_compass_sensitivity (void)

  Compass sensitivity.
- void inv_get_compass_set (long ∗data, int8_t ∗accuracy, inv_time_t ∗timestamp)

  Gets a whole set of compass data including data, accuracy and timestamp.
- void inv_get_compass_soft_iron_input_data (long ∗data)

  This subroutine gets the fixed point Q30 compass data before the soft iron transformation.
- void inv_get_compass_soft_iron_matrix_d (long ∗matrix)

  Gets the 3x3 compass transform matrix in 32 bit Q30 fixed point format.
- void inv_get_compass_soft_iron_matrix_f (float ∗matrix)

  Gets the 3x3 compass transform matrix in 32 bit floating point format.
- void inv_get_compass_soft_iron_output_data (long ∗data)

This subroutine gets the fixed point Q30 compass data after the soft iron transformation.

- void inv_get_gyro (long ∗gyro)

  Get's latest gyro data.
- int inv_get_gyro_accuracy (void)

  Returns accuracy of gyro.
- void inv_get_gyro_bias (long ∗bias, long ∗temp)

  Get the gyro biases and temperature record from MPL.
- int inv_get_gyro_on ()

  Helper function stating whether the gyro is on or off.
- long inv_get_gyro_sensitivity (void)

  Gyro sensitivity.
- void inv_get_gyro_set (long ∗data, int8_t ∗accuracy, inv_time_t ∗timestamp)

  Gets a whole set of gyro data including data, accuracy and timestamp.
- void inv_get_gyro_set_raw (long ∗data, int8_t ∗accuracy, inv_time_t ∗timestamp)

  Gets a whole set of gyro raw data including data, accuracy and timestamp.
- inv_time_t inv_get_last_timestamp ()

  Get last timestamp across all 3 sensors that are on.
- int inv_get_mag_accuracy (void)

  Returns accuracy of compass.
- void inv_get_temp_set (long ∗data, int ∗accuracy, inv_time_t ∗timestamp)

  Gets a whole set of temperature data including data, accuracy and timestamp.
- void inv_gyro_was_turned_off ()

  This should be called when the gyro has been turned off.
- inv_error_t inv_init_data_builder (void)

  Initialize the data builder.
- void inv_quaternion_sensor_was_turned_off (void)

  This should be called when the quaternion data from the DMP has been turned off.
- inv_error_t inv_register_data_cb (inv_error_t(∗func)(struct inv_sensor_cal_t ∗data), int priority, int sensor_type)

  Registers to receive a callback when there is new sensor data.
- void inv_reset_compass_soft_iron_matrix (void)

  This subroutine resets the the soft iron transformation to unity matrix and disable the soft iron transformation process by default.
- void inv_set_accel_accuracy (int accuracy)

  Sets the accel accuracy.
- void inv_set_accel_bandwidth (int bandwidth_hz)

  Set Accel Bandwidth in Hz.
- void inv_set_accel_bias (const long ∗bias, int accuracy)

Sets the accel bias.

- void inv_set_accel_bias_mask (const long ∗bias, int accuracy, int mask)

  Sets the accel bias with control over which axis.
- void inv_set_accel_orientation_and_scale (int orientation, long sensitivity)

  Sets the orientation and sensitivity of the gyro data.
- void inv_set_accel_sample_rate (long sample_rate_us)

  Set Accel Sample rate in micro seconds.
- void inv_set_compass_bandwidth (int bandwidth_hz)

  Set Compass Bandwidth in Hz.
- void inv_set_compass_disturbance (int dist)

  Set the state of a compass disturbance.
- void inv_set_compass_orientation_and_scale (int orientation, long sensitivity)

  Sets the Orientation and Sensitivity of the gyro data.
- void inv_set_compass_sample_rate (long sample_rate_us)

  Set Compass Sample rate in micro seconds.
- void inv_set_compass_soft_iron_input_data (const long ∗data)

  This subroutine sets the compass raw data for the soft iron transformation.
- void inv_set_compass_soft_iron_matrix_d (long ∗matrix)

  Sets the 3x3 compass transform matrix in 32 bit Q30 fixed point format.
- void inv_set_compass_soft_iron_matrix_f (float ∗matrix)

  Sets the 3x3 compass transform matrix in 32 bit floating point format.
- void inv_set_gyro_bandwidth (int bandwidth_hz)

  Set Gyro Bandwidth in Hz.
- void inv_set_gyro_bias (const long ∗bias, int accuracy)

  Sets the gyro bias.
- void inv_set_gyro_orientation_and_scale (int orientation, long sensitivity)

  Sets the Orientation and Sensitivity of the gyro data.
- void inv_set_gyro_sample_rate (long sample_rate_us)

  Set Gyro Sample rate in micro seconds.
- void inv_set_quat_sample_rate (long sample_rate_us)

  Set Quat Sample rate in micro seconds.
- void inv_temperature_was_turned_off ()

  This should be called when the temperature sensor has been turned off.
- inv_error_t inv_unregister_data_cb (inv_error_t(∗func)(struct inv_sensor_cal_t ∗data))

  Unregisters the callback that happens when new sensor data is received.
- void set_sensor_orientation_and_scale (struct inv_single_sensor_t ∗sensor, int orientation, long sensitivity)

  Sets orientation and sensitivity field for a sensor.

4.1.1 Detailed Description

Motion Library - Data Builder Constructs and Creates the data for MPL.

4.1.2 Function Documentation

4.1.2.1 void inv_accel_was_turned_off ( )

This should be called when the accel has been turned off.

This is so that we will know if the data is contiguous.

4.1.2.2 void inv_apply_calibration (struct inv_single_sensor_t * *sensor*, const long * *bias*)

Takes raw data stored in the sensor, removes bias, and converts it to calibrated data in the body frame.

Also store raw data for body frame.

Parameters

`in,out`	*sensor*	structure to modify
`in`	*bias*	bias in the mounting frame, in hardware units scaled by 2^16. Length 3.

4.1.2.3 inv_error_t inv_build_accel (const long * *accel*, int *status*, inv_time_t *timestamp*)

Record new accel data for use when inv_execute_on_data() is called.

Parameters

`in`	*accel*	accel data. Length 3. Calibrated data is in m/s^2 scaled by 2^16 in body frame. Raw data is in device units in chip mounting frame.
`in`	*status*	Lower 2 bits are the accuracy, with 0 being inaccurate, and 3 being most accurate. The upper bit INV_CALIBRATED, is set if the data was calibrated outside MPL and it is not set if the data being passed is raw. Raw data should be in device units, typically in a 16-bit range.
`in`	*timestamp*	Monotonic time stamp, for Android it's in nanoseconds.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.1.2.4 inv_error_t inv_build_compass (const long ∗ *compass,* int *status,* inv_time_t *timestamp*)

Record new compass data for use when inv_execute_on_data() is called.

Parameters

in	*compass*	Compass data, if it was calibrated outside MPL, the units are uT scaled by 2^16. Length 3.
in	*status*	Lower 2 bits are the accuracy, with 0 being inaccurate, and 3 being most accurate. The upper bit INV_CALIBRATED, is set if the data was calibrated outside MPL and it is not set if the data being passed is raw. Raw data should be in device units, typically in a 16-bit range.
in	*timestamp*	Monotonic time stamp, for Android it's in nanoseconds.
out	*executed*	Set to 1 if data processing was done.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.1.2.5 inv_error_t inv_build_gyro (const short ∗ *gyro,* inv_time_t *timestamp*)

Record new gyro data and calls inv_execute_on_data() if previous sample has not been processed.

Parameters

in	*gyro*	Data is in device units. Length 3.
in	*timestamp*	Monotonic time stamp, for Android it's in nanoseconds.
out	*executed*	Set to 1 if data processing was done.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.1.2.6 inv_error_t inv_build_quat (const long * *quat*, int *status*, inv_time_t *timestamp*)

quaternion data

Parameters

in	*quat*	Quaternion data. 2^30 = 1.0 or 2^14=1 for 16-bit data. - Real part first. Length 4.
in	*status*	number of axis, 16-bit or 32-bit
in	*timestamp*	
in	*timestamp*	Monotonic time stamp; for Android it's in nanoseconds.
out	*executed*	Set to 1 if data processing was done.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.1.2.7 inv_error_t inv_build_temp (const long *temp*, inv_time_t *timestamp*)

Record new temperature data for use when inv_execute_on_data() is called.

Parameters

in	*temp*	Temperature data in q16 format.
in	*timestamp*	Monotonic time stamp; for Android it's in nanoseconds.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.1.2.8 void inv_compass_was_turned_off ( )

This should be called when the compass has been turned off.

This is so that we will know if the data is contiguous.

4.1.2.9 inv_error_t inv_execute_on_data (void)

After at least one of inv_build_gyro(), inv_build_accel(), or inv_build_compass() has been called, this function should be called.

It will process the data it has received and update all the internal states and features that have been turned on.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.1.2.10 int inv_get_accel_accuracy (void)

Returns accuracy of accel.

Returns

Accuracy of accel with 0 being not accurate, and 3 being most accurate.

4.1.2.11 void inv_get_accel_bias (long ∗ *bias*, long ∗ *temp*)

Get Accel Bias.

Parameters

out	*bias*	Accel bias where
out	*temp*	Temperature where 1 C = 2^16

4.1.2.12 int inv_get_accel_on ( )

Helper function stating whether the acceleromter is on or off.

Returns

TRUE if accel if on, 0 if accel if off

4.1.2.13 long inv_get_accel_sensitivity (void)

Accel sensitivity.

Returns

A scale factor to convert device units to g's scaled by 2^16 such that g_s = device_units ∗ sensitivity / 2^30. Typically it works out to be the maximum accel value in g's ∗ 2^15.

4.1.2.14 void inv_get_accel_set (long ∗ *data*, int8_t ∗ *accuracy*, inv_time_t ∗ *timestamp*)

Gets a whole set of accel data including data, accuracy and timestamp.

Parameters

out	*data*	Accel Data where 1g = 2^16
out	*accuracy*	Accuracy 0 being not accurate, and 3 being most accurate.
out	*timestamp*	The timestamp of the data sample.

4.1.2.15 void inv_get_compass_bias (long ∗ *bias*)

Returns the current bias for the compass.

Parameters

out	*bias*	Compass bias in hardware units scaled by 2^16. In mounting frame. Length 3.

4.1.2.16 int inv_get_compass_on ( )

Helper function stating whether the compass is on or off.

Returns

TRUE if compass if on, 0 if compass if off

4.1.2.17 long inv_get_compass_sensitivity (void)

Compass sensitivity.

