U-BET

Generated by Doxygen 1.7.6.1

Sat Apr 27 2013 15:33:50

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class Hierarchy

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

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 cPWM Namespace Reference

Simple C++ class wrapper for beaglebone PWM eHRPWM interface.

Classes

- class cPWM

5.1.1 Detailed Description

Simple C++ class wrapper for beaglebone PWM eHRPWM interface.

5.2 setPWMReg Namespace Reference

Variables

- int MMAP_OFFSET = 0x44c00000
- int MMAP_SIZE = 0x48ffffff
- int CM_PER_BASE = 0x44e00000
- int CM_PER_EPWMSS1_CLKCTRL = 0xcc
- int CM_PER_EPWMSS0_CLKCTRL = 0xd4
- int CM_PER_EPWMSS2_CLKCTRL = 0xd8
- tuple mem = mmap(f.fileno(), MMAP_SIZE, offset=MMAP_OFFSET)
- tuple val = _getReg(CM_PER_EPWMSS1_CLKCTRL)

5.2.1 Variable Documentation

5.2.1.1 int setPWMReg::CM_PER_BASE = 0x44e00000

Definition at line 7 of file setPWMReg.py.

5.2.1.2 int setPWMReg::CM_PER_EPWMSS0_CLKCTRL = 0xd4

Definition at line 9 of file setPWMReg.py.

5.2.1.3 int setPWMReg::CM_PER_EPWMSS1_CLKCTRL = 0xcc

Definition at line 8 of file setPWMReg.py.

5.2.1.4 int setPWMReg::CM_PER_EPWMSS2_CLKCTRL = 0xd8

Definition at line 10 of file setPWMReg.py.

5.2.1.5 tuple setPWMReg::mem = mmap(f.fileno(), MMAP_SIZE, offset=MMAP_OFFSET)

Definition at line 12 of file setPWMReg.py.

5.2.1.6 int setPWMReg::MMAP_OFFSET = 0x44c00000

Definition at line 5 of file setPWMReg.py.

5.2.1.7 int setPWMReg::MMAP_SIZE = 0x48ffffff

Definition at line 6 of file setPWMReg.py.

5.2.1.8 tuple setPWMReg::val = _getReg(CM_PER_EPWMSS1_CLKCTRL)

Definition at line 28 of file setPWMReg.py.

5.3 USU Namespace Reference

TODO: Make some proper exceptions.

Classes

- class GX3Communicator

  Represents the Thread class for communication with the 3DM-GX3-25.

- class GX3Packet

  Abstract base class for received packets.
- class RawAccAng

  Representation for receiving (raw) acceleration & angular rate packets.
- class AccAngMag

  Representation for receiving acceleration, angular rate and magnetometer packets.
- class AccAngMagOrientationMat

  Representation for packets containing the 3 sensor vectors and orientation matrix This class can be used with the commands which return 3 Vectors and a 3x3 Matrix. The units are:
- class GX3Command

  Base class for commands send to the 3DM-GX3-25.
- class SetCountinuousMode

  Represents the "Set continuous mode" command.
- class SamplingSettings

  Represents the "Sampling Settings" command.
- class KalmanFilter

  Represents the Periodic Thread class for state estimation.
- class MotorControl

  Represents the Periodic task for motor control.
- class Motor
- class Lock

  Wrapper class for pthread mutexes.
- class ScopedLock

  Provides a helper class for Scoped Mutexes.
- class PeriodicRtThread

  TODO: Make some proper exceptions.
- class RtThread

  Abstract wrapper class for the pthread library with RT-priority.

Variables

- const uint8_t RAW_ACC_ANG = 0xC1
- const uint8_t ACC_ANG = 0xC2
- const uint8_t DELTA_ANGLE_VEL = 0xC3
- const uint8_t SET_CONTINUOUS_MODE = 0xC4
- const uint8_t ORIENTATION_MATRIX = 0xC5
- const uint8_t ORIENTATION_UPDATE_MAT = 0xC6
- const uint8_t MAG_VEC = 0xC7
- const uint8_t ACC_ANG_ORIENTATION_MAT = 0xC8
- const uint8_t WRITE_ACC_BIAS_CORRECTION = 0xC9
- const uint8_t WRITE_GYRO_BIAS_CORRECTION = 0xCA
- const uint8_t ACC_ANG_MAG_VEC = 0xCB
- const uint8_t ACC_ANG_MAG_VEC_ORIENTATION_MAT = 0xCC

- const uint8_t CAPTURE_GYRO_BIAS = 0xCD
- const uint8_t EULER_ANGLES = 0xCE
- const uint8_t EULER_ANGLES_ANG_RATES = 0xCF
- const uint8_t TRANSFER_TO_NONVOL_MEM = 0xD0
- const uint8_t TEMPERATURES = 0xD1
- const uint8_t GYRO_STABIL_ACC_ANG_MAG = 0xD2
- const uint8_t DELTA_ANGLE_VEL_MAG_VEC = 0xD3
- const uint8_t MODE = 0xD4
- const uint8_t MODE_PRESET = 0xD5
- const uint8_t CONTINUOUS_PRESET = 0xD6
- const uint8_t TIMER = 0xD7
- const uint8_t COMM_SETTINGS = 0xD9
- const uint8_t STATIONARY_TEST = 0xDA
- const uint8_t SAMPLING_SETTINGS = 0xDB
- const uint8_t REALIGN_UP_NORTH = 0xDD
- const uint8_t QUATERNION = 0xDF
- const uint8_t WRITE_WORD_EEPROM = 0xE4
- const uint8_t READ_WORD_EEPROM = 0xE5
- const uint8_t READ_FIRMWARE_VER = 0xE9
- const uint8_t READ_DEVICE_ID = 0xEA
- const uint8_t STOP_CONTINUOUS = 0xFA
- const uint8_t FIRMWARE_UPDATE = 0xFD
- const uint8_t DEVICE_RESET = 0xFE

5.3.1 Detailed Description

TODO: Make some proper exceptions.

5.3.2 Variable Documentation

5.3.2.1 const uint8_t USU::ACC_ANG = 0xC2

Acceleration & Angular Rate

Definition at line 25 of file messages.h.

5.3.2.2 const uint8_t USU::ACC_ANG_MAG_VEC = 0xCB

Acceleration, Angular Rate & Magnetometer Vector

Definition at line 34 of file messages.h.

5.3.2.3 const uint8_t USU::ACC_ANG_MAG_VEC_ORIENTATION_MAT = 0xCC

Acceleration, Angular Rate & Magnetometer Vectors & Orientation Matrix

Definition at line 35 of file messages.h.

5.3.2.4 const uint8_t USU::ACC_ANG_ORIENTATION_MAT = 0xC8

Acceleration, Angular Rate & Orientation Matrix

Definition at line 31 of file messages.h.

5.3.2.5 const uint8_t USU::CAPTURE_GYRO_BIAS = 0xCD

Capture Gyro Bias

Definition at line 36 of file messages.h.

5.3.2.6 const uint8_t USU::COMM_SETTINGS = 0xD9

Communications Settings

Definition at line 47 of file messages.h.

5.3.2.7 const uint8_t USU::CONTINUOUS_PRESET = 0xD6

Continuous Preset

Definition at line 45 of file messages.h.

5.3.2.8 const uint8_t USU::DELTA_ANGLE_VEL = 0xC3

DeltaAngle & DeltaVelocity

Definition at line 26 of file messages.h.

5.3.2.9 const uint8_t USU::DELTA_ANGLE_VEL_MAG_VEC = 0xD3

DeltaAngle & DeltaVelocity & Magnetometer Vectors

Definition at line 42 of file messages.h.

5.3.2.10 const uint8_t USU::DEVICE_RESET = 0xFE

Device Reset (no reply)

Definition at line 58 of file messages.h.

5.3.2.11 const uint8_t USU::EULER_ANGLES = 0xCE

Euler Angles

Definition at line 37 of file messages.h.

5.3.2.12 const uint8_t USU::EULER_ANGLES_ANG_RATES = 0xCF

Euler Angles and Angular Rates

Definition at line 38 of file messages.h.

5.3.2.13 const uint8_t USU::FIRMWARE_UPDATE = 0xFD

Firmware Update (no reply)

Definition at line 57 of file messages.h.

5.3.2.14 const uint8_t USU::GYRO_STABIL_ACC_ANG_MAG = 0xD2

Gyro Stabilized Acceleration, Angular Rate & Magnetometer

Definition at line 41 of file messages.h.

5.3.2.15 const uint8_t USU::MAG_VEC = 0xC7

Magnetometer Vector

Definition at line 30 of file messages.h.

5.3.2.16 const uint8_t USU::MODE = 0xD4

Mode

Definition at line 43 of file messages.h.

5.3.2.17 const uint8_t USU::MODE_PRESET = 0xD5

Mode Preset

Definition at line 44 of file messages.h.

5.3.2.18 const uint8_t USU::ORIENTATION_MATRIX = 0xC5

Orientation Matrix

Definition at line 28 of file messages.h.

5.3.2.19 const uint8_t USU::ORIENTATION_UPDATE_MAT = 0xC6

Orientation Update Matrix

Definition at line 29 of file messages.h.

5.3.2.20 const uint8_t USU::QUATERNION = 0xDF

Quaternion

Definition at line 51 of file messages.h.

5.3.2.21 const uint8_t USU::RAW_ACC_ANG = 0xC1

Raw Accelerometer and Angular Rate Sensor Outputs

Definition at line 24 of file messages.h.

5.3.2.22 const uint8_t USU::READ_DEVICE_ID = 0xEA

Read Device ID String

Definition at line 55 of file messages.h.

5.3.2.23 const uint8_t USU::READ_FIRMWARE_VER = 0xE9

Read Firmware Version Number

Definition at line 54 of file messages.h.

5.3.2.24 const uint8_t USU::READ_WORD_EEPROM = 0xE5

Read Word from EEPROM

Definition at line 53 of file messages.h.

5.3.2.25 const uint8_t USU::REALIGN_UP_NORTH = 0xDD

Realign Up and North

Definition at line 50 of file messages.h.

5.3.2.26 const uint8_t USU::SAMPLING_SETTINGS = 0xDB

Sampling Settings

Definition at line 49 of file messages.h.