Returns

A scale factor to convert device units to micro Tesla scaled by 2^16 such that uT = device_units ∗ sensitivity / 2^30. Typically it works out to be the maximum uT ∗ 2^15.

4.1.2.18 void inv_get_compass_set (long * *data*, int8_t * *accuracy*, inv_time_t * *timestamp*)

Gets a whole set of compass data including data, accuracy and timestamp.

Parameters

out	*data*	Compass Data where 1 uT = 2^16
out	*accuracy*	Accuracy 0 being not accurate, and 3 being most accurate.
out	*timestamp*	The timestamp of the data sample.

4.1.2.19 void inv_get_compass_soft_iron_input_data (long * *data*)

This subroutine gets the fixed point Q30 compass data before the soft iron transformation.

Parameters

out	*the*	pointer of the 3x1 vector compass data in MPL format

4.1.2.20 void inv_get_compass_soft_iron_matrix_d (long * *matrix*)

Gets the 3x3 compass transform matrix in 32 bit Q30 fixed point format.

Parameters

out	*the*	pointer of the 3x3 matrix in Q30 format

4.1.2.21 void inv_get_compass_soft_iron_matrix_f (float * *matrix*)

Gets the 3x3 compass transform matrix in 32 bit floating point format.

Parameters

out	*the*	pointer of the 3x3 matrix in floating point format

4.1.2.22 void inv_get_compass_soft_iron_output_data (long * *data*)

This subroutine gets the fixed point Q30 compass data after the soft iron transformation.

Parameters

out	*the*	pointer of the 3x1 vector compass data in MPL format

4.1.2.23 void inv_get_gyro (long ∗ *gyro*)

Get's latest gyro data.

Parameters

out	*gyro*	Gyro Data, Length 3. 1 dps = 2^16.

4.1.2.24 int inv_get_gyro_accuracy (void)

Returns accuracy of gyro.

Returns

Accuracy of gyro with 0 being not accurate, and 3 being most accurate.

4.1.2.25 void inv_get_gyro_bias (long ∗ *bias,* long ∗ *temp*)

Get the gyro biases and temperature record from MPL.

Parameters

in	*bias*	Gyro bias in hardware units scaled by 2^16. In chip mounting frame. Length 3.
in	*temp*	Tempearature in degrees C.

4.1.2.26 int inv_get_gyro_on ( )

Helper function stating whether the gyro is on or off.

Returns

TRUE if gyro if on, 0 if gyro if off

4.1.2.27 long inv_get_gyro_sensitivity (void)

Gyro sensitivity.

Returns

A scale factor to convert device units to degrees per second scaled by 2^16 such that degrees_per_second = device_units ∗ sensitivity / 2^30. Typically it works out to be the maximum rate ∗ 2^15.

4.1.2.28 void inv_get_gyro_set (long ∗ *data*, int8_t ∗ *accuracy*, inv_time_t ∗ *timestamp*)

Gets a whole set of gyro data including data, accuracy and timestamp.

Parameters

out	*data*	Gyro Data where 1 dps = 2^16
out	*accuracy*	Accuracy 0 being not accurate, and 3 being most accurate.
out	*timestamp*	The timestamp of the data sample.

4.1.2.29 void inv_get_gyro_set_raw (long ∗ *data*, int8_t ∗ *accuracy*, inv_time_t ∗ *timestamp*)

Gets a whole set of gyro raw data including data, accuracy and timestamp.

Parameters

out	*data*	Gyro Data where 1 dps = 2^16
out	*accuracy*	Accuracy 0 being not accurate, and 3 being most accurate.
out	*timestamp*	The timestamp of the data sample.

4.1.2.30 inv_time_t inv_get_last_timestamp ( )

Get last timestamp across all 3 sensors that are on.

This find out which timestamp has the largest value for sensors that are on.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.1.2.31 int inv_get_mag_accuracy (void)

Returns accuracy of compass.

Returns

Accuracy of compass with 0 being not accurate, and 3 being most accurate.

4.1.2.32 void inv_get_temp_set (long ∗ *data,* int ∗ *accuracy,* inv_time_t ∗ *timestamp*)

Gets a whole set of temperature data including data, accuracy and timestamp.

Parameters

out	*data*	Temperature data where 1 degree C = 2^16
out	*accuracy*	0 to 3, where 3 is most accurate.
out	*timestamp*	The timestamp of the data sample.

4.1.2.33 void inv_gyro_was_turned_off ( )

This should be called when the gyro has been turned off.

This is so that we will know if the data is contiguous.

4.1.2.34 void inv_quaternion_sensor_was_turned_off (void)

This should be called when the quaternion data from the DMP has been turned off.

This is so that we will know if the data is contiguous.

4.1.2.35 void inv_set_accel_accuracy (int *accuracy*)

Sets the accel accuracy.

Parameters

in	*accuracy*	Accuracy rating from 0 to 3, with 3 being most accurate.

4.1.2.36 void inv_set_accel_bandwidth (int *bandwidth_hz*)

Set Accel Bandwidth in Hz.

Parameters

in	*bandwidth_hz*	Gyro bandwidth in Hz

4.1.2.37 void inv_set_accel_bias (const long ∗ *bias*, int *accuracy*)

Sets the accel bias.

Parameters

in	*bias*	Accel bias, length 3. In HW units scaled by 2^16 in body frame
in	*accuracy*	Accuracy rating from 0 to 3, with 3 being most accurate.

4.1.2.38 void inv_set_accel_bias_mask (const long ∗ *bias*, int *accuracy*, int *mask*)

Sets the accel bias with control over which axis.

Parameters

in	*bias*	Accel bias, length 3. In HW units scaled by 2^16 in body frame
in	*accuracy*	Accuracy rating from 0 to 3, with 3 being most accurate.
in	*mask*	Mask to select axis to apply bias set.

4.1.2.39 void inv_set_accel_orientation_and_scale (int *orientation*, long *sensitivity*)

Sets the orientation and sensitivity of the gyro data.

Parameters

in	*orientation*	A scalar defining the transformation from chip mounting to the body frame. The function inv_orientation_matrix_to_scalar() can convert the transformation matrix to this scalar and describes the scalar in further detail.
in	*sensitivity*	A scale factor to convert device units to g's such that g's = device_units ∗ sensitivity / 2^30. Typically it works out to be the maximum g_value ∗ 2^15.

4.1.2.40 void inv_set_accel_sample_rate (long *sample_rate_us*)

Set Accel Sample rate in micro seconds.

Parameters

in	*sample_rate_us*	Set Accel Sample rate in us

4.1.2.41 void inv_set_compass_bandwidth (int *bandwidth_hz*)

Set Compass Bandwidth in Hz.

Parameters

in	*bandwidth_hz*	Gyro bandwidth in Hz

4.1.2.42 void inv_set_compass_disturbance (int *dist*)

Set the state of a compass disturbance.

Parameters

in	*dist*	1=disturbance, 0=no disturbance

4.1.2.43 void inv_set_compass_orientation_and_scale (int *orientation,* long *sensitivity*)

Sets the Orientation and Sensitivity of the gyro data.

Parameters

in	*orientation*	A scalar defining the transformation from chip mounting to the body frame. The function inv_orientation_matrix_to_scalar() can convert the transformation matrix to this scalar and describes the scalar in further detail.
in	*sensitivity*	A scale factor to convert device units to uT such that uT = device_units ∗ sensitivity / 2^30. Typically it works out to be the maximum uT_value ∗ 2^15.

4.1.2.44 void inv_set_compass_sample_rate (long *sample_rate_us*)

Set Compass Sample rate in micro seconds.

Parameters

in	*sample_-rate_us*	Set Gyro Sample rate in micro seconds.

4.1.2.45 void inv_set_compass_soft_iron_input_data (const long * *data*)

This subroutine sets the compass raw data for the soft iron transformation.

Parameters

int]	the pointer of the 3x1 vector compass raw data in MPL format

4.1.2.46 void inv_set_compass_soft_iron_matrix_d (long * *matrix*)

Sets the 3x3 compass transform matrix in 32 bit Q30 fixed point format.

Parameters

in	*the*	pointer of the 3x3 matrix in Q30 format

4.1.2.47 void inv_set_compass_soft_iron_matrix_f (float * *matrix*)

Sets the 3x3 compass transform matrix in 32 bit floating point format.

Parameters

in	*the*	pointer of the 3x3 matrix in floating point format

4.1.2.48 void inv_set_gyro_bandwidth (int *bandwidth_hz*)

Set Gyro Bandwidth in Hz.

Parameters

in	*bandwidth_-hz*	Gyro bandwidth in Hz

4.1.2.49 void inv_set_gyro_bias (const long * *bias*, int *accuracy*)

Sets the gyro bias.

Parameters

in	*bias*	Gyro bias in hardware units scaled by 2^16. In chip mounting frame. Length 3.
in	*accuracy*	Accuracy of bias. 0 = least accurate, 3 = most accurate.

4.1.2.50 void inv_set_gyro_orientation_and_scale (int *orientation*, long *sensitivity*)

Sets the Orientation and Sensitivity of the gyro data.

Parameters

in	*orientation*	A scalar defining the transformation from chip mounting to the body frame. The function inv_orientation_matrix_to_scalar() can convert the transformation matrix to this scalar and describes the scalar in further detail.
in	*sensitivity*	A scale factor to convert device units to degrees per second scaled by 2^16 such that degrees_per_second = device_units * sensitivity / 2^30. Typically it works out to be the maximum rate * 2^15.

4.1.2.51 void inv_set_gyro_sample_rate (long *sample_rate_us*)

Set Gyro Sample rate in micro seconds.

Parameters

in	*sample_rate_us*	Set Gyro Sample rate in us

4.1.2.52 void inv_set_quat_sample_rate (long *sample_rate_us*)

Set Quat Sample rate in micro seconds.

Parameters

in	*sample_rate_us*	Set Quat Sample rate in us

4.1.2.53 void inv_temperature_was_turned_off ( )

This should be called when the temperature sensor has been turned off.

This is so that we will know if the data is contiguous.

4.1.2.54 void set_sensor_orientation_and_scale (struct inv_single_sensor_t * *sensor,* int *orientation,* long *sensitivity*)

Sets orientation and sensitivity field for a sensor.

Parameters

`out`	*sensor*	Structure to apply settings to
`in`	*orientation*	Orientation description of how part is mounted.
`in`	*sensitivity*	A Scale factor to convert from hardware units to standard units (dps, uT, g).

4.2 hal_outputs

Motion Library - HAL Outputs Sets up common outputs for HAL.

Files

- file hal_outputs.c

  HAL Outputs.

Functions

- inv_error_t inv_disable_hal_outputs (void)

  Turns off creation and storage of HAL type results.
- inv_error_t inv_enable_hal_outputs (void)

  Turns on creation and storage of HAL type results.
- inv_error_t inv_generate_hal_outputs (struct inv_sensor_cal_t ∗sensor_cal)

  Main callback to generate HAL outputs.
- int inv_get_sensor_type_accelerometer (float ∗values, int8_t ∗accuracy, inv_time_t ∗timestamp)

  Acceleration (m/s^2) in body frame.
- int inv_get_sensor_type_gravity (float ∗values, int8_t ∗accuracy, inv_time_t ∗timestamp)

  Gravity vector (m/s^2) in Body Frame.
- int inv_get_sensor_type_gyroscope (float ∗values, int8_t ∗accuracy, inv_time_t ∗timestamp)

  Gyroscope calibrated data (rad/s) in body frame.
- int inv_get_sensor_type_gyroscope_raw (float ∗values, int8_t ∗accuracy, inv_time_t ∗timestamp)

  Gyroscope raw data (rad/s) in body frame.
- int inv_get_sensor_type_linear_acceleration (float ∗values, int8_t ∗accuracy, inv_time_t ∗timestamp)

  Linear Acceleration (m/s^2) in Body Frame.
- int inv_get_sensor_type_magnetic_field (float ∗values, int8_t ∗accuracy, inv_time_t ∗timestamp)

  Compass data (uT) in body frame.
- int inv_get_sensor_type_orientation (float ∗values, int8_t ∗accuracy, inv_time_t ∗timestamp)

  This corresponds to Sensor.TYPE_ORIENTATION.
- int inv_get_sensor_type_rotation_vector (float ∗values, int8_t ∗accuracy, inv_time_t ∗timestamp)

This corresponds to Sensor.TYPE_ROTATION_VECTOR.

- inv_error_t inv_init_hal_outputs (void)

  Initializes hal outputs class.
- inv_error_t inv_start_hal_outputs (void)

  Turns on generation of HAL outputs.
- inv_error_t inv_stop_hal_outputs (void)

  Turns off generation of HAL outputs.

4.2.1 Detailed Description

Motion Library - HAL Outputs Sets up common outputs for HAL.

4.2.2 Function Documentation

4.2.2.1 inv_error_t inv_enable_hal_outputs (void)

Turns on creation and storage of HAL type results.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.2.2.2 inv_error_t inv_generate_hal_outputs (struct inv_sensor_cal_t * *sensor_cal*)

Main callback to generate HAL outputs.

Typically not called by library users.

Parameters

in	*sensor_cal*	Input variable to take sensor data whenever there is new sensor data.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.2.2.3 int inv_get_sensor_type_accelerometer (float * *values*, int8_t * *accuracy*, inv_time_t * *timestamp*)

Acceleration (m/s^2) in body frame.

Parameters

out	*values*	Acceleration in m/s^2 includes gravity. So while not in motion, it should return a vector of magnitude near 9.81 m/s^2
out	*accuracy*	Accuracy of the measurment, 0 is least accurate, while 3 is most accurate.
out	*timestamp*	The timestamp for this sensor. Derived from the timestamp sent to inv_build_accel().

Returns

Returns 1 if the data was updated or 0 if it was not updated.

4.2.2.4 int inv_get_sensor_type_gravity (float ∗ *values,* int8_t ∗ *accuracy,* inv_time_t ∗ *timestamp*)

Gravity vector (m/s^2) in Body Frame.

Parameters

out	*values*	Gravity vector in body frame, length 3, (m/s^2)
out	*accuracy*	Accuracy of the measurment, 0 is least accurate, while 3 is most accurate.
out	*timestamp*	The timestamp for this sensor. Derived from the timestamp sent to inv_build_accel().

Returns

Returns 1 if the data was updated or 0 if it was not updated.

4.2.2.5 int inv_get_sensor_type_gyroscope (float ∗ *values,* int8_t ∗ *accuracy,* inv_time_t ∗ *timestamp*)

Gyroscope calibrated data (rad/s) in body frame.

Parameters

out	*values*	Rotation Rate in rad/sec.
out	*accuracy*	Accuracy of the measurment, 0 is least accurate, while 3 is most accurate.
out	*timestamp*	The timestamp for this sensor. Derived from the timestamp sent to inv_build_gyro().

Returns

Returns 1 if the data was updated or 0 if it was not updated.

4.2.2.6 int inv_get_sensor_type_gyroscope_raw (float ∗ *values,* int8_t ∗ *accuracy,* inv_time_t ∗ *timestamp*)

Gyroscope raw data (rad/s) in body frame.

Parameters

out	*values*	Rotation Rate in rad/sec.
out	*accuracy*	Accuracy of the measurment, 0 is least accurate, while 3 is most accurate.
out	*timestamp*	The timestamp for this sensor. Derived from the timestamp sent to inv_build_gyro().

Returns

Returns 1 if the data was updated or 0 if it was not updated.

4.2.2.7 int inv_get_sensor_type_linear_acceleration (float ∗ *values,* int8_t ∗ *accuracy,* inv_time_t ∗ *timestamp*)

Linear Acceleration (m/s^2) in Body Frame.

Parameters

out	*values*	Linear Acceleration in body frame, length 3, (m/s^2). May show accel biases while at rest.
out	*accuracy*	Accuracy of the measurment, 0 is least accurate, while 3 is most accurate.
out	*timestamp*	The timestamp for this sensor. Derived from the timestamp sent to inv_build_accel().

Returns

Returns 1 if the data was updated or 0 if it was not updated.

4.2.2.8 int inv_get_sensor_type_magnetic_field (float ∗ *values,* int8_t ∗ *accuracy,* inv_time_t ∗ *timestamp*)

Compass data (uT) in body frame.

Parameters

out	*values*	Compass data in (uT), length 3. May be calibrated by having biases removed and sensitivity adjusted
out	*accuracy*	Accuracy 0 to 3, 3 = most accurate
out	*timestamp*	Timestamp. In (ns) for Android.

Returns

Returns 1 if the data was updated or 0 if it was not updated.

4.2.2.9 int inv_get_sensor_type_orientation (float * *values*, int8_t * *accuracy*, inv_time_t * *timestamp*)

This corresponds to Sensor.TYPE_ORIENTATION.

All values are angles in degrees.

Parameters

out	*values*	Length 3, Degrees. • values[0]: Azimuth, angle between the magnetic north direction and the y-axis, around the z-axis (0 to 359). 0=North, 90=East, 180=South, 270=West • values[1]: Pitch, rotation around x-axis (-180 to 180), with positive values when the z-axis moves toward the y-axis. • values[2]: Roll, rotation around y-axis (-90 to 90), with positive values when the x-axis moves toward the z-axis.

Note

This definition is different from yaw, pitch and roll used in aviation where the X axis is along the long side of the plane (tail to nose). Note: This sensor type exists for legacy reasons, please use getRotationMatrix() in conjunction with remapCoordinateSystem() and getOrientation() to compute these values instead. Important note: For historical reasons the roll angle is positive in the clockwise direction (mathematically speaking, it should be positive in the counter-clockwise direction).

Parameters

out	*accuracy*	Accuracy of the measurment, 0 is least accurate, while 3 is most accurate.
out	*timestamp*	The timestamp for this sensor.

Returns

Returns 1 if the data was updated or 0 if it was not updated.

4.2.2.10 int inv_get_sensor_type_rotation_vector (float ∗ *values,* int8_t ∗ *accuracy,* inv_time_t ∗ *timestamp*)

This corresponds to Sensor.TYPE_ROTATION_VECTOR.

The rotation vector represents the orientation of the device as a combination of an angle and an axis, in which the device has rotated through an angle θ around an axis {x, y, z}.

The three elements of the rotation vector are {x∗sin(θ/2), y∗sin(θ/2), z∗sin(θ/2)}, such that the magnitude of the rotation vector is equal to sin(θ/2), and the direction of the rotation vector is equal to the direction of the axis of rotation.

The three elements of the rotation vector are equal to the last three components of a unit quaternion {x∗sin(θ/2), y∗sin(θ/2), z∗sin(θ/2)>. The 4th element is cos(θ/2).

Elements of the rotation vector are unitless. The x,y and z axis are defined in the same way as the acceleration sensor. The reference coordinate system is defined as a direct orthonormal basis, where:

-X is defined as the vector product Y.Z (It is tangential to the ground at the device's current location and roughly points East). -Y is tangential to the ground at the device's current location and points towards the magnetic North Pole. -Z points towards the sky and is perpendicular to the ground.

Parameters

out	*values*	Length 4.
out	*accuracy*	Accuracy 0 to 3, 3 = most accurate
out	*timestamp*	Timestamp. In (ns) for Android.

Returns

Returns 1 if the data was updated or 0 if it was not updated.

4.2.2.11 inv_error_t inv_init_hal_outputs (void)

Initializes hal outputs class.

This is called automatically by the enable function. It may be called any time the feature is enabled, but is typically not needed to be called by outside callers.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.2.2.12 inv_error_t inv_start_hal_outputs (void)

Turns on generation of HAL outputs.

This should be called after inv_stop_hal_outputs() to turn generation of HAL outputs back on. It is automatically called by inv_enable_hal_outputs().

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.2.2.13 inv_error_t inv_stop_hal_outputs (void)

Turns off generation of HAL outputs.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.3 ml_math_func

Motion Library - Math Functions Common math functions the Motion Library.

Files

- file ml_math_func.c

  Math Functions.

Functions

- float inv_angle_diff (float ang1, float ang2)

  Finds the minimum angle difference ang1-ang2 such that difference is between [-M_-PI,M_PI].
- short inv_big8_to_int16 (const unsigned char ∗big8)

  Converts a big endian byte stream into a 16-bit integer (short)
- long inv_big8_to_int32 (const unsigned char ∗big8)

  Converts a big endian byte stream into a 32-bit long.
- uint32_t inv_checksum (const unsigned char ∗str, int len)

  bernstein hash, derived from public domain source
- void inv_convert_to_body (unsigned short orientation, const long ∗input, long ∗output)

  Uses the scalar orientation value to convert from chip frame to body frame.
- void inv_convert_to_body_with_scale (unsigned short orientation, long sensitivity, const long ∗input, long ∗output)

  Uses the scalar orientation value to convert from chip frame to body frame and apply appropriate scaling.
- void inv_convert_to_chip (unsigned short orientation, const long ∗input, long ∗output)

  Uses the scalar orientation value to convert from body frame to chip frame.
- unsigned long inv_get_gyro_sum_of_sqr (const long ∗gyro)

  The gyro data magnitude squared : (1 degree per second)^2 = 2^6 = 2^GYRO_MA-G_SQR_SHIFT.
- unsigned char ∗ inv_int16_to_big8 (short x, unsigned char ∗big8)

  Converts a 16-bit short to a big endian byte stream.
- unsigned char ∗ inv_int32_to_big8 (long x, unsigned char ∗big8)

  Converts a 32-bit long to a big endian byte stream.
- short inv_little8_to_int16 (const unsigned char ∗little8)

  Converts a little endian byte stream into a 16-bit integer (short)
- unsigned short inv_orientation_matrix_to_scalar (const signed char ∗mtx)

Converts an orientation matrix made up of 0,+1,and -1 to a scalar representation.