5.3.2.27 const uint8_t USU::SET_CONTINUOUS_MODE = 0xC4

Set Continuous Mode

Definition at line 27 of file messages.h.

5.3.2.28 const uint8_t USU::STATIONARY_TEST = 0xDA

Stationary Test

Definition at line 48 of file messages.h.

5.3.2.29 const uint8_t USU::STOP_CONTINUOUS = 0xFA

Stop Continuous Mode (no reply)

Definition at line 56 of file messages.h.

5.3.2.30 const uint8_t USU::TEMPERATURES = 0xD1

Temperatures

Definition at line 40 of file messages.h.

5.3.2.31 const uint8_t USU::TIMER = 0xD7

Timer

Definition at line 46 of file messages.h.

5.3.2.32 const uint8_t USU::TRANSFER_TO_NONVOL_MEM = 0xD0

Transfer Quantity to Non-Volatile Memory

Definition at line 39 of file messages.h.

5.3.2.33 const uint8_t USU::WRITE_ACC_BIAS_CORRECTION = 0xC9

Write Accel Bias Correction

Definition at line 32 of file messages.h.

5.3.2.34 const uint8_t USU::WRITE_GYRO_BIAS_CORRECTION = 0xCA

Write Gyro Bias Correction

Definition at line 33 of file messages.h.

5.3.2.35 const uint8_t USU::WRITE_WORD_EEPROM = 0xE4

Write Word to EEPROM

Definition at line 52 of file messages.h.

Chapter 6

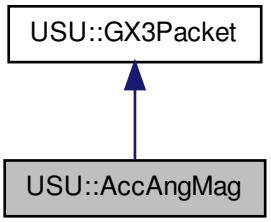
Class Documentation

6.1 USU::AccAngMag Class Reference

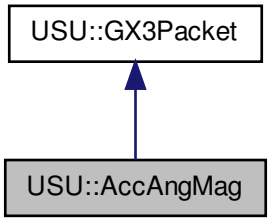
Representation for receiving acceleration, angular rate and magnetometer packets.

`#include <messages.h>`

Inheritance diagram for USU::AccAngMag:



Collaboration diagram for USU::AccAngMag:



Public Types

- enum { size = 43 }

Public Member Functions

- AccAngMag (uint8_t *buffer)

  Creates a packet object from the passed buffer.

Public Attributes

- vector acc
- vector gyro
- vector mag
- unsigned int timer

6.1.1 Detailed Description

Representation for receiving acceleration, angular rate and magnetometer packets.

This class can be used with the commands which return 3 Vectors. The units are:

- acceleration: g
- angular rate: rad/s
- magnetic field: gauß

Definition at line 181 of file messages.h.

6.1.2 Member Enumeration Documentation

6.1.2.1 anonymous enum

Enumerator:

size

Definition at line 207 of file messages.h.

6.1.3 Constructor & Destructor Documentation

6.1.3.1 USU::AccAngMag::AccAngMag (uint8_t * *buffer*) [inline]

Creates a packet object from the passed buffer.

The checksum should have been tested before.

Parameters

buffer	pointer to the byte array containing the received data

Definition at line 191 of file messages.h.

6.1.4 Member Data Documentation

6.1.4.1 vector USU::AccAngMag::acc

Vector containing the accelerometer data

Definition at line 201 of file messages.h.

6.1.4.2 vector USU::AccAngMag::gyro

Vector containing the gyroscope (angular rate) data

Definition at line 202 of file messages.h.

6.1.4.3 vector USU::AccAngMag::mag

Vector containing the magnetometer data

Definition at line 203 of file messages.h.

6.1.4.4 unsigned int USU::AccAngMag::timer

The value of the timestamp for the package

Definition at line 205 of file messages.h.

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

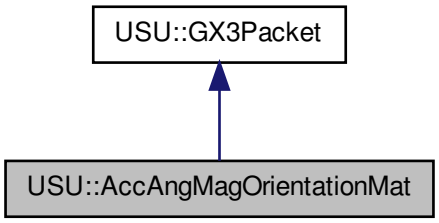
- 3dm/messages.h

6.2 USU::AccAngMagOrientationMat Class Reference

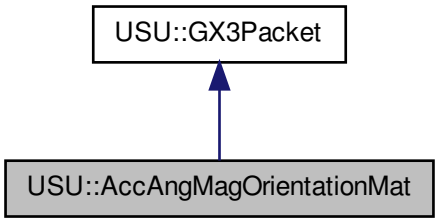
Representation for packets containing the 3 sensor vectors and orientation matrix This class can be used with the commands which return 3 Vectors and a 3x3 Matrix. The units are:

`#include <messages.h>`

Inheritance diagram for USU::AccAngMagOrientationMat:



Collaboration diagram for USU::AccAngMagOrientationMat:



Public Types

- enum { size = 79 }

Public Member Functions

- AccAngMagOrientationMat (uint8_t *buffer)

  Creates a packet object from the passed buffer.

Public Attributes

- vector acc
- vector gyro
- vector mag
- matrix orientation
- unsigned int timer

6.2.1 Detailed Description

Representation for packets containing the 3 sensor vectors and orientation matrix This class can be used with the commands which return 3 Vectors and a 3x3 Matrix. The units are:

- acceleration: g
- angular rate: rad/s
- magnetic field: gauß

Definition at line 219 of file messages.h.

6.2.2 Member Enumeration Documentation

6.2.2.1 anonymous enum

Enumerator:

size

Definition at line 246 of file messages.h.

6.2.3 Constructor & Destructor Documentation

6.2.3.1 USU::AccAngMagOrientationMat::AccAngMagOrientationMat (uint8_t * *buffer*) [inline]

Creates a packet object from the passed buffer.

The checksum should have been tested before.

Parameters

buffer	pointer to the byte array containing the received data

Definition at line 229 of file messages.h.

6.2.4 Member Data Documentation

6.2.4.1 vector USU::AccAngMagOrientationMat::acc

Vector containing the accelerometer data

Definition at line 239 of file messages.h.

6.2.4.2 vector USU::AccAngMagOrientationMat::gyro

Vector containing the gyroscope (angular rate) data

Definition at line 240 of file messages.h.

6.2.4.3 vector USU::AccAngMagOrientationMat::mag

Vector containing the magnetometer data

Definition at line 241 of file messages.h.

6.2.4.4 matrix USU::AccAngMagOrientationMat::orientation

3x3 Matrix containing the orientation

Definition at line 243 of file messages.h.

6.2.4.5 unsigned int USU::AccAngMagOrientationMat::timer

The value of the timestamp for the package

Definition at line 244 of file messages.h.

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

- 3dm/messages.h

6.3 Beagle_GPIO Class Reference

`#include <Beagle_GPIO.h>`

Public Types

- enum Beagle_GPIO_Status { kFail = 0, kSuccess = 1 }
- enum { kREVISION = 0x0, kSYSCONFIG = 0x10, kIRQSTATUS_RAW_0 = 0x24, kIRQSTATUS_RAW_1 = 0x28, kIRQSTATUS_0 = 0x2C, kIRQSTATUS_1 = 0x30, kIRQSTATUS_SET_0 = 0x34, kIRQSTATUS_SET_1 = 0x38, kIRQSTATUS_CLR_0 = 0x3C, kIRQSTATUS_CLR_1 = 0x40, kIRQWAKEN_0 = 0x44, kIRQWAKEN_1 = 0x48, kSYSSTATUS = 0x114, kCTRL = 0x130, kOE = 0x134, kDATAIN = 0x138, kDATAOUT = 0x13C, kLEVELDETECT0 = 0x140, kLEVELDETECT1 = 0x144, kRISINGDETECT = 0x148, kFALLINGDETECT = 0x14C, kDEBOUNCEENABLE = 0x150, kDEBOUNCINGTIME = 0x154, kCLEARDATAOUT = 0x190, kSETDATAOUT = 0x194 }
- enum Beagle_GPIO_Direction { kINPUT = 0, kOUTPUT = 1 }
- enum { P8_1, P8_2, P8_3, P8_4, P8_5, P8_6, P8_7, P8_8, P8_9, P8_10, P8_11, P8_12, P8_13, P8_14, P8_15, P8_16, P8_17, P8_18, P8_19, P8_20, P8_21, P8_22, P8_23, P8_24, P8_25, P8_26, P8_27, P8_28, P8_29, P8_30, P8_31, P8_32, P8_33, P8_34, P8_35, P8_36, P8_37, P8_38, P8_39, P8_40, P8_41, P8_42, P8_43, P8_44, P8_45, P8_46, P9_1, P9_2, P9_3, P9_4, P9_5, P9_6, P9_7, P9_8, P9_9, P9_10, P9_11, P9_12, P9_13, P9_14, P9_15, P9_16, P9_17, P9_18, P9_19, P9_20, P9_21, P9_22, P9_23, P9_24, P9_25, P9_26, P9_27, P9_28, P9_29, P9_30, P9_31, P9_32, P9_33, P9_34, P9_35, P9_36, P9_37, P9_38, P9_39, P9_40, P9_41, P9_42, P9_43, P9_44, P9_45, P9_46 }

Public Member Functions

- Beagle_GPIO ()
- ~Beagle_GPIO ()
- Beagle_GPIO_Status configurePin (unsigned short _pin, Beagle_GPIO_Direction _direction)
- Beagle_GPIO_Status enablePinInterrupts (unsigned short _pin, bool _enable)
- Beagle_GPIO_Status writePin (unsigned short _pin, unsigned char _value)
- unsigned char readPin (unsigned short _pin)
- void openSPI (unsigned char _mode=0, unsigned char _bits=8, unsigned long _speed=4800000, unsigned short _delay=0)
- void closeSPI ()
- void sendSPIBuffer (unsigned long buffer, int size)
- bool isActive ()

Public Attributes

- enum Beagle_GPIO:: { ... } Beagle_GPIO_Registers
- enum Beagle_GPIO:: { ... } GPIO_Pins

Static Public Attributes

- static const int GPIO_Pin_Bank []
- static const int GPIO_Pin_Id []
- static const unsigned long GPIO_Pad_Control []
- static const unsigned long GPIO_Control_Module_Registers = 0x44E10000
- static const unsigned long GPIO_Base []

6.3.1 Detailed Description

Definition at line 48 of file Beagle_GPIO.h.