- long inv_q29_mult (long a, long b)

  Performs a multiply and shift by 29.
- long inv_q30_mult (long a, long b)

  Performs a multiply and shift by 30.
- void inv_q_add (long ∗q1, long ∗q2, long ∗qSum)

  Performs a fixed point quaternion addition.
- void inv_q_mult (const long ∗q1, const long ∗q2, long ∗qProd)

  Performs a fixed point quaternion multiply.
- void inv_q_norm4 (float ∗q)

  Performs a length 4 vector normalization with a square root.
- void inv_q_rotate (const long ∗q, const long ∗in, long ∗out)

  Rotates a 3-element vector by Rotation defined by Q.
- long inv_q_shift_mult (long a, long b, int shift)

  Performs a multiply and shift by shift.
- void inv_quaternion_to_rotation (const long ∗quat, long ∗rot)

  Converts a quaternion to a rotation matrix.
- void inv_quaternion_to_rotation_vector (const long ∗quat, long ∗rot)

  Converts a quaternion to a rotation vector.
- double inv_vector_norm (const float ∗x)

  find a norm for a vector
- float inv_wrap_angle (float ang)

  Wraps angle from (-M_PI,M_PI].

4.3.1 Detailed Description

Motion Library - Math Functions Common math functions the Motion Library.

4.3.2 Function Documentation

4.3.2.1 float inv_angle_diff (float *ang1*, float *ang2*)

Finds the minimum angle difference ang1-ang2 such that difference is between [-M_-PI,M_PI].

Parameters

in	*ang1*	
in	*ang2*	

Returns

angle difference ang1-ang2

4.3.2.2 void inv_convert_to_body (unsigned short *orientation*, const long ∗ *input*, long ∗ *output*)

Uses the scalar orientation value to convert from chip frame to body frame.

Parameters

in	*orientation*	A scalar that represent how to go from chip to body frame
in	*input*	Input vector, length 3
out	*output*	Output vector, length 3

4.3.2.3 void inv_convert_to_body_with_scale (unsigned short *orientation*, long *sensitivity*, const long ∗ *input*, long ∗ *output*)

Uses the scalar orientation value to convert from chip frame to body frame and apply appropriate scaling.

Parameters

in	*orientation*	A scalar that represent how to go from chip to body frame
in	*sensitivity*	Sensitivity scale
in	*input*	Input vector, length 3
out	*output*	Output vector, length 3

4.3.2.4 void inv_convert_to_chip (unsigned short *orientation*, const long ∗ *input*, long ∗ *output*)

Uses the scalar orientation value to convert from body frame to chip frame.

Parameters

in	*orientation*	A scalar that represent how to go from chip to body frame
in	*input*	Input vector, length 3
out	*output*	Output vector, length 3

4.3.2.5 unsigned long inv_get_gyro_sum_of_sqr (const long * *gyro*)

The gyro data magnitude squared : (1 degree per second)^2 = 2^6 = 2^GYRO_MAG_SQR_SHIFT.

Parameters

in	*gyro*	Gyro data scaled with 1 dps = 2^16

Returns

the computed magnitude squared output of the gyroscope.

4.3.2.6 unsigned short inv_orientation_matrix_to_scalar (const signed char * *mtx*)

Converts an orientation matrix made up of 0,+1,and -1 to a scalar representation.

Parameters

in	*mtx*	Orientation matrix to convert to a scalar.

Returns

Description of orientation matrix. The lowest 2 bits (0 and 1) represent the column the one is on for the first row, with the bit number 2 being the sign. The next 2 bits (3 and 4) represent the column the one is on for the second row with bit number 5 being the sign. The next 2 bits (6 and 7) represent the column the one is on for the third row with bit number 8 being the sign. In binary the identity matrix would therefor be: 010_001_000 or 0x88 in hex.

4.3.2.7 long inv_q29_mult (long *a*, long *b*)

Performs a multiply and shift by 29.

These are good functions to write in assembly on with devices with small memory where you want to get rid of the long long which some assemblers don't handle well

Parameters

in	*a*	
in	*b*	

Returns

((long long)a∗b)>>29

4.3.2.8 long inv_q30_mult (long *a*, long *b*)

Performs a multiply and shift by 30.

These are good functions to write in assembly on with devices with small memory where you want to get rid of the long long which some assemblers don't handle well

Parameters

in	*a*	
in	*b*	

Returns

((long long)a∗b)>>30

4.3.2.9 void inv_q_add (long ∗ *q1*, long ∗ *q2*, long ∗ *qSum*)

Performs a fixed point quaternion addition.

Parameters

in	*q1*	First Quaternion term, length 4. 1.0 scaled to 2^30
in	*q2*	Second Quaternion term, length 4. 1.0 scaled to 2^30
out	*qSum*	Sum after quaternion summation. Length 4. 1.0 scaled to 2^30.

4.3.2.10 void inv_q_mult (const long ∗ *q1*, const long ∗ *q2*, long ∗ *qProd*)

Performs a fixed point quaternion multiply.

Parameters

in	*q1*	First Quaternion Multicand, length 4. 1.0 scaled to 2^30
in	*q2*	Second Quaternion Multicand, length 4. 1.0 scaled to 2^30
out	*qProd*	Product after quaternion multiply. Length 4. 1.0 scaled to 2^30.

4.3.2.11 void inv_q_norm4 (float * *q*)

Performs a length 4 vector normalization with a square root.

Parameters

in,out	*q*	vector to normalize. Returns [1,0,0,0] is magnitude is zero.

4.3.2.12 long inv_q_shift_mult (long *a,* long *b,* int *shift*)

Performs a multiply and shift by shift.

These are good functions to write in assembly on with devices with small memory where you want to get rid of the long long which some assemblers don't handle well

Parameters

in	*a*	First multicand
in	*b*	Second multicand
in	*shift*	Shift amount after multiplying

Returns

((long long)a*b)<<shift

4.3.2.13 void inv_quaternion_to_rotation (const long * *quat,* long * *rot*)

Converts a quaternion to a rotation matrix.

Parameters

in	*quat*	4-element quaternion in fixed point. One is 2^30.
out	*rot*	Rotation matrix in fixed point. One is 2^30. The First 3 elements of the rotation matrix, represent the first row of the matrix. Rotation matrix multiplied by a 3 element column vector transform a vector from Body to World.

4.3.2.14 void inv_quaternion_to_rotation_vector (const long * *quat,* long * *rot*)

Converts a quaternion to a rotation vector.

A rotation vector is a method to represent a 4-element quaternion vector in 3-elements. To get the quaternion from the 3-elements, The last 3-elements of the quaternion will be the given rotation vector. The first element of the quaternion will be the positive value that will be required to make the magnitude of the quaternion 1.0 or 2^30 in fixed point units.

Parameters

in	*quat*	4-element quaternion in fixed point. One is 2^30.
out	*rot*	Rotation vector in fixed point. One is 2^30.

4.3.2.15 double inv_vector_norm (const float ∗ *x*)

find a norm for a vector

Parameters

in	*a*	vector [3x1]
out	*output*	the norm of the input vector

4.3.2.16 float inv_wrap_angle (float *ang*)

Wraps angle from (-M_PI,M_PI].

Parameters

in	*ang*	Angle in radians to wrap

Returns

Wrapped angle from (-M_PI,M_PI]

4.4 message_layer

Motion Library - Message Layer Holds Low Occurance messages.

Files

- file message_layer.c

  Holds Low Occurance Messages.

Functions

- long inv_get_message_level_0 (int clear)

  Returns Message Flags for Level 0 Messages.
- void inv_set_message (long set, long clear, int level)

  Sets a message.

4.4.1 Detailed Description

Motion Library - Message Layer Holds Low Occurance messages.

4.4.2 Function Documentation

4.4.2.1 long inv_get_message_level_0 (int *clear*)

Returns Message Flags for Level 0 Messages.

Levels are to allow expansion of more messages in the future.

Parameters

in	*clear*	If set, will clear the message. Typically this will be set for one reader, so that you don't get the same message over and over.

Returns

bit field to corresponding message.

4.4.2.2 void inv_set_message (long *set,* long *clear,* int *level*)

Sets a message.

Parameters

in	*set*	The flags to set.
in	*clear*	Before setting anything this will clear these messages, which is useful for mutually exclusive messages such a motion or no motion message.
in	*level*	Level of the messages. It starts at 0, and may increase in the future to allow more messages if the bit storage runs out.

4.5 mpl

Motion Library - Start Point Initializes MPL.

Files

- file mpl.c

  MPL start point.
- file quaternion_supervisor.c

  Performs the quaternion fusion.

Functions

- inv_error_t inv_disable_quaternion (void)

  Disables generating the gyro and accel quaternion.
- inv_error_t inv_enable_quaternion ()

  Turns on quaternion computation.
- inv_error_t inv_get_version (char ∗∗version)

  used to get the MPL version.
- inv_error_t inv_init_mpl (void)

  Initializes the MPL.
- inv_error_t inv_init_quaternion (void)

  Initializes all quaternion data.
- void inv_set_quaternion (long ∗quat)

  Set the quaternion to the given value.
- inv_error_t inv_start_mpl (void)

  Starts the MPL.
- inv_error_t inv_start_quaternion (void)

  Starts gyro and accel quaternion generation.
- inv_error_t inv_stop_quaternion (void)

  Stops gyro and accel quaternion generation.

4.5.1 Detailed Description

Motion Library - Start Point Initializes MPL. Motion Library Example Architecture.

4.5.2 Function Documentation

4.5.2.1 inv_error_t inv_enable_quaternion ( )

Turns on quaternion computation.

This must be called after inv_init_mpl() and before inv_start_mpl(). It is typically only called once per session. inv_start_quaterion() and inv_stop_quaternion() are used to start and stop this feature. This feature is started automatically and inv_start_-quaterion() would only need to be called after turning this feature off with inv_stop_-quaternion().

Returns

INV_SUCCESS=0 on success, a non-zero error code otherwise.

4.5.2.2 inv_error_t inv_get_version (char ** *version*)

used to get the MPL version.

Parameters

version	a string where the MPL version gets stored.

Returns

INV_SUCCESS if successful or a non-zero error code otherwise.

4.5.2.3 inv_error_t inv_init_mpl (void)

Initializes the MPL.

Should be called first and once

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.5.2.4 inv_error_t inv_init_quaternion (void)

Initializes all quaternion data.

This is called automatically by the enable function. It may be called any time the feature is enabled, but is typically not needed to be called by outside callers.

Returns

INV_SUCCESS=0 on success, a non-zero error code otherwise.

4.5.2.5 void inv_set_quaternion (long * *quat*)

Set the quaternion to the given value.

Parameters

in	*quat*	What to set quaternion to. Fixed point scaled by 2^30, - Length 4.

4.5.2.6 inv_error_t inv_start_mpl (void)

Starts the MPL.

Typically called after inv_init_mpl() or after a inv_stop_mpl() to start the MPL back up an running.

Returns

INV_SUCCESS if successful or a non-zero error code otherwise.

4.5.2.7 inv_error_t inv_start_quaternion (void)

Starts gyro and accel quaternion generation.

Automatically called by inv_enable_quaterion() and therefor would only need to be called after inv_stop_quaternion().

Returns

INV_SUCCESS=0 on success, a non-zero error code otherwise.

4.5.2.8 inv_error_t inv_stop_quaternion (void)

Stops gyro and accel quaternion generation.

Call inv_start_quaternion() to turn this back on after the stop command.

Returns

INV_SUCCESS=0 on success, a non-zero error code otherwise.

4.6 results_holder

Motion Library - Results Holder Holds the data for MPL.

Files

- file results_holder.c

  Results Holder for HAL.

Functions

- inv_error_t inv_enable_results_holder ()

  Turns on storage of results.
- inv_error_t inv_generate_results (struct inv_sensor_cal_t ∗sensor_cal)

  Callback that gets called everytime there is new data.
- inv_error_t inv_get_6axis_quaternion (long ∗data)

  Returns a quaternion based only on gyro and accel.
- int inv_get_acc_state ()

  Gets the accel state set by inv_set_acc_state()
- inv_error_t inv_get_accel (long ∗data)

  Returns 3-element vector of accelerometer data in body frame.
- inv_error_t inv_get_accel_float (float ∗data)

  Returns 3-element vector of accelerometer float data.
- void inv_get_compass_bias_error (long ∗bias_error)

  Get's compass bias error.
- int inv_get_compass_state ()

  Get's the compass state.
- inv_error_t inv_get_gravity (long ∗data)

  Gets gravity vector.
- inv_error_t inv_get_gyro_float (float ∗data)

  Returns 3-element vector of gyro float data.
- float inv_get_heading_confidence_interval (void)

  Get 9 axis 95% heading confidence interval for quaternion.
- int inv_get_large_mag_field ()

  Returns non-zero if there is a large magnetic field.
- inv_error_t inv_get_linear_accel (long ∗data)

  Returns 3-element vector of accelerometer data in body frame with gravity removed.
- inv_error_t inv_get_linear_accel_float (float ∗data)

  Returns 3-element vector of linear accel float data.

- void inv_get_local_field (long ∗data)

  Gets the local earth's magnetic field.
- void inv_get_mag_scale (long ∗data)

  Gets the compass sensitivity.
- int inv_get_motion_state (unsigned int ∗cntr)

  Returns the motion state.
- inv_error_t inv_get_quaternion (long ∗data)

  Returns a quaternion.
- inv_error_t inv_get_quaternion_float (float ∗data)

  Returns a quaternion.
- void inv_get_quaternion_set (long ∗data, int ∗accuracy, inv_time_t ∗timestamp)

  Returns a quaternion with accuracy and timestamp.
- int inv_got_accel_bias ()

  Sets state of if we know the accel bias.
- int inv_got_compass_bias ()

  Sets state of if we know the compass bias.
- inv_error_t inv_init_results_holder (void)

  Initializes results holder.
- void inv_set_acc_state (int state)

  Sets the accel state.
- void inv_set_accel_bias_found (int state)

  Sets whether we know the accel bias.
- void inv_set_compass_bias_error (const long ∗bias_error)

  Set compass bias error.
- void inv_set_compass_bias_found (int state)

  Sets whether we know the compass bias.
- void inv_set_compass_state (int state)

  Sets the compass state.
- void inv_set_heading_confidence_interval (float ci)

  Set 9 axis 95% heading confidence interval for quaternion.
- void inv_set_large_mag_field (int state)

  Set to non-zero if there as a large magnetic field.
- void inv_set_local_field (const long ∗data)

  Sets the local earth's magnetic field.
- void inv_set_mag_scale (const long ∗data)

  Sets the compass sensitivity.
- void inv_set_motion_state (unsigned char state)

  Sets the motion state.
- inv_error_t inv_start_results_holder (void)

  Function to turn on this module.

4.6.1 Detailed Description

Motion Library - Results Holder Holds the data for MPL.

4.6.2 Function Documentation

4.6.2.1 inv_error_t inv_generate_results (struct inv_sensor_cal_t * *sensor_cal*)

Callback that gets called everytime there is new data.

It is registered by inv_start_results_holder().

Parameters

in	*sensor_cal*	New sensor data to process.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.6.2.2 inv_error_t inv_get_6axis_quaternion (long * *data*)

Returns a quaternion based only on gyro and accel.

Parameters

out	*data*	6-axis gyro and accel quaternion scaled such that 1.0 = 2^30.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.6.2.3 int inv_get_acc_state ( )

Gets the accel state set by inv_set_acc_state()

Returns

accel state.

4.6.2.4 inv_error_t inv_get_accel (long ∗ *data*)

Returns 3-element vector of accelerometer data in body frame.

Parameters

out	*data*	3-element vector of accelerometer data in body frame

Returns

INV_SUCCESS if successful INV_ERROR_INVALID_PARAMETER if invalid input pointer

4.6.2.5 inv_error_t inv_get_accel_float (float ∗ *data*)

Returns 3-element vector of accelerometer float data.

Parameters

out	*data*	3-element vector of accelerometer float data

Returns

INV_SUCCESS if successful INV_ERROR_INVALID_PARAMETER if invalid input pointer

4.6.2.6 void inv_get_compass_bias_error (long ∗ *bias_error*)

Get's compass bias error.

See inv_set_compass_bias_error() for setting.

Parameters

out	*bias_error*	Accuracy as to how well the compass bias is known. It is the error squared.

4.6.2.7 int inv_get_compass_state ( )

Get's the compass state.

Returns

the compass state that was set with inv_set_compass_state()

4.6.2.8 inv_error_t inv_get_gravity (long ∗ *data*)

Gets gravity vector.

Parameters

out	*data*	gravity vector in body frame scaled such that 1.0 = 2^30.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.6.2.9 inv_error_t inv_get_gyro_float (float ∗ *data*)

Returns 3-element vector of gyro float data.

Parameters

out	*data*	3-element vector of gyro float data

Returns

INV_SUCCESS if successful INV_ERROR_INVALID_PARAMETER if invalid input pointer

4.6.2.10 float inv_get_heading_confidence_interval (void)

Get 9 axis 95% heading confidence interval for quaternion.

Returns

Confidence interval in radians.

4.6.2.11 int inv_get_large_mag_field ( )

Returns non-zero if there is a large magnetic field.

See inv_set_large_mag_field() for setting this variable.

Returns

Returns non-zero if there is a large magnetic field.

4.6.2.12 inv_error_t inv_get_linear_accel (long * *data*)

Returns 3-element vector of accelerometer data in body frame with gravity removed.

Parameters

out	*data*	3-element vector of accelerometer data in body frame with gravity removed

Returns

INV_SUCCESS if successful INV_ERROR_INVALID_PARAMETER if invalid input pointer

4.6.2.13 inv_error_t inv_get_linear_accel_float (float * *data*)

Returns 3-element vector of linear accel float data.

Parameters

out	*data*	3-element vector of linear aceel float data

Returns

INV_SUCCESS if successful INV_ERROR_INVALID_PARAMETER if invalid input pointer

4.6.2.14 void inv_get_local_field (long * *data*)

Gets the local earth's magnetic field.

Parameters

out	*data*	Local earth's magnetic field in uT scaled by 2^16. Length = 3. Y typically points north, Z typically points down in northern hemisphere and up in southern hemisphere.

4.6.2.15 void inv_get_mag_scale (long ∗ *data*)

Gets the compass sensitivity.

Parameters

out	*data*	Length 3, sensitivity for each compass axis scaled such that 1.0 = 2^30.

4.6.2.16 int inv_get_motion_state (unsigned int ∗ *cntr*)

Returns the motion state.

Parameters

out	*cntr*	Number of previous times a no motion event has occured in a row.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.6.2.17 inv_error_t inv_get_quaternion (long ∗ *data*)

Returns a quaternion.

Parameters

out	*data*	9-axis quaternion scaled such that 1.0 = 2^30.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.6.2.18 inv_error_t inv_get_quaternion_float (float ∗ *data*)

Returns a quaternion.

Parameters

out	*data*	9-axis quaternion.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.6.2.19 void inv_get_quaternion_set (long * *data*, int * *accuracy*, inv_time_t * *timestamp*)

Returns a quaternion with accuracy and timestamp.

Parameters

out	*data*	9-axis quaternion scaled such that 1.0 = 2^30.
out	*accuracy*	Accuracy of quaternion, 0-3, where 3 is most accurate.
out	*timestamp*	Timestamp of this quaternion in nanoseconds

4.6.2.20 int inv_got_accel_bias ( )

Sets state of if we know the accel bias.

Returns

return 1 if we know the accel bias, 0 if not. it is set with inv_set_accel_bias_found()

4.6.2.21 int inv_got_compass_bias ( )

Sets state of if we know the compass bias.

Returns

return 1 if we know the compass bias, 0 if not. it is set with inv_set_compass_bias_found()

4.6.2.22 inv_error_t inv_init_results_holder (void)

Initializes results holder.

This is called automatically by the enable function inv_enable_results_holder(). It may be called any time the feature is enabled, but is typically not needed to be called by outside callers.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.6.2.23 void inv_set_acc_state (int *state*)

Sets the accel state.

See inv_get_acc_state() to get the value.

Parameters

in	*state*	value to set accel state to.

4.6.2.24 void inv_set_accel_bias_found (int *state*)

Sets whether we know the accel bias.