6.3.2 Member Enumeration Documentation

6.3.2.1 anonymous enum

Enumerator:

kREVISION
kSYSCONFIG
kIRQSTATUS_RAW_0
kIRQSTATUS_RAW_1
kIRQSTATUS_0
kIRQSTATUS_1
kIRQSTATUS_SET_0
kIRQSTATUS_SET_1
kIRQSTATUS_CLR_0
kIRQSTATUS_CLR_1
kIRQWAKEN_0
kIRQWAKEN_1
kSYSSTATUS
kCTRL
kOE
kDATAIN
kDATAOUT
kLEVELDETECT0
kLEVELDETECT1
kRISINGDETECT
kFALLINGDETECT
kDEBOUNCEENABLE
kDEBOUNCINGTIME
kCLEARDATAOUT
kSETDATAOUT

Definition at line 59 of file Beagle_GPIO.h.

6.3.2.2 anonymous enum

Enumerator:

P8_1

P8_2

P8_3

P8_4

P8_5

P8_6

P8_7

P8_8

P8_9

P8_10

P8_11

P8_12

P8_13

P8_14

P8_15

P8_16

P8_17

P8_18

P8_19

P8_20

P8_21

P8_22

P8_23

P8_24

P8_25

P8_26

P8_27

P8_28

P8_29

P8_30

P8_31

P8_32

P8_33

P8_34

P8_35

P8_36

P8_37

P8_38

P8_39

P8_40

P8_41

P8_42

P8_43

P8_44

P8_45

P8_46

P9_1

P9_2

P9_3

P9_4

P9_5

P9_6

P9_7

P9_8

P9_9

P9_10

P9_11

P9_12

P9_13

P9_14

P9_15

P9_16

P9_17

P9_18

P9_19

P9_20

P9_21

P9_22

P9_23

P9_24

P9_25

P9_26

P9_27

P9_28

P9_29

P9_30

P9_31

P9_32

P9_33

P9_34

P9_35

P9_36

P9_37

P9_38

P9_39

P9_40

P9_41

P9_42

P9_43

P9_44

P9_45

P9_46

Definition at line 96 of file Beagle_GPIO.h.

6.3.2.3 enum Beagle_GPIO::Beagle_GPIO_Direction

Enumerator:

kINPUT

kOUTPUT

Definition at line 89 of file Beagle_GPIO.h.

6.3.2.4 enum Beagle_GPIO::Beagle_GPIO_Status

Enumerator:

kFail

kSuccess

Definition at line 52 of file Beagle_GPIO.h.

6.3.3 Constructor & Destructor Documentation

6.3.3.1 Beagle_GPIO::Beagle_GPIO ( )

Definition at line 127 of file Beagle_GPIO.cpp.

6.3.3.2 Beagle_GPIO::~Beagle_GPIO ( )

Definition at line 172 of file Beagle_GPIO.cpp.

6.3.4 Member Function Documentation

6.3.4.1 void Beagle_GPIO::closeSPI ( )

Definition at line 363 of file Beagle_GPIO.cpp.

6.3.4.2 Beagle_GPIO::Beagle_GPIO_Status Beagle_GPIO::configurePin (unsigned short *_pin,* Beagle_GPIO_Direction *_direction*)

Definition at line 183 of file Beagle_GPIO.cpp.

6.3.4.3 Beagle_GPIO::Beagle_GPIO_Status Beagle_GPIO::enablePinInterrupts (unsigned short *_pin,* bool *_enable*)

Definition at line 216 of file Beagle_GPIO.cpp.

6.3.4.4 bool Beagle_GPIO::isActive ( ) [inline]

Definition at line 165 of file Beagle_GPIO.h.

6.3.4.5 void Beagle_GPIO::openSPI (unsigned char *_mode* = `0`, unsigned char *_bits* = `8`, unsigned long *_speed* = `4800000`, unsigned short *_delay* = `0`)

Definition at line 284 of file Beagle_GPIO.cpp.

6.3.4.6 unsigned char Beagle_GPIO::readPin (unsigned short *_pin*)

Definition at line 268 of file Beagle_GPIO.cpp.

6.3.4.7 void Beagle_GPIO::sendSPIBuffer (unsigned long *buffer,* int *size*)

Definition at line 377 of file Beagle_GPIO.cpp.

6.3.4.8 Beagle_GPIO::Beagle_GPIO_Status Beagle_GPIO::writePin (unsigned short *_pin,* unsigned char *_value*)

Definition at line 248 of file Beagle_GPIO.cpp.

6.3.5 Member Data Documentation

6.3.5.1 enum { ... } Beagle_GPIO::Beagle_GPIO_Registers

6.3.5.2 const unsigned long Beagle_GPIO::GPIO_Base [static]

Initial value:

```
{
        0x44E07000,
        0x4804C000,
        0x481AC000,
        0x481AE000
}
```

Definition at line 133 of file Beagle_GPIO.h.

6.3.5.3 const unsigned long Beagle_GPIO::GPIO_Control_Module_Registers = 0x44E10000 [static]

Definition at line 130 of file Beagle_GPIO.h.

6.3.5.4 const unsigned long Beagle_GPIO::GPIO_Pad_Control [static]

Initial value:

```
{
        0x0000, 0x0000, 0x0818, 0x081C, 0x0808,
        0x080C, 0x0890, 0x0894, 0x089C, 0x0898,
        0x0834, 0x0830, 0x0824, 0x0828, 0x083C,
        0x0838, 0x082C, 0x088C, 0x0820, 0x0884,
        0x0880, 0x0814, 0x0810, 0x0804, 0x0800,
        0x087C, 0x08E0, 0x08E8, 0x08E4, 0x08EC,
        0x08D8, 0x08DC, 0x08D4, 0x08CC, 0x08D0,
        0x08C8, 0x08C0, 0x08C4, 0x08B8, 0x08BC,
        0x08B0, 0x08B4, 0x08A8, 0x08AC, 0x08A0,
        0x08A4,
        0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
        0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
        0x0870, 0x0878, 0x0874, 0x0848, 0x0840,
        0x084C, 0x095C, 0x0958, 0x097C, 0x0978,
        0x0954, 0x0950, 0x0844, 0x0984, 0x09AC,
        0x0980, 0x09A4, 0x099C, 0x0994, 0x0998,
        0x0990, 0x0000, 0x0000, 0x0000, 0x0000,
        0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
        0x09B4, 0x0964, 0x0000, 0x0000, 0x0000,
        0x0000
}
```

Definition at line 127 of file Beagle_GPIO.h.

6.3.5.5 const int Beagle_GPIO::GPIO_Pin_Bank [static]

Initial value:

```
{
        -1, -1,  1,  1,  1,
         1,  2,  2,  2,  2,
         1,  1,  0,  0,  1,
         1,  0,  2,  0,  1,
         1,  1,  1,  1,  1,
         1,  2,  2,  2,  2,
         0,  0,  0,  2,  0,
         2,  2,  2,  2,  2,
         2,  2,  2,  2,  2,
         2,
        -1, -1, -1, -1, -1,
        -1, -1, -1, -1, -1,
         0,  1,  0,  1,  1,
         1,  0,  0,  0,  0,
         0,  0,  1,  0,  3,
         0,  3,  3,  3,  3,
         3, -1, -1, -1, -1,
        -1, -1, -1, -1, -1,
         0,  0, -1, -1, -1,
        -1
}
```

Definition at line 121 of file Beagle_GPIO.h.

6.3.5.6 const int Beagle_GPIO::GPIO_Pin_Id [static]

Initial value:

```
{
        -1, -1,  6,  7,  2,
         3,  2,  3,  5,  4,
        13, 12, 23, 26, 15,
        14, 27,  1, 22, 31,
        30,  5,  4,  1,  0,
        29, 22, 24, 23, 25,
        10, 11,  9, 17,  8,
        16, 14, 15, 12, 13,
        10, 11,  8,  9,  6,
         7,
        -1, -1, -1, -1, -1,
        -1, -1, -1, -1, -1,
        30, 28, 31, 18, 16,
        19,  5,  4, 13, 12,
         3,  2, 17, 15, 21,
        14, 19, 17, 15, 16,
        14, -1, -1, -1, -1,
        -1, -1, -1, -1, -1,
        20,  7, -1, -1, -1,
        -1
}
```

Definition at line 124 of file Beagle_GPIO.h.

6.3.5.7 enum { ... } Beagle_GPIO::GPIO_Pins

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

- pwm/Beagle_GPIO.h
- pwm/Beagle_GPIO.cpp

6.4 cPWM::cPWM Class Reference

`#include <cPWM.h>`

Public Types

- enum Polarity { ActiveHigh, ActiveLow }

Public Member Functions

- cPWM (int id)
- virtual ~cPWM ()
- void DutyA_ns (unsigned int nanoseconds)
- void DutyA_percent (unsigned int percent)
- void DutyB_ns (unsigned int nanoseconds)
- void DutyB_percent (unsigned int percent)
- void Period_ns (unsigned int nanoseconds)
- void Period_freq (unsigned int freq_Hz)
- void PolarityA (cPWM::Polarity polarity)
- void RunA ()
- void StopA ()
- void PolarityB (cPWM::Polarity polarity)
- void RunB ()
- void StopB ()

6.4.1 Detailed Description

Definition at line 20 of file cPWM.h.

6.4.2 Member Enumeration Documentation

6.4.2.1 enum cPWM::cPWM::Polarity

Enumerator:

ActiveHigh

ActiveLow

Definition at line 23 of file cPWM.h.

6.4.3 Constructor & Destructor Documentation

6.4.3.1 cPWM::cPWM::cPWM (int *id*)

This class wraps the PWMss of the beaglebone, but it accesses the PWMss by means of the sysfs interface, so probably other systems are supported as well. The sysfs filenames are defined in cPWM.h. The constructor just opens the sysfs files but doesn't write anything, so in order to properly use the PWMss you need to follow all the steps (frequency, period, polarity) before calling run.

Parameters

in	*id*	id of the PWMss to be initializaed. There are 3 of them, eHRPWM0 thru 2.