Parameters

in	*state*	Set to 1 if we know the accel bias. Can be retrieved with inv_got_accel_bias()

4.6.2.25 void inv_set_compass_bias_error (const long ∗ *bias_error*)

Set compass bias error.

See inv_get_compass_bias_error()

Parameters

in	*bias_error*	Set's how accurate we know the compass bias. It is the error squared.

4.6.2.26 void inv_set_compass_bias_found (int *state*)

Sets whether we know the compass bias.

Parameters

in	*state*	Set to 1 if we know the compass bias. Can be retrieved with inv_got_compass_bias()

4.6.2.27 void inv_set_compass_state (int *state*)

Sets the compass state.

Parameters

in	*state*	Compass state. It can be retrieved with inv_get_compass_state().

4.6.2.28 void inv_set_heading_confidence_interval (float *ci*)

Set 9 axis 95% heading confidence interval for quaternion.

Parameters

in	*ci*	Confidence interval in radians.

4.6.2.29 void inv_set_large_mag_field (int *state*)

Set to non-zero if there as a large magnetic field.

See inv_get_large_mag_field() for getting this variable.

Parameters

in	*state*	value to set for magnetic field strength. Should be non-zero if it is large.

4.6.2.30 void inv_set_local_field (const long ∗ *data*)

Sets the local earth's magnetic field.

Parameters

in	*data*	Local earth's magnetic field in uT scaled by 2^16. Length = 3. Y typically points north, Z typically points down in northern hemisphere and up in southern hemisphere.

4.6.2.31 void inv_set_mag_scale (const long ∗ *data*)

Sets the compass sensitivity.

Parameters

in	*data*	Length 3, sensitivity for each compass axis scaled such that 1.0 = 2^30.

4.6.2.32 void inv_set_motion_state (unsigned char *state*)

Sets the motion state.

Parameters

in	*state*	motion state where INV_NO_MOTION is not moving and INV_MOTION is moving.

4.6.2.33 inv_error_t inv_start_results_holder (void)

Function to turn on this module.

This is automatically called by inv_enable_results_holder(). Typically not called by users.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.7 start_manager

Motion Library - Start Manager Start Manager.

Files

- file start_manager.c

  This handles all the callbacks when inv_start_mpl() is called.

Functions

- inv_error_t inv_execute_mpl_start_notification (void)

  Callback all the functions that want to be notified when inv_start_mpl() was called.
- inv_error_t inv_init_start_manager (void)

  Initilize the start manager.
- inv_error_t inv_register_mpl_start_notification (inv_error_t(∗start_cb)(void))

  Register a callback to receive when inv_start_mpl() is called.
- inv_error_t inv_unregister_mpl_start_notification (inv_error_t(∗start_cb)(void))

  Removes a callback from start notification.

4.7.1 Detailed Description

Motion Library - Start Manager Start Manager.

4.7.2 Function Documentation

4.7.2.1 inv_error_t inv_execute_mpl_start_notification (void)

Callback all the functions that want to be notified when inv_start_mpl() was called.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.7.2.2 inv_error_t inv_init_start_manager (void)

Initilize the start manager.

Typically called by inv_start_mpl();

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.7.2.3 inv_error_t inv_register_mpl_start_notification (inv_error_t(∗)(void) *start_cb*)

Register a callback to receive when inv_start_mpl() is called.

Parameters

in	*start_cb*	Function callback that will be called when inv_start_mpl() is called.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.7.2.4 inv_error_t inv_unregister_mpl_start_notification (inv_error_t(∗)(void) *start_cb*)

Removes a callback from start notification.

Parameters

in	*start_cb*	function to remove from start notification

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.8 storage_manager

Motion Library - Stores Data for functions.

Files

- file storage_manager.c

  Load and Store Manager.

Defines

- #define NUM_STORAGE_BOXES 20

  Max number of entites that can be stored.

Functions

- inv_error_t inv_get_mpl_state_size (size_t ∗size)

  Returns the memory size needed to perform a store.
- void inv_init_storage_manager ()

  Should be called once before using any of the storage methods.
- inv_error_t inv_load_mpl_states (const unsigned char ∗data, size_t length)

  This function takes a block of data that has been saved in non-volatile memory and pushes to the proper locations.
- inv_error_t inv_register_load_store (inv_error_t(∗load_func)(const unsigned char ∗data), inv_error_t(∗save_func)(unsigned char ∗data), size_t size, unsigned int key)

  Used to register your mechanism to load and store non-volative data.
- inv_error_t inv_save_mpl_states (unsigned char ∗data, size_t sz)

  This function fills up a block of memory to be stored in non-volatile memory.

4.8.1 Detailed Description

Motion Library - Stores Data for functions.

4.8.2 Function Documentation

4.8.2.1 inv_error_t inv_get_mpl_state_size (size_t ∗ *size*)

Returns the memory size needed to perform a store.

Parameters

out	*size*	Size in bytes of memory needed to store.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.8.2.2 void inv_init_storage_manager ( )

Should be called once before using any of the storage methods.

Typically called first by inv_init_mpl().

4.8.2.3 inv_error_t inv_load_mpl_states (const unsigned char ∗ *data*, size_t *length*)

This function takes a block of data that has been saved in non-volatile memory and pushes to the proper locations.

Multiple error checks are performed on the data.

Parameters

in	*data*	Data that was saved to be loaded up by MPL
in	*length*	Length of data vector in bytes

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.8.2.4 inv_error_t inv_register_load_store (inv_error_t(∗)(const unsigned char ∗data) *load_func*, inv_error_t(∗)(unsigned char ∗data) *save_func*, size_t *size*, unsigned int *key*)

Used to register your mechanism to load and store non-volative data.

This should typical be called during the enable function for your feature.

Parameters

in	*load_func*	function pointer you will use to receive data that was stored for you.
in	*save_func*	function pointer you will use to save any data you want saved to non-volatile memory between runs.

in	*size*	The size in bytes of the amount of data you want loaded and saved.
in	*key*	The key associated with your data type should be unique across MPL. The key should change when your type of data for storage changes.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.8.2.5 inv_error_t inv_save_mpl_states (unsigned char ∗ *data*, size_t *sz*)

This function fills up a block of memory to be stored in non-volatile memory.

Parameters

out	*data*	Place to store data, size of sz, must be at least size returned by inv_get_mpl_state_size()
in	*sz*	Size of data.

Returns

Returns INV_SUCCESS if successful or an error code if not.

4.9 accel_calibration

Accel calibration.

Files

- file accel_auto_cal.c

  Accel calibration.

Defines

- #define INV_ACCEL_CAL_SAVE_KEY (8230)

  Change this key if the definition of the struct auto_cal_obj_t changes.

Functions

- inv_error_t inv_disable_in_use_auto_calibration (void)

  Disables an algorithm to set accel biases.
- inv_error_t inv_enable_in_use_auto_calibration (void)

  Turns on an algorithm to set accel biases.
- inv_error_t inv_init_in_use_auto_calibration (void)

  Init in-use auto calibration.
- inv_error_t inv_start_in_use_auto_calibration (void)

  Start accel bias calibration.
- inv_error_t inv_stop_in_use_auto_calibration (void)

  Turns on an algorithm to set accel biases.

4.9.1 Detailed Description

Accel calibration.

4.9.2 Define Documentation

4.9.2.1 #define INV_ACCEL_CAL_SAVE_KEY (8230)

Change this key if the definition of the struct auto_cal_obj_t changes.

Previous keys: 8227, 8228, 8229

4.9.3 Function Documentation

4.9.3.1 inv_error_t inv_disable_in_use_auto_calibration (void)

Disables an algorithm to set accel biases.

Typically called once per session. See inv_stop_in_use_auto_calibration() to stop the algorithm.

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.9.3.2 inv_error_t inv_enable_in_use_auto_calibration (void)

Turns on an algorithm to set accel biases.

This may be called after inv_init_mpl() and before inv_start_mpl(). It is typically only called once per session.

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.9.3.3 inv_error_t inv_start_in_use_auto_calibration (void)

Start accel bias calibration.

It is automatically called in start mode after an enable. This function only needs to be called to start after a stop command generated by inv_stop_in_use_auto_calibration().

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.9.3.4 inv_error_t inv_stop_in_use_auto_calibration (void)

Turns on an algorithm to set accel biases.

This may be called after inv_init_mpl() and before inv_start_mpl(). It is typically only called once per session. It does not return a motion state.

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.10 small_motion_compass_cal

Calibrates a compass quickly using gyro's but is less accurate than other algorithms.

Files

- file compass_bias_w_gyro.c

Functions

- inv_error_t inv_disable_compass_bias_w_gyro (void)

  Turns off a compass bias from from gyro aglorithm.
- inv_error_t inv_enable_compass_bias_w_gyro (void)

  Turns on a compass bias from from gyro aglorithm.
- void inv_init_compass_bias_w_gyro ()

  Initializes/Resets this module.
- inv_error_t inv_start_compass_bias_w_gyro (void)

  Allows the user to start the coarse compass bias algorithm.
- inv_error_t inv_stop_compass_bias_w_gyro (void)

  Allows the user to stop the coarse compass bias algorithm.

4.10.1 Detailed Description

Calibrates a compass quickly using gyro's but is less accurate than other algorithms.

4.10.2 Function Documentation

4.10.2.1 inv_error_t inv_disable_compass_bias_w_gyro (void)

Turns off a compass bias from from gyro aglorithm.

It is typically only called once per session. It does not return a motion state.

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.10.2.2 inv_error_t inv_enable_compass_bias_w_gyro (void)

Turns on a compass bias from from gyro aglorithm.

This may be called after inv_enable_mpl() and before inv_start_mpl(). It is typically only called once per session. It will automatically turn off, when the more precise algorithms determine a compass bias solution.

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.10.2.3 void inv_init_compass_bias_w_gyro ( )

Initializes/Resets this module.

Called by inv_enable_compass_from_gyro().

4.10.2.4 inv_error_t inv_start_compass_bias_w_gyro (void)

Allows the user to start the coarse compass bias algorithm.

It is automatically called in start mode after an enable. This function only needs to be called to start after a stop command generated by inv_stop_compass_bias_w_gyro().

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.10.2.5 inv_error_t inv_stop_compass_bias_w_gyro (void)

Allows the user to stop the coarse compass bias algorithm.

To start the algorithm back up call inv_start_compass_bias_w_gyro()

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.11 compass_fit

A precise compass bias algorithm.

Files

- file compass_fit.c

Functions

- inv_error_t inv_enable_compass_fit (void)

  Enables the compass fit algorithm.
- void inv_init_compass_fit ()

  Initializes/Resets this module.
- inv_error_t inv_start_compass_fit (void)

  Starts the compass fit algorithm.
- inv_error_t inv_stop_compass_fit (void)

  Stops the compass fit algorithm.