Returns

a cPWM object

TODO: Add clock selection (mmap). By now you must use setPWMReg.py method
FIXME: pin mux settings should be done here? or at a highet level?

Definition at line 36 of file cPWM.cpp.

6.4.3.2 cPWM::cPWM::~cPWM ( ) [virtual]

cPWM Destructor, stops the PWMss

Definition at line 264 of file cPWM.cpp.

6.4.4 Member Function Documentation

6.4.4.1 void cPWM::cPWM::DutyA_ns (unsigned int *nanoseconds*)

Set the duty cycle for A channel of the PWMss

Parameters

in	*nanoseconds,:*	duty cycle time in nanoseconds for A channel

Definition at line 101 of file cPWM.cpp.

6.4.4.2 void cPWM::cPWM::DutyA_percent (unsigned int *percent*)

Set the duty cycle for A channel of the PWMss

Parameters

in	*percent,:*	duty cycle time in percent for A channel

Definition at line 116 of file cPWM.cpp.

6.4.4.3 void cPWM::cPWM::DutyB_ns (unsigned int *nanoseconds*)

Set the duty cycle for B channel of the PWMss

Parameters

in	*nanoseconds,:*	duty cycle time in nanoseconds for B channel

Definition at line 130 of file cPWM.cpp.

6.4.4.4 void cPWM::cPWM::DutyB_percent (unsigned int *percent*)

Set the duty cycle for B channel of the PWMss

Parameters

in	*percent,:*	duty cycle time in percent for B channel

Definition at line 146 of file cPWM.cpp.

6.4.4.5 void cPWM::cPWM::Period_freq (unsigned int *freq_Hz*)

Set the period for the PWMss

Parameters

in	*freq_Hz,:*	PWM frequency in Hz

Definition at line 174 of file cPWM.cpp.

6.4.4.6 void cPWM::cPWM::Period_ns (unsigned int *nanoseconds*)

Set the period for the PWMss

Parameters

in	*nanoseconds* :	period time in nanoseconds -

Definition at line 161 of file cPWM.cpp.

6.4.4.7 void cPWM::cPWM::PolarityA (cPWM::Polarity *polarity*)

Set the polarity for the A channel of the PWMss

Parameters

in	*polarity*	polarity

Definition at line 187 of file cPWM.cpp.

6.4.4.8 void cPWM::cPWM::PolarityB (cPWM::Polarity *polarity*)

Set the polarity for the B channel of the PWMss

Parameters

in	*polarity*	polarity

Definition at line 227 of file cPWM.cpp.

6.4.4.9 void cPWM::cPWM::RunA ( )

Set the A channel to run status

Definition at line 204 of file cPWM.cpp.

6.4.4.10 void cPWM::cPWM::RunB ( )

Set the B channel to run

Definition at line 244 of file cPWM.cpp.

6.4.4.11 void cPWM::cPWM::StopA ( )

Stop the A channel

Definition at line 215 of file cPWM.cpp.

6.4.4.12 void cPWM::cPWM::StopB ( )

Stop the B channel

Definition at line 254 of file cPWM.cpp.

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

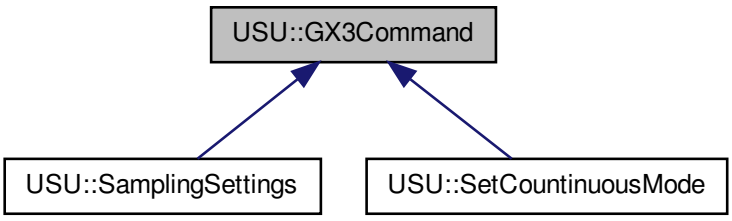
- pwm/cPWM.h
- pwm/cPWM.cpp

6.5 USU::GX3Command Class Reference

Base class for commands send to the 3DM-GX3-25.

`#include <messages.h>`

Inheritance diagram for USU::GX3Command:



6.5.1 Detailed Description

Base class for commands send to the 3DM-GX3-25.

Just an empty base class, so that all commands share the same base class.

Definition at line 254 of file messages.h.

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

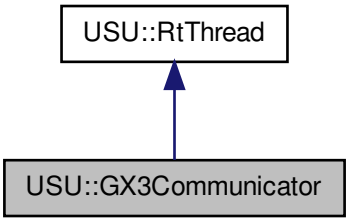
- 3dm/messages.h

6.6 USU::GX3Communicator Class Reference

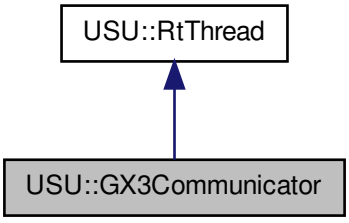
Represents the Thread class for communication with the 3DM-GX3-25.

`#include <gx3communicator.h>`

Inheritance diagram for USU::GX3Communicator:



Collaboration diagram for USU::GX3Communicator:



Public Member Functions

- GX3Communicator (int priority, const char *serialDevice, SerialPort::BaudRate baudRate=SerialPort::BAUD_115200)

  Constructor of the class.

- virtual void run ()

  Thread routine.
- void stop ()

  Signals the thread to stop.

6.6.1 Detailed Description

Represents the Thread class for communication with the 3DM-GX3-25.

The class is derived from RtThread. It initializes the serial interface to the 3DM and sets the sampling settings. Finally it starts the continuous mode and polls the serial port for new arrived data. If new data arrived it is presented for the KalmanFilter to take and consider.

Definition at line 30 of file gx3communicator.h.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 GX3Communicator::GX3Communicator (int *priority*, const char * *serialDevice*, SerialPort::BaudRate *baudRate* = `SerialPort::BAUD_115200`)

Constructor of the class.

Sets up the serial port and thread attributes.

Parameters

priority	Priority of the pthread (1..99)
serialDevice	Name of the serial device
baudRate	Baud rate for the serial device (if different from 115200)

Definition at line 46 of file gx3communicator.cpp.

6.6.3 Member Function Documentation

6.6.3.1 void GX3Communicator::run ( ) `[virtual]`

Thread routine.

- Set sampling settings of 3DM
- Start continuous mode
- Poll serial port for newly arrived packages
- Convert binary data
- TODO: Send new package to KalmanFilter

Implements USU::RtThread.

Definition at line 73 of file gx3communicator.cpp.

6.6.3.2 void USU::GX3Communicator::stop ( ) [inline]

Signals the thread to stop.

Definition at line 60 of file gx3communicator.h.

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

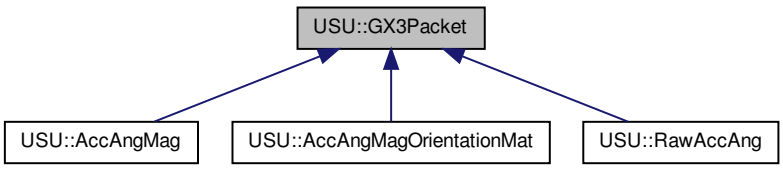
- 3dm/gx3communicator.h
- 3dm/gx3communicator.cpp

6.7 USU::GX3Packet Class Reference

Abstract base class for received packets.

`#include <messages.h>`

Inheritance diagram for USU::GX3Packet:



Static Public Member Functions

- static bool calculateChecksum (uint8_t *buffer, unsigned int length)

  Calculates the checksum of a received byte array.
- static vector createVector (uint8_t *buffer)

  Creates a Eigen::Vector3f consisting of 3 floats from 12 sucessive bytes.
- static unsigned int createUInt (uint8_t *buffer)

  Creates an unsigned integer from 4 successive bytes.
- static void createMatrix (uint8_t *buffer, matrix &mat)

  Creates a Eigen::Matrix3f from byte array.

6.7.1 Detailed Description

Abstract base class for received packets.

The class provides some useful function available to all derived classes such as checksum calculation and creation of vectors and matrizes from the received binary data.

Definition at line 69 of file messages.h.

6.7.2 Member Function Documentation

6.7.2.1 static bool USU::GX3Packet::calculateChecksum (uint8_t * *buffer*, unsigned int *length*) [inline, static]

Calculates the checksum of a received byte array.

Parameters

buffer	pointer to the byte array
length	length of the byte array

Returns

bool true: checksum matches, false: checksum does not match

Definition at line 79 of file messages.h.

6.7.2.2 static void USU::GX3Packet::createMatrix (uint8_t * *buffer*, matrix & *mat*) [inline, static]

Creates a Eigen::Matrix3f from byte array.

NOTE: Make sure that the endianess of the host system and the 3DM match. The endianess of the sent floats can be set with the SamplingSettings command.

Parameters

buffer	Pointer to the byte array
mat	reference to a matrix which will be filled with the data from the byte array

Definition at line 126 of file messages.h.

6.7.2.3 static unsigned int USU::GX3Packet::createUInt (uint8_t * *buffer*) [inline, static]

Creates an unsigned integer from 4 successive bytes.

Parameters

buffer	Pointer to the byte array

Returns

unsigned int created unsigned integer

Definition at line 112 of file messages.h.

6.7.2.4 static vector USU::GX3Packet::createVector (uint8_t * *buffer*) [inline, static]

Creates a Eigen::Vector3f consisting of 3 floats from 12 sucessive bytes.

NOTE: Make sure that the endianess of the host system and the 3DM match. The endianess of the sent floats can be set with the SamplingSettings command.

Parameters

buffer	Pointer to the byte array

Returns

vector vector created from the byte array

Definition at line 99 of file messages.h.

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

- 3dm/messages.h

6.8 I2CBus Class Reference

`#include <I2CBus.h>`

Public Member Functions

- I2CBus (const char *deviceName)
- ~I2CBus ()
- void addressSet (uint8_t address)
- void writeByte (uint8_t command, uint8_t data)
- uint8_t readByte (uint8_t command)
- int tryReadByte (uint8_t command)
- void readBlock (uint8_t command, uint8_t size, uint8_t *data)

6.8.1 Detailed Description

Definition at line 7 of file I2CBus.h.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 I2CBus::I2CBus (const char * *deviceName*)

Definition at line 7 of file I2CBus.cpp.

6.8.2.2 I2CBus::~I2CBus ( )

Definition at line 16 of file I2CBus.cpp.

6.8.3 Member Function Documentation

6.8.3.1 void I2CBus::addressSet (uint8_t *address*)

Definition at line 21 of file I2CBus.cpp.

6.8.3.2 void I2CBus::readBlock (uint8_t *command*, uint8_t *size*, uint8_t * *data*)

Definition at line 54 of file I2CBus.cpp.

6.8.3.3 uint8_t I2CBus::readByte (uint8_t *command*)

Definition at line 39 of file I2CBus.cpp.

6.8.3.4 int I2CBus::tryReadByte (uint8_t *command*)

Definition at line 49 of file I2CBus.cpp.

6.8.3.5 void I2CBus::writeByte (uint8_t *command*, uint8_t *data*)

Definition at line 30 of file I2CBus.cpp.

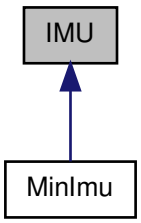
The documentation for this class was generated from the following files:

- minimu/I2CBus.h
- minimu/I2CBus.cpp

6.9 IMU Class Reference

`#include <IMU.h>`

Inheritance diagram for IMU:



Public Member Functions

- virtual vector readMag ()=0
- virtual vector readAcc ()=0
- virtual vector readGyro ()=0
- void read ()
- virtual void enable ()=0

Public Attributes

- int_vector raw_m
- int_vector raw_a
- int_vector raw_g

6.9.1 Detailed Description

Definition at line 6 of file IMU.h.

6.9.2 Member Function Documentation

6.9.2.1 virtual void IMU::enable ( ) [pure virtual]

Implemented in MinImu.

6.9.2.2 void IMU::read ( ) [inline]

Definition at line 12 of file IMU.h.

6.9.2.3 virtual vector IMU::readAcc ( ) [pure virtual]

Implemented in MinImu.

6.9.2.4 virtual vector IMU::readGyro ( ) [pure virtual]

Implemented in MinImu.

6.9.2.5 virtual vector IMU::readMag ( ) [pure virtual]

Implemented in MinImu.

6.9.3 Member Data Documentation

6.9.3.1 int_vector IMU::raw_a

Definition at line 22 of file IMU.h.

6.9.3.2 int_vector IMU::raw_g

Definition at line 22 of file IMU.h.

6.9.3.3 int_vector IMU::raw_m

Definition at line 22 of file IMU.h.

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

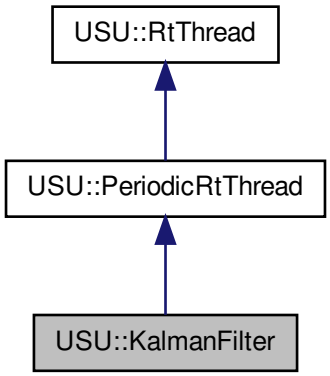
- minimu/IMU.h

6.10 USU::KalmanFilter Class Reference

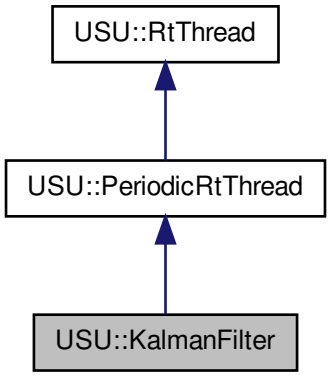
Represents the Periodic Thread class for state estimation.

`#include <kalmanfilter.h>`

Inheritance diagram for USU::KalmanFilter:



Collaboration diagram for USU::KalmanFilter:



Public Member Functions

- KalmanFilter (int priority, unsigned int period_us, char *i2cBus)

  Constructor of the class.

- virtual void run ()

  Thread routine.
- void stop ()

  Signals the thread to stop.
- bool getState () const

  Returns the current system state estimate.

6.10.1 Detailed Description

Represents the Periodic Thread class for state estimation.

This class is derived from PeriodicRtThread. It initializes the interface to the MinIMU9v2 and estimates the system state using Kalman filtering techniques. The state estimate can be accessed from other threads (protected by mutex).

TODO:

- Add interface to 3DM-GX3-25
- Add interface to star camera

Definition at line 35 of file kalmanfilter.h.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 KalmanFilter::KalmanFilter (int *priority*, unsigned int *period_us*, char * *i2cBus*)

Constructor of the class.

Initializes the interface to the MinIMU9 sensors

Parameters

priority	priority of the underlying periodic thread
period_us	period (in us) of the underlying periodic thread
i2cBus	name of the I2C-device (e.g. /dev/i2c-1)

Definition at line 16 of file kalmanfilter.cpp.

6.10.3 Member Function Documentation

6.10.3.1 bool KalmanFilter::getState ( ) const

Returns the current system state estimate.

Copies the current system state estimate. Acquires mutex before acessing the internal variable to avoid read/write-conflicts.

Returns

bool Current system state TODO: Currently only dummy variable. Replace with actual state representation (quaternion?)

Definition at line 45 of file kalmanfilter.cpp.

6.10.3.2 void KalmanFilter::run ( ) [virtual]

Thread routine.

- Gets sensor data from MinIMU9
- Calculate state estimate
- wait for next timer event

TODO: Its only an idea no actual implementation yet. TODO: Do some Kalman-Filtering magic here

Implements USU::PeriodicRtThread.

Definition at line 21 of file kalmanfilter.cpp.

6.10.3.3 void USU::KalmanFilter::stop ( ) [inline]

Signals the thread to stop.

Definition at line 64 of file kalmanfilter.h.

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

- kalmanfilter.h
- kalmanfilter.cpp

6.11 L3G Class Reference

`#include <L3G.h>`

Public Member Functions

- L3G (const char *i2cDeviceName)
- void enable (void)
- void writeReg (uint8_t reg, uint8_t value)
- uint8_t readReg (uint8_t reg)
- void read ()

Public Attributes

- int g [3]

6.11.1 Detailed Description

Definition at line 37 of file L3G.h.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 L3G::L3G (const char * *i2cDeviceName*)

Definition at line 9 of file L3G.cpp.

6.11.3 Member Function Documentation

6.11.3.1 void L3G::enable (void)

Definition at line 29 of file L3G.cpp.

6.11.3.2 void L3G::read ( )

Definition at line 45 of file L3G.cpp.

6.11.3.3 uint8_t L3G::readReg (uint8_t *reg*)

Definition at line 40 of file L3G.cpp.

6.11.3.4 void L3G::writeReg (uint8_t *reg*, uint8_t *value*)

Definition at line 35 of file L3G.cpp.

6.11.4 Member Data Documentation

6.11.4.1 int L3G::g[3]

Definition at line 43 of file L3G.h.

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

- minimu/L3G.h
- minimu/L3G.cpp

6.12 USU::Lock Class Reference

Wrapper class for pthread mutexes.

`#include <Lock.h>`

Public Member Functions

- Lock ()
- virtual ~Lock ()
- void lock ()
- void unlock ()

6.12.1 Detailed Description

Wrapper class for pthread mutexes.

Definition at line 22 of file Lock.h.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 USU::Lock::Lock ( ) [inline]

Constructor: Creates the pthread-mutex

Definition at line 45 of file Lock.h.

6.12.2.2 USU::Lock::~Lock ( ) [inline, virtual]

Destructor: Frees the pthread-mutex

Definition at line 55 of file Lock.h.

6.12.3 Member Function Documentation

6.12.3.1 void USU::Lock::lock ( ) [inline]

Locks the mutex

Definition at line 66 of file Lock.h.

6.12.3.2 void USU::Lock::unlock ( ) [inline]

Unlocks the mutex

Definition at line 72 of file Lock.h.

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

- threading/Lock.h

6.13 LSM303 Class Reference

`#include <LSM303.h>`

Public Member Functions

- LSM303 (const char *i2cDeviceName)
- void enable (void)
- void writeAccReg (uint8_t reg, uint8_t value)
- uint8_t readAccReg (uint8_t reg)
- void writeMagReg (uint8_t reg, uint8_t value)
- uint8_t readMagReg (uint8_t reg)
- void readAcc (void)
- void readMag (void)
- void read (void)

Public Attributes

- int a [3]
- int m [3]

6.13.1 Detailed Description

Definition at line 78 of file LSM303.h.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 LSM303::LSM303 (const char * *i2cDeviceName*)

Definition at line 22 of file LSM303.cpp.

6.13.3 Member Function Documentation

6.13.3.1 void LSM303::enable (void)

Definition at line 73 of file LSM303.cpp.

6.13.3.2 void LSM303::read (void)

Definition at line 118 of file LSM303.cpp.

6.13.3.3 void LSM303::readAcc (void)

Definition at line 93 of file LSM303.cpp.

6.13.3.4 uint8_t LSM303::readAccReg (uint8_t *reg*)

Definition at line 56 of file LSM303.cpp.

6.13.3.5 void LSM303::readMag (void)

Definition at line 103 of file LSM303.cpp.

6.13.3.6 uint8_t LSM303::readMagReg (uint8_t *reg*)

Definition at line 51 of file LSM303.cpp.

6.13.3.7 void LSM303::writeAccReg (uint8_t *reg*, uint8_t *value*)

Definition at line 66 of file LSM303.cpp.

6.13.3.8 void LSM303::writeMagReg (uint8_t *reg*, uint8_t *value*)

Definition at line 61 of file LSM303.cpp.

6.13.4 Member Data Documentation

6.13.4.1 int LSM303::a[3]

Definition at line 81 of file LSM303.h.

6.13.4.2 int LSM303::m[3]

Definition at line 82 of file LSM303.h.

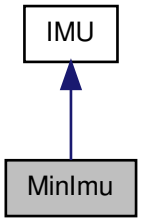
The documentation for this class was generated from the following files:

- minimu/LSM303.h
- minimu/LSM303.cpp

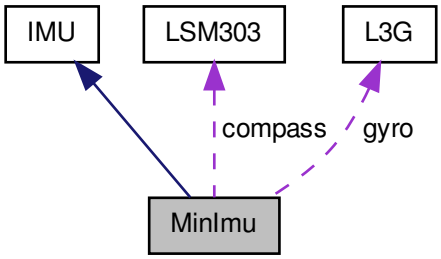
6.14 MinImu Class Reference

```
#include <minimu.h>
```

Inheritance diagram for MinImu:



Collaboration diagram for MinImu:



Public Member Functions

- MinImu (const char *i2cDeviceName)
- virtual vector readMag ()
- virtual vector readAcc ()
- virtual vector readGyro ()
- virtual void enable ()

Public Attributes

- LSM303 compass
- L3G gyro

6.14.1 Detailed Description

Definition at line 9 of file minimu.h.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 MinImu::MinImu (const char * *i2cDeviceName*)

Definition at line 4 of file minimu.cpp.