4.11.1 Detailed Description

A precise compass bias algorithm.

4.11.2 Function Documentation

4.11.2.1 inv_error_t inv_enable_compass_fit (void)

Enables the compass fit algorithm.

This should only be called once per library load. See inv_start_compass_fit() and inv_stop_compass_fit() for starting and stopping. Automatically calls inv_start_compass_fit() and inv_init_compass_fit(). Mutually exclusive with inv_enable_vector_compass_cal().

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.11.2.2 void inv_init_compass_fit ( )

Initializes/Resets this module.

Called by inv_enable_compass_fit().

4.11.2.3 inv_error_t inv_start_compass_fit (void)

Starts the compass fit algorithm.

This is automatically called by inv_enable_compass_fit() and only needs to be called after a call to inv_stop_compass_fit().

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.11.2.4 inv_error_t inv_stop_compass_fit (void)

Stops the compass fit algorithm.

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.12 compass_vector_cal

A compass calibration algorithm that is mutually exclusive with compass_fit.

Files

- file compass_fit.c

Functions

- inv_error_t inv_disable_vector_compass_cal (void)

  Disables a precise compass bias algorithm.
- inv_error_t inv_enable_vector_compass_cal (void)

  Enables a precise compass bias algorithm.
- inv_error_t inv_init_vector_compass_cal (void)

  Initializes/Resets this module.
- inv_error_t inv_start_vector_compass_cal (void)

  Allows the user to start a precise compass bias algorithm.
- inv_error_t inv_stop_vector_compass_cal (void)

  Allows the user to stop a precise compass bias algorithm.

4.12.1 Detailed Description

A compass calibration algorithm that is mutually exclusive with compass_fit.

4.12.2 Function Documentation

4.12.2.1 inv_error_t inv_disable_vector_compass_cal (void)

Disables a precise compass bias algorithm.

Should only be called once per library load when you wish to remove this functionality. See inv_stop_vector_compass_cal() if you wish to simply stop the algorithm.

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.12.2.2 inv_error_t inv_enable_vector_compass_cal (void)

Enables a precise compass bias algorithm.

This may be called after inv_init_mpl() and before inv_start_mpl(). It is typically only called once per session. It does not return a motion state. Mutually exclusive with inv_enable_compass_fit().

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.12.2.3 inv_error_t inv_init_vector_compass_cal (void)

Initializes/Resets this module.

Called by inv_enable_vector_compass_cal(). If you are calling this for testing, you probably also want to call inv_init_adv_fusion_obj()

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.12.2.4 inv_error_t inv_start_vector_compass_cal (void)

Allows the user to start a precise compass bias algorithm.

It is automatically called in start mode after an enable. This function only needs to be called to start after a stop command generated by inv_stop_vector_compass_cal().

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.12.2.5 inv_error_t inv_stop_vector_compass_cal (void)

Allows the user to stop a precise compass bias algorithm.

To start the algorithm back up call inv_start_vector_compass_cal()

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.13 fast_no_mot

Fast no motion algorithm used to set the gyro bias.

Files

- file fast_no_motion.c

  Fast no motion algorithm.

Functions

- void int_set_fast_nomot_gyro_threshold (long long thresh)

  Sets internal threshold for fast no motion.
- inv_error_t inv_disable_fast_nomot (void)

  Turns off a faster Motion/No Motion to set gyro biases (see inv_enable_fast_nomot()).
- inv_error_t inv_enable_fast_nomot (void)

  Turns on a faster Motion/No Motion to set gyro biases.
- void inv_fast_nomot_set_gyro_bias (struct inv_sensor_cal_t ∗sensor_cal)

  Used to set gyro bias when no motion is detected.
- void inv_get_fast_nomot_accel_param (long ∗cntr, long long ∗param)

  This is used to help set inv_set_fast_nomot_accel_threshold().
- void inv_get_fast_nomot_compass_param (long ∗cntr, long long ∗param)

  This is used to help set inv_set_fast_nomot_compass_threshold().
- long long inv_get_fnm_gyro_no_motion_param (void)

  Get gyro parameters.
- inv_error_t inv_init_fast_nomot (void)

  Initializes the fast no motion algorithm.
- void inv_set_default_number_of_samples (int N)

  Set default number of samples.
- void inv_set_fast_nomot_accel_threshold (long long thresh)

  Used to set internal threshold.
- void inv_set_fast_nomot_compass_threshold (long long thresh)

  Used to set internal threshold.
- inv_error_t inv_start_fast_nomot (void)

  Allows the user to start the fast no motion algorithm.
- inv_error_t inv_stop_fast_nomot (void)

  Allows the user to stop the fast no motion algorithm.

4.13.1 Detailed Description

Fast no motion algorithm used to set the gyro bias.

4.13.2 Function Documentation

4.13.2.1 inv_error_t inv_disable_fast_nomot (void)

Turns off a faster Motion/No Motion to set gyro biases (see inv_enable_fast_nomot()).

It is typically only called once per session. It does not return a motion state. It is mutually exclusive with inv_enable_motion_no_motion().

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.13.2.2 inv_error_t inv_enable_fast_nomot (void)

Turns on a faster Motion/No Motion to set gyro biases.

This may be called after inv_init_mpl() and before inv_start_mpl(). It is typically only called once per session. It does not return a motion state. It is mutually exclusive with inv_enable_motion_no_motion().

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.13.2.3 void inv_fast_nomot_set_gyro_bias (struct inv_sensor_cal_t ∗ *sensor_cal*)

Used to set gyro bias when no motion is detected.

Parameters

in	*sensor_cal,:*	pointer of the sensor data strurcture

4.13.2.4 void inv_get_fast_nomot_accel_param (long ∗ *cntr,* long long ∗ *param*)

This is used to help set inv_set_fast_nomot_accel_threshold().

cntr is incremented each time there is a new value of param. 100 new values should be sorted from low to high and the 97th value should be used as the threshold parameter for inv_set_fast_nomot_accel_threshold(). The compass must be on.

Parameters

out	*cntr*	Counter for when param changes
out	*param*	Parameter used to help set threshold

4.13.2.5 void inv_get_fast_nomot_compass_param (long ∗ *cntr*, long long ∗ *param*)

This is used to help set inv_set_fast_nomot_compass_threshold().

cntr is incremented each time there is a new value of param. 100 new values should be sorted from low to high and the 97th value should be used as the threshold in inv_set_fast_nomot_compass_threshold(). The compass must be on.

Parameters

out	*cntr*	Counter for when param changes
out	*param*	Parameter used to help set threshold

4.13.2.6 inv_error_t inv_init_fast_nomot (void)

Initializes the fast no motion algorithm.

Automatically called by inv_enable_fast_nomot(). Not typically called by the user.

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.13.2.7 void inv_set_default_number_of_samples (int *N*)

Set default number of samples.

Not typically called by users.

Parameters

in	*N*	Number of samples to use for algorithm

4.13.2.8 void inv_set_fast_nomot_accel_threshold (long long *thresh*)

Used to set internal threshold.

This may need to be set based upon device environment. See inv_get_fast_nomot_accel_param() for values a range of values to set this too.

Parameters

in	*thresh*	

4.13.2.9 void inv_set_fast_nomot_compass_threshold (long long *thresh*)

Used to set internal threshold.

This may need to be set based upon device environment. See inv_get_fast_nomot_compass_param() for values a range of values to set this too.

Parameters

in	*thresh*	

4.13.2.10 inv_error_t inv_start_fast_nomot (void)

Allows the user to start the fast no motion algorithm.

It is automatically in start mode after an enable. This function only needs to be called to start after a stop command generated by inv_stop_fast_nomot().

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.13.2.11 inv_error_t inv_stop_fast_nomot (void)

Allows the user to stop the fast no motion algorithm.

See inv_start_fast_nomot() to start the algorithm back up.

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.14 nine_axis_fusion

Performs nine axis sensor fusion.

Files

- file fusion_9axis.c

  Performs nine axis sensor fusion.

Functions

- inv_error_t inv_9x_fusion_enable_jitter_reduction (int en)

  This enables the jitter reduction feature.
- inv_error_t inv_9x_fusion_set_mag_fb (double fb)

  This sets the magnetic feedback.
- inv_error_t inv_9x_fusion_use_timestamps (int en)

  Use timestamps when evaluating compass correction gain.
- inv_error_t inv_disable_9x_sensor_fusion ()

  Disables the 9 axis sensor fusion algorithm.
- inv_error_t inv_enable_9x_sensor_fusion (void)

  Enables the 9 axis sensor fusion algorithm.
- void inv_init_9x_fusion (void)

  Initializes the algorithm.
- inv_error_t inv_start_9x_sensor_fusion (void)

  Starts the 9 axis sensor fusion.
- inv_error_t inv_stop_9x_sensor_fusion (void)

  Stops the 9 axis sensor fusion from running.

4.14.1 Detailed Description

Performs nine axis sensor fusion.

4.14.2 Function Documentation

4.14.2.1 inv_error_t inv_9x_fusion_enable_jitter_reduction (int *en*)

This enables the jitter reduction feature.

Parameters

in	*en*	Should be non-zero to enable the feature. Initialized to 0, i.e. off

Returns

heading correction angle

4.14.2.2 inv_error_t inv_9x_fusion_set_mag_fb (double *fb*)

This sets the magnetic feedback.

Increasing it resutls in faster compass correction in the 9 axis quaternion.

Parameters

in	*fb*	Desired magnetic feedback value. Typical value is 1. Also, initialized to 1 in inv_init_9x_fusion.

Returns

heading correction angle

4.14.2.3 inv_error_t inv_9x_fusion_use_timestamps (int *en*)

Use timestamps when evaluating compass correction gain.

This feature should be used when the MPL is not receiving compass data at a constant rate.

Parameters

in	*en*	1 to enable the feature.

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.14.2.4 inv_error_t inv_disable_9x_sensor_fusion ( )

Disables the 9 axis sensor fusion algorithm.

Should only be called once per library load when you wish to remove this functionality. See inv_stop_9x_sensor_fusion() if you wish to simply stop the algorithm.

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.14.2.5 inv_error_t inv_enable_9x_sensor_fusion (void)

Enables the 9 axis sensor fusion algorithm.

This should only be called once per library load. See inv_start_9x_sensor_fusion() and inv_stop_9x_sensor_fusion() for starting and stopping. Automatically calls inv_start_9x_sensor_fusion() and inv_init_9x_fusion().

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.14.2.6 void inv_init_9x_fusion (void)

Initializes the algorithm.