6.14.3 Member Function Documentation

6.14.3.1 void MinImu::enable (void) [virtual]

Implements IMU.

Definition at line 10 of file minimu.cpp.

6.14.3.2 vector MinImu::readAcc (void) [virtual]

Implements IMU.

Definition at line 27 of file minimu.cpp.

6.14.3.3 vector MinImu::readGyro ( ) [virtual]

Implements IMU.

Definition at line 16 of file minimu.cpp.

6.14.3.4 vector MinImu::readMag (void) [virtual]

Implements IMU.

Definition at line 38 of file minimu.cpp.

6.14.4 Member Data Documentation

6.14.4.1 LSM303 MinImu::compass

Definition at line 12 of file minimu.h.

6.14.4.2 L3G MinImu::gyro

Definition at line 13 of file minimu.h.

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

- minimu/minimu.h
- minimu/minimu.cpp

6.15 USU::Motor Class Reference

`#include <motor.h>`

Public Member Functions

- Motor (Beagle_GPIO &beagleGpio, Beagle_GPIO::GPIO_Pins clockwise, Beagle_GPIO::GPIO_Pins counterClockwise, SetDutyCyle dutyCycle)
- void setSpeed (int speed)
- int getSpeed () const

6.15.1 Detailed Description

Definition at line 23 of file motor.h.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 Motor::Motor (Beagle_GPIO & *beagleGpio,* Beagle_GPIO::GPIO_Pins *clockwise,* Beagle_GPIO::GPIO_Pins *counterClockwise,* SetDutyCyle *dutyCycle*)

Definition at line 14 of file motor.cpp.

6.15.3 Member Function Documentation

6.15.3.1 int USU::Motor::getSpeed ( ) const [inline]

Definition at line 29 of file motor.h.

6.15.3.2 void Motor::setSpeed (int *speed*)

Definition at line 29 of file motor.cpp.

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

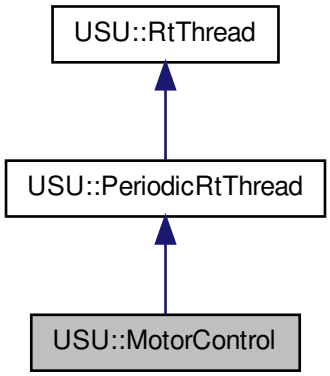
- pwm/motor.h
- pwm/motor.cpp

6.16 USU::MotorControl Class Reference

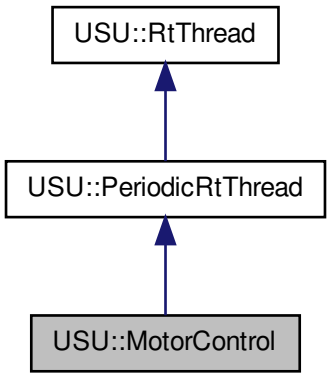
Represents the Periodic task for motor control.

`#include <motorcontrol.h>`

Inheritance diagram for USU::MotorControl:



Collaboration diagram for USU::MotorControl:



Public Member Functions

- MotorControl (int priority=0, unsigned int period_us=1000000, KalmanFilter &kalmanfilter)

  Constructor of the class.
- void stop ()

  Signals the thread to stop.
- virtual void run ()

  Thread routine.

6.16.1 Detailed Description

Represents the Periodic task for motor control.

This class is derived from PeriodicRtThread. It initializes the interface to the 4 motors. In a periodic loop it takes the last system state estimate from the Kalman filter, calculates the appropiate control response and sets the speed (duty cycle) of the motors.

Definition at line 34 of file motorcontrol.h.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 MotorControl::MotorControl (int *priority* = `0`, unsigned int *period_us* = `1000000`, KalmanFilter & *kalmanfilter*)

Constructor of the class.

Initializes the underlying PeriodicRtThread, the GPIO-class, the PWMs and the 4 - Motors.

Parameters

priority	priority of the periodic pthread
period_us	period (in us) of the periodic pthread
kalmanfilter	reference to the KalmanFilter to get state estimates

TODO: use meaningful Pin numbers (declare consts)

Definition at line 16 of file motorcontrol.cpp.

6.16.3 Member Function Documentation

6.16.3.1 void MotorControl::run ( ) `[virtual]`

Thread routine.

- Gets the newest estimate from KalmanFilter
- Calculate the control response

- Set the motor speed of the 4 Motors
- wait for the next timer event TODO: Its only an idea, no actual implementation yet.

TODO: Make some control magic

[...]

Implements USU::PeriodicRtThread.

Definition at line 27 of file motorcontrol.cpp.

6.16.3.2 void USU::MotorControl::stop ( ) [inline]

Signals the thread to stop.

Definition at line 55 of file motorcontrol.h.

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

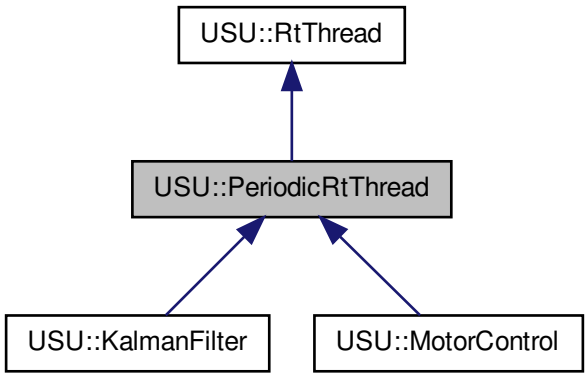
- motorcontrol.h
- motorcontrol.cpp

6.17 USU::PeriodicRtThread Class Reference

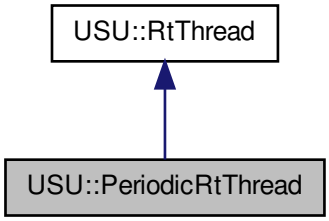
TODO: Make some proper exceptions.

`#include <periodicrtthread.h>`

Inheritance diagram for USU::PeriodicRtThread:



Collaboration diagram for USU::PeriodicRtThread:



Public Member Functions

- PeriodicRtThread (int priority=0, unsigned int period_us=1000000)

  Creates the PeriodicRtThread object.
- virtual void run ()=0

  Actual method of the thread is running.

Protected Member Functions

- void makeThreadPeriodic ()

  Registers the Periodic timer.
- void waitPeriod ()

  Blocks the thread until the next timer event.

6.17.1 Detailed Description

TODO: Make some proper exceptions.

Abstract wrapper class for a periodic thread usign the pthread library with RT-priority

Based on RtThread this class uses pthread underneath but creates a periodic timer event it can wait for in a (forever) loop. This is more accurate than the use of nanosleep as the execution time of the loop will not be taken into account. It is therefore designed for periodic work where high accuracy is desired.

Definition at line 30 of file periodicrtthread.h.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 PeriodicRtThread::PeriodicRtThread (int *priority* = 0, unsigned int *period_us* = 1000000)

Creates the PeriodicRtThread object.

Calls the constructor of the parent RtThread and registers the periodic timer

Parameters

priority	the Priority of the Thread (Linux: 1..99)
period_us	Period of the thread in us

Definition at line 20 of file periodicrtthread.cpp.

6.17.3 Member Function Documentation

6.17.3.1 void PeriodicRtThread::makeThreadPeriodic ( ) [protected]

Registers the Periodic timer.

TODO: create exception

Definition at line 27 of file periodicrtthread.cpp.

6.17.3.2 virtual void USU::PeriodicRtThread::run ( ) [pure virtual]

Actual method of the thread is running.

Every child class has to implement this function in order to do some threaded work.

Implements USU::RtThread.

Implemented in USU::MotorControl, and USU::KalmanFilter.

6.17.3.3 void PeriodicRtThread::waitPeriod ( ) [protected]

Blocks the thread until the next timer event.

Waits the remaining time until the next timer event happens. Thus waitTime = mPeriod-_us - runtime since last timer event

Definition at line 54 of file periodicrtthread.cpp.

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

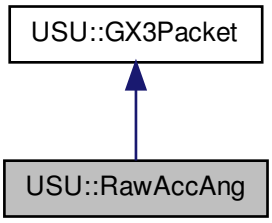
- threading/periodicrtthread.h
- threading/periodicrtthread.cpp

6.18 USU::RawAccAng Class Reference

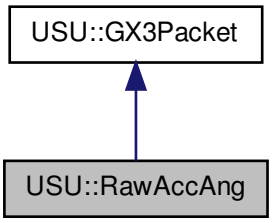
Representation for receiving (raw) acceleration & angular rate packets.

`#include <messages.h>`

Inheritance diagram for USU::RawAccAng:



Collaboration diagram for USU::RawAccAng:



Public Types

- enum { size = 31 }

Public Member Functions

- RawAccAng (uint8_t *buffer)

  Creates a packet object from the passed buffer.

Public Attributes

- vector acc
- vector gyro
- unsigned int timer

6.18.1 Detailed Description

Representation for receiving (raw) acceleration & angular rate packets.

This class can be used with the commands for raw acceleration and angular rates and acceleration and angular rate. For the latter the units are:

- acceleration: g
- angular rate: rad/s For the units of the raw values see the protocol data sheet.

Definition at line 144 of file messages.h.

6.18.2 Member Enumeration Documentation

6.18.2.1 anonymous enum

Enumerator:

size

Definition at line 168 of file messages.h.

6.18.3 Constructor & Destructor Documentation

6.18.3.1 USU::RawAccAng::RawAccAng (uint8_t * *buffer*) [inline]

Creates a packet object from the passed buffer.

The checksum should have been tested before.

Parameters

buffer	pointer to the byte array containing the received data

Definition at line 154 of file messages.h.

6.18.4 Member Data Documentation

6.18.4.1 vector USU::RawAccAng::acc

Vector containing the accelerometer data

Definition at line 163 of file messages.h.

6.18.4.2 vector USU::RawAccAng::gyro

Vector containing the gyroscope (angular rate) data

Definition at line 164 of file messages.h.

6.18.4.3 unsigned int USU::RawAccAng::timer

The value of the timestamp for the package

Definition at line 166 of file messages.h.

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

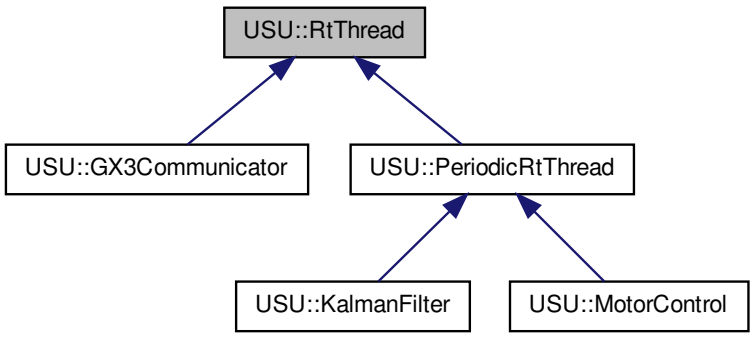
- 3dm/messages.h

6.19 USU::RtThread Class Reference

Abstract wrapper class for the pthread library with RT-priority.

`#include <RtThread.h>`

Inheritance diagram for USU::RtThread:



Public Member Functions

- RtThread (int priority=0)

  Creates the RtThread object.

- virtual ~RtThread ()

  Destructor of the RtThread object.
- pthread_t getThreadId () const

  Return the pthread handle.
- int getPriority () const

  Returns the priority of the thread.
- void start (void *args=NULL)

  Creates and starts the pthread.
- void join ()

  Waits for the thread to join.
- virtual void run ()=0

  Actual method of the thread is running.

Static Protected Member Functions

- static void * exec (void *thr)

  Function passed to pthread_create, do not call manually!

Protected Attributes

- pthread_t mId
- bool mStarted
- void * mArgs

6.19.1 Detailed Description

Abstract wrapper class for the pthread library with RT-priority.

This class is a thin wrapper for the pthread library. Inherited classes need to implement the run function with the tasks for the thread. The thread will run with the SCHED_FIFO-scheduler at the set priority. Therefore root rights are necessary for changing the scheduling policy.

Definition at line 29 of file RtThread.h.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 RtThread::RtThread (int *priority* = 0)

Creates the RtThread object.

Prepares the Attribute object which is passed to pthread_create later.

Parameters

priority	the Priority of the Thread (Linux: 1..99)

Definition at line 17 of file RtThread.cpp.

6.19.2.2 RtThread::~RtThread ( ) [virtual]

Destructor of the RtThread object.

Waits for the thread to join (if not already) and releases the Attributes object.

Definition at line 60 of file RtThread.cpp.

6.19.3 Member Function Documentation

6.19.3.1 void * RtThread::exec (void * *thr*) [static, protected]

Function passed to pthread_create, do not call manually!

This function builds the interface to the pthread library. Only purpose is to be compatible to pthread_create, as it will immediately call run of this class.

Parameters

thr	pointer to this instance of the class.

Definition at line 118 of file RtThread.cpp.

6.19.3.2 int RtThread::getPriority ( ) const [inline]

Returns the priority of the thread.

Returns

int priority

Definition at line 82 of file RtThread.cpp.

6.19.3.3 pthread_t RtThread::getThreadId ( ) const [inline]

Return the pthread handle.

Returns

pthread_t the thread handle of the last started pthread or -1 (if no pthread was started)

Definition at line 76 of file RtThread.cpp.

6.19.3.4 void RtThread::join ( )

Waits for the thread to join.

Definition at line 108 of file RtThread.cpp.

6.19.3.5 virtual void USU::RtThread::run ( ) [pure virtual]

Actual method of the thread is running.

Every child class has to implement this function in order to do some threaded work.

Implemented in USU::PeriodicRtThread, USU::MotorControl, USU::KalmanFilter, and USU::GX3Communicator.

6.19.3.6 void RtThread::start (void * *args* = NULL)

Creates and starts the pthread.

Creates the pthread with the desired attributes.

Parameters

args	optional arguments for the thread

Definition at line 87 of file RtThread.cpp.

6.19.4 Member Data Documentation

6.19.4.1 void* USU::RtThread::mArgs [protected]

Arguments which can be passed to a certain thread thread

Definition at line 42 of file RtThread.h.

6.19.4.2 pthread_t USU::RtThread::mId [protected]

The thread handle

Definition at line 40 of file RtThread.h.

6.19.4.3 bool USU::RtThread::mStarted [protected]

Keeps the status of the thread TODO: Useful??

Definition at line 41 of file RtThread.h.

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

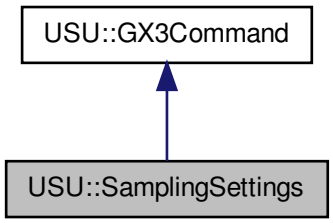
- threading/RtThread.h
- threading/RtThread.cpp

6.20 USU::SamplingSettings Class Reference

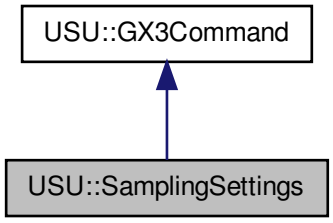
Represents the "Sampling Settings" command.

`#include <messages.h>`

Inheritance diagram for USU::SamplingSettings:



Collaboration diagram for USU::SamplingSettings:



Public Types

- enum FunctionSelector { ReturnOnly = 0, Change = 1, ChangeAndSave = 2, ChangeWithoutReply = 3 }

  Sets the function Selector.
- enum DataConditioning { FlagCalcOrientation = 0x01, FlagEnableConingSculling = 0x02, FlagDefault = 0x03, FlagFloatLittleEndian = 0x10, FlagSuppressNaN = 0x20, FlagFiniteSizeCorrection = 0x40, FlagDisableMag =

0x100, FlagDisableMagNorthComp = 0x400, FlagDisableGravComp = 0x800, FlagEnableQuaternion = 0x1000 }

Flags for the Data conditioning.

- enum { size = 20, responseSize = 19 }

Public Member Functions

- SamplingSettings (FunctionSelector funSel, uint16_t samplingPeriod_ms=10, uint16_t dataCondFlags=SamplingSettings::FlagDefault, uint8_t gyroAccFilter=15, uint8_t magFilter=17, uint16_t upCompensation=10, uint16_t northCompensation=10, uint8_t magPower=0)

  Creates the command.
- bool checkResponse (uint8_t *buffer, unsigned int length)

  Checks if the response to this command has the correct setup.

Public Attributes

- uint8_t mCommand [size]

6.20.1 Detailed Description

Represents the "Sampling Settings" command.

Definition at line 311 of file messages.h.

6.20.2 Member Enumeration Documentation

6.20.2.1 anonymous enum

Enumerator:

size

responseSize

Definition at line 419 of file messages.h.

6.20.2.2 enum USU::SamplingSettings::DataConditioning

Flags for the Data conditioning.

Sets the bits for Data conditioning bytes. Combine multiple flags using the "or" operator ("|")

Enumerator:

FlagCalcOrientation

FlagEnableConingSculling
FlagDefault
FlagFloatLittleEndian
FlagSuppressNaN
FlagFiniteSizeCorrection
FlagDisableMag
FlagDisableMagNorthComp
FlagDisableGravComp
FlagEnableQuaternion

Definition at line 336 of file messages.h.

6.20.2.3 enum USU::SamplingSettings::FunctionSelector

Sets the function Selector.

The function selector has 4 states:

- ReturnOnly: Does not change the Sampling Settings, only returns the current state
- Change: Set new Sampling settings, but do not store them in non-volatile memory (will be reset after shutdown)
- ChangeAndSave: Set new Sampling Settings and store them in non-volatile memory (will be permanent)
- ChangeWithoutReply: As Change but no response is sent

Enumerator:

ReturnOnly
Change
ChangeAndSave
ChangeWithoutReply

Definition at line 325 of file messages.h.

6.20.3 Constructor & Destructor Documentation

6.20.3.1 USU::SamplingSettings::SamplingSettings (FunctionSelector *funSel*, uint16_t *samplingPeriod_ms* = 10, uint16_t *dataCondFlags* = SamplingSettings::FlagDefault, uint8_t *gyroAccFilter* = 15, uint8_t *magFilter* = 17, uint16_t *upCompensation* = 10, uint16_t *northCompensation* = 10, uint8_t *magPower* = 0) [inline]

Creates the command.

Allocates a buffer for the byte commands. Sets the static bytes and fills the settings bytes based on the passed parameters.

Parameters

funSel	Sets the functions selector
sampling-Period_ms	Sets the sampling period in ms (1 to 1000)
dataCond-Flags	Sets general behaviour of the 3DM; use DataConditioning-flags
gyroAcc-Filter	Sets the filter value for the gyro and accelerometer
magFilter	Sets the filter value for the magnetometer
up-Compensation	Sets the time for up compensation
north-Compensation	Sets the time for north compensation
magPower	Sets the Power state

Definition at line 366 of file messages.h.

6.20.4 Member Function Documentation

6.20.4.1 bool USU::SamplingSettings::checkResponse (uint8_t * *buffer*, unsigned int *length*) [inline]

Checks if the response to this command has the correct setup.

Parameters

buffer	pointer to the byte array containing the response from the 3DM
length	length of the pointer

Returns

bool true if the response is correct, false if it suggests an error

Definition at line 402 of file messages.h.

6.20.5 Member Data Documentation

6.20.5.1 uint8_t USU::SamplingSettings::mCommand[size]

Buffer which contains the byte array for the command

Definition at line 420 of file messages.h.

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

- 3dm/messages.h

6.21 USU::ScopedLock Class Reference

Provides a helper class for Scoped Mutexes.

`#include <Lock.h>`

Public Member Functions

- ScopedLock (Lock &lock)

  Constructor: will lock the mutex.
- virtual ~ScopedLock ()

  Destructor: will unlock the mutex.