Automatically called by inv_enable_9x_sensor_fusion(). Not normally called by users.

4.14.2.7 inv_error_t inv_start_9x_sensor_fusion (void)

Starts the 9 axis sensor fusion.

Automatically called by inv_enable_9x_sensor_fusion() and only needs to be called after stopping with inv_stop_9x_sensor_fusion().

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.14.2.8 inv_error_t inv_stop_9x_sensor_fusion (void)

Stops the 9 axis sensor fusion from running.

See inv_start_9x_sensor_fusion() to start it back up again.

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.15 gyro_tc

Gyro Temperature Compensation algorithm.

Files

- file gyro_tc.c

  Gyro bias temperature compensation.

Defines

- #define INV_GTC_SAVE_KEY (308)

  Change this key if the definition of the struct inv_gtc changes.

Functions

- inv_error_t inv_disable_gyro_tc (void)

  Enable the gyro temp comp algorithm.
- inv_error_t inv_enable_gyro_tc (void)

  Enable the gyro temp comp algorithm.
- inv_error_t inv_init_gyro_ts (void)

  Reset the gyro temp slope.
- inv_error_t inv_start_gyro_tc (void)

  Registers callback to receive new temperature data.
- inv_error_t inv_stop_gyro_tc (void)

  Unregisters callback.

4.15.1 Detailed Description

Gyro Temperature Compensation algorithm.

4.15.2 Define Documentation

4.15.2.1 #define INV_GTC_SAVE_KEY (308)

Change this key if the definition of the struct inv_gtc changes.

Previous keys: -none-

4.15.3 Function Documentation

4.15.3.1 inv_error_t inv_disable_gyro_tc (void)

Enable the gyro temp comp algorithm.

Returns

INV_SUCCESS if successful.

4.15.3.2 inv_error_t inv_enable_gyro_tc (void)

Enable the gyro temp comp algorithm.

Returns

INV_SUCCESS if successful.

4.15.3.3 inv_error_t inv_init_gyro_ts (void)

Reset the gyro temp slope.

Returns

INV_SUCCESS if successful.

4.15.3.4 inv_error_t inv_start_gyro_tc (void)

Registers callback to receive new temperature data.

Returns

INV_SUCCESS if successful.

4.15.3.5 inv_error_t inv_stop_gyro_tc (void)

Unregisters callback.

Returns

INV_SUCCESS if successful.

4.16 heading_from_gyro

A less accurate but fast algorithm for 9 axis sensor fusion.

Files

- file heading_from_gyro.c

Functions

- inv_error_t inv_disable_heading_from_gyro (void)

  Turns off a heading from gyro.
- inv_error_t inv_enable_heading_from_gyro (void)

  Turns on a heading from gyro algorithm which performs sensor fusion when the compass bias hasn't been fully solved for.
- void inv_init_heading_from_gyro (void)

  Initializes/Resets this module.
- inv_error_t inv_start_heading_from_gyro (void)

  Registers callback to recieve gyro and compass data.
- inv_error_t inv_stop_heading_from_gyro (void)

  Unregisters callback.

4.16.1 Detailed Description

A less accurate but fast algorithm for 9 axis sensor fusion.

4.16.2 Function Documentation

4.16.2.1 inv_error_t inv_disable_heading_from_gyro (void)

Turns off a heading from gyro.

It is typically only called once per session.

Returns

INV_SUCCESS if successful.

4.16.2.2 inv_error_t inv_enable_heading_from_gyro (void)

Turns on a heading from gyro algorithm which performs sensor fusion when the compass bias hasn't been fully solved for.

This may be called after inv_init_mpl() and before inv_start_mpl(). It is typically only called once per session.

Returns

INV_SUCCESS if successful.

4.16.2.3 void inv_init_heading_from_gyro (void)

Initializes/Resets this module.

Called by inv_enable_heading_from_gyro().

Returns

INV_SUCCESS if successful.

4.16.2.4 inv_error_t inv_start_heading_from_gyro (void)

Registers callback to recieve gyro and compass data.

Returns

INV_SUCCESS if successful.

4.16.2.5 inv_error_t inv_stop_heading_from_gyro (void)

Unregisters callback.

Returns

INV_SUCCESS if successful.

4.17 mag_disturb

Determines magnetic disturbances and sets compass accuracy appropriately.

Files

- file mag_disturb.c

Functions

- inv_error_t inv_disable_magnetic_disturbance (void)

  Turns off a magnetic disturbance algorithm (see inv_enable_magnetic_disturbance()).
- inv_error_t inv_enable_magnetic_disturbance (void)

  Enables a magnetic disturbance algorithm.
- inv_error_t inv_start_magnetic_disturbance (void)

  Allows the user to start the magnetic disturbance algorithm.
- inv_error_t inv_stop_magnetic_disturbance (void)

  Allows the user to stop the magnetic disturbance algorithm.

4.17.1 Detailed Description

Determines magnetic disturbances and sets compass accuracy appropriately.

4.17.2 Function Documentation

4.17.2.1 inv_error_t inv_disable_magnetic_disturbance (void)

Turns off a magnetic disturbance algorithm (see inv_enable_magnetic_disturbance()).

It is typically only called once per session. See inv_stop_magnetic_disturbance() to stop the algorithm

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.17.2.2 inv_error_t inv_enable_magnetic_disturbance (void)

Enables a magnetic disturbance algorithm.

This may be called after inv_init_mpl() and before inv_start_mpl(). It is typically only called once per session. It does not return a motion state.

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.17.2.3 inv_error_t inv_start_magnetic_disturbance (void)

Allows the user to start the magnetic disturbance algorithm.

It is automatically called in start mode after an enable. This function only needs to be called to start after a stop command generated by inv_stop_magnetic_disturbance().

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.17.2.4 inv_error_t inv_stop_magnetic_disturbance (void)

Allows the user to stop the magnetic disturbance algorithm.

To start the algorithm back up call inv_start_no_gyro_fusion()

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.18 motion_no_motion

A motion detection algorithm that is used to set gyro bias when the device is not moving.

Files

- file motion_no_motion.c

  A motion detection algorithm that is used to set gyro bias when the device is not moving.

Functions

- inv_error_t inv_disable_motion_no_motion (void)

  Turns off Motion/No Motion to set gyro biases (see inv_enable_motion_no_motion()).
- inv_error_t inv_enable_motion_no_motion ()

  Turns on Motion/No Motion used to set gyro biases.
- inv_error_t inv_init_motion_no_motion (void)

  Initializes the motion no motion algorithm.
- inv_error_t inv_set_no_motion_time (long time_ms)

  Allows the user to set the time to be in a no motion state before setting the gyro bias.
- inv_error_t inv_start_motion_no_motion (void)

  Allows the user to start the no motion algorithm.
- inv_error_t inv_stop_motion_no_motion (void)

  Allows the user to stop the no motion algorithm.

4.18.1 Detailed Description

A motion detection algorithm that is used to set gyro bias when the device is not moving.

4.18.2 Function Documentation

4.18.2.1 inv_error_t inv_disable_motion_no_motion (void)

Turns off Motion/No Motion to set gyro biases (see inv_enable_motion_no_motion()).

It is typically only called once per session. It does not return a motion state. It is mutually exclusive with inv_enable_fast_nomot().

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.18.2.2 inv_error_t inv_enable_motion_no_motion ( )

Turns on Motion/No Motion used to set gyro biases.

This may be called after inv_init_mpl() and before inv_start_mpl(). It is typically only called once per session. It does not return a motion state. It is mutually exclusive with inv_enable_motion_no_motion().

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.18.2.3 inv_error_t inv_init_motion_no_motion (void)

Initializes the motion no motion algorithm.

Automatically called by inv_enable_motion_no_motion(). Not typically called by the user.

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.18.2.4 inv_error_t inv_set_no_motion_time (long *time_ms*)

Allows the user to set the time to be in a no motion state before setting the gyro bias.

Parameters

in	*time_ms*	Time in milliseconds. Default is 8000ms or 8 seconds.

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.18.2.5 inv_error_t inv_start_motion_no_motion (void)

Allows the user to start the no motion algorithm.

It is automatically called in start mode after an enable. This function only needs to be called to start after a stop command generated by inv_stop_motion_no_motion().

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.18.2.6 inv_error_t inv_stop_motion_no_motion (void)

Allows the user to stop the no motion algorithm.

See inv_start_motion_no_motion() to start the algorithm back up.

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.19 no_gyro_fusion

Accel/Compass Sensor fusion.

Files

- file no_gyro_fusion.c

  Accel/Compass Sensor fusion.

Functions

- inv_error_t inv_disable_no_gyro_fusion (void)

  Turns off a sensor fusion using accel and compass only (see inv_enable_no_gyro_fusion()).
- inv_error_t inv_enable_no_gyro_fusion (void)

  Enables a sensor fusion using accel and compass only.
- inv_error_t inv_init_no_gyro_fusion (void)

  Initializes the algorithm.
- inv_error_t inv_start_no_gyro_fusion (void)

  Allows the user to start the sensor fusion using accel and compass only algorithm.
- inv_error_t inv_stop_no_gyro_fusion (void)

  Allows the user to stop the sensor fusion using accel and compass only algorithm.

4.19.1 Detailed Description

Accel/Compass Sensor fusion.

4.19.2 Function Documentation

4.19.2.1 inv_error_t inv_disable_no_gyro_fusion (void)

Turns off a sensor fusion using accel and compass only (see inv_enable_no_gyro_fusion()).

It is typically only called once per session. See inv_stop_no_gyro_fusion() to stop the algorithm

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.19.2.2 inv_error_t inv_enable_no_gyro_fusion (void)

Enables a sensor fusion using accel and compass only.

This may be called after inv_init_mpl() and before inv_start_mpl(). It is typically only called once per session. It does not return a motion state.

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.19.2.3 inv_error_t inv_init_no_gyro_fusion (void)

Initializes the algorithm.

Automatically called by the enable function.

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.19.2.4 inv_error_t inv_start_no_gyro_fusion (void)

Allows the user to start the sensor fusion using accel and compass only algorithm.

It is automatically called in start mode after an enable. This function only needs to be called to start after a stop command generated by inv_stop_no_gyro_fusion().

Returns

INV_SUCCESS on success or an error code if call was not successful.

4.19.2.5 inv_error_t inv_stop_no_gyro_fusion (void)

Allows the user to stop the sensor fusion using accel and compass only algorithm.

See inv_start_no_gyro_fusion() to start the algorithm back up call inv_start_no_gyro_fusion()

Returns

INV_SUCCESS on success or an error code if call was not successful.

Index