6.21.1 Detailed Description

Provides a helper class for Scoped Mutexes.

Create this object by passing a reference to a Lock object. It will lock the mutex when created and unlock it when destroyed, i.e. when going out of scope at the end of the "}". Can make it more convenient than manual (un)locking.

TODO: Test if it works correctly with a getter-method

Definition at line 89 of file Lock.h.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 USU::ScopedLock::ScopedLock (Lock & *lock*) [inline]

Constructor: will lock the mutex.

Parameters

lock	Reference to the Lock it needs to hold

Definition at line 112 of file Lock.h.

6.21.2.2 USU::ScopedLock::~ScopedLock ( ) [inline, virtual]

Destructor: will unlock the mutex.

Definition at line 119 of file Lock.h.

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

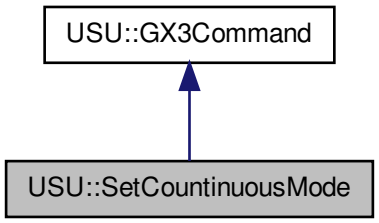
- threading/Lock.h

6.22 USU::SetCountinuousMode Class Reference

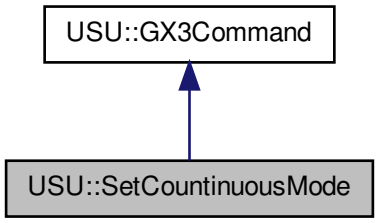
Represents the "Set continuous mode" command.

`#include <messages.h>`

Inheritance diagram for USU::SetCountinuousMode:



Collaboration diagram for USU::SetCountinuousMode:



Public Types

- enum { size = 4 }

Public Member Functions

- SetCountinuousMode (uint8_t CommandByte=0)

Creates the command.

- bool checkResponse (uint8_t *buffer, unsigned int length)

  Checks if the response to this command has the correct setup.

Public Attributes

- uint8_t mCommand [size]

6.22.1 Detailed Description

Represents the "Set continuous mode" command.

Definition at line 263 of file messages.h.

6.22.2 Member Enumeration Documentation

6.22.2.1 anonymous enum

Enumerator:

size

Definition at line 304 of file messages.h.

6.22.3 Constructor & Destructor Documentation

6.22.3.1 USU::SetCountinuousMode::SetCountinuousMode (uint8_t *CommandByte* = 0) [inline]

Creates the command.

Allocates a buffer for the byte commands. Sets the static bytes and fills the settings bytes based on the passed parameters.

Parameters

Command-Byte	Command code of the command which is to be executed periodically (Default stop continuous mode)

Definition at line 275 of file messages.h.

6.22.4 Member Function Documentation

6.22.4.1 bool USU::SetCountinuousMode::checkResponse (uint8_t * *buffer*, unsigned int *length*) [inline]

Checks if the response to this command has the correct setup.

Parameters

buffer	pointer to the byte array containing the response from the 3DM
length	length of the pointer

Returns

bool true if the response is correct, false if it suggests an error

Definition at line 290 of file messages.h.

6.22.5 Member Data Documentation

6.22.5.1 uint8_t USU::SetCountinuousMode::mCommand[size]

Buffer which contains the byte array for the command

Definition at line 305 of file messages.h.

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

- 3dm/messages.h

Chapter 7

File Documentation

7.1 3dm/gx3communicator.cpp File Reference

```
#include "gx3communicator.h" #include "messages.h" #include
<stdint.h> #include <iostream> #include <iomanip> #include
<stdexcept> #include <sys/time.h>
```

Functions

- int timeval_subtract (struct timeval *result, struct timeval *x, struct timeval *y)

7.1.1 Detailed Description

Contains the thread which handles the communication to the 3DM-GX3-25.

Author

Jan Sommer Created on: Apr 26, 2013

Definition in file gx3communicator.cpp.

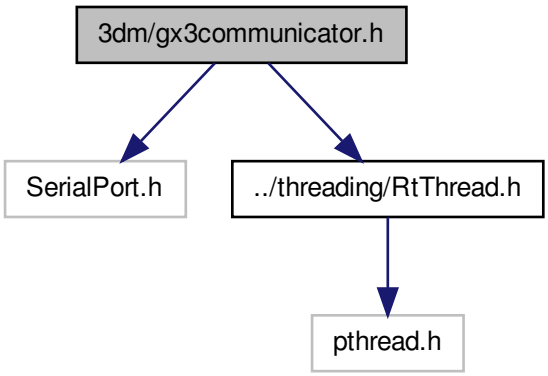
7.1.2 Function Documentation

7.1.2.1 int timeval_subtract (struct timeval * *result*, struct timeval * *x*, struct timeval * *y*)

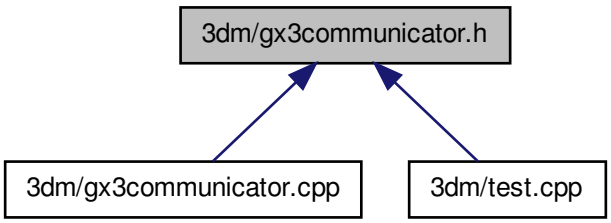
Definition at line 23 of file gx3communicator.cpp.

7.2 3dm/gx3communicator.h File Reference

`#include <SerialPort.h> #include "../threading/RtThread.-h"` Include dependency graph for gx3communicator.h:



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



Classes

- class USU::GX3Communicator

  Represents the Thread class for communication with the 3DM-GX3-25.

Namespaces

- namespace USU

  TODO: Make some proper exceptions.

7.2.1 Detailed Description

Contains the thread which handles the communication to the 3DM-GX3-25.

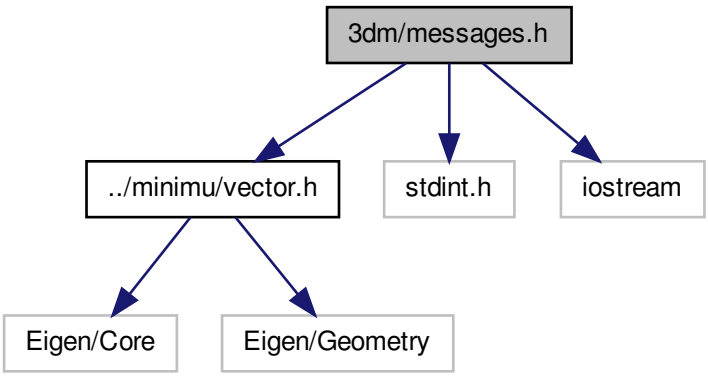
Author

Jan Sommer Created on: Apr 26, 2013

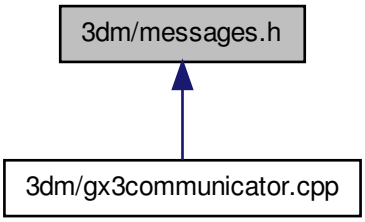
Definition in file gx3communicator.h.

7.3 3dm/messages.h File Reference

`#include "../minimu/vector.h"` `#include <stdint.h>` `#include <iostream>` Include dependency graph for messages.h:



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



Classes

- class USU::GX3Packet

  Abstract base class for received packets.
- class USU::RawAccAng

  Representation for receiving (raw) acceleration & angular rate packets.
- class USU::AccAngMag

  Representation for receiving acceleration, angular rate and magnetometer packets.
- class USU::AccAngMagOrientationMat

  Representation for packets containing the 3 sensor vectors and orientation matrix This class can be used with the commands which return 3 Vectors and a 3x3 Matrix. The units are:
- class USU::GX3Command

  Base class for commands send to the 3DM-GX3-25.
- class USU::SetCountinuousMode

  Represents the "Set continuous mode" command.
- class USU::SamplingSettings

  Represents the "Sampling Settings" command.

Namespaces

- namespace USU

  TODO: Make some proper exceptions.

Variables

- const uint8_t USU::RAW_ACC_ANG = 0xC1

- const uint8_t USU::ACC_ANG = 0xC2
- const uint8_t USU::DELTA_ANGLE_VEL = 0xC3
- const uint8_t USU::SET_CONTINUOUS_MODE = 0xC4
- const uint8_t USU::ORIENTATION_MATRIX = 0xC5
- const uint8_t USU::ORIENTATION_UPDATE_MAT = 0xC6
- const uint8_t USU::MAG_VEC = 0xC7
- const uint8_t USU::ACC_ANG_ORIENTATION_MAT = 0xC8
- const uint8_t USU::WRITE_ACC_BIAS_CORRECTION = 0xC9
- const uint8_t USU::WRITE_GYRO_BIAS_CORRECTION = 0xCA
- const uint8_t USU::ACC_ANG_MAG_VEC = 0xCB
- const uint8_t USU::ACC_ANG_MAG_VEC_ORIENTATION_MAT = 0xCC
- const uint8_t USU::CAPTURE_GYRO_BIAS = 0xCD
- const uint8_t USU::EULER_ANGLES = 0xCE
- const uint8_t USU::EULER_ANGLES_ANG_RATES = 0xCF
- const uint8_t USU::TRANSFER_TO_NONVOL_MEM = 0xD0
- const uint8_t USU::TEMPERATURES = 0xD1
- const uint8_t USU::GYRO_STABIL_ACC_ANG_MAG = 0xD2
- const uint8_t USU::DELTA_ANGLE_VEL_MAG_VEC = 0xD3
- const uint8_t USU::MODE = 0xD4
- const uint8_t USU::MODE_PRESET = 0xD5
- const uint8_t USU::CONTINUOUS_PRESET = 0xD6
- const uint8_t USU::TIMER = 0xD7
- const uint8_t USU::COMM_SETTINGS = 0xD9
- const uint8_t USU::STATIONARY_TEST = 0xDA
- const uint8_t USU::SAMPLING_SETTINGS = 0xDB
- const uint8_t USU::REALIGN_UP_NORTH = 0xDD
- const uint8_t USU::QUATERNION = 0xDF
- const uint8_t USU::WRITE_WORD_EEPROM = 0xE4
- const uint8_t USU::READ_WORD_EEPROM = 0xE5
- const uint8_t USU::READ_FIRMWARE_VER = 0xE9
- const uint8_t USU::READ_DEVICE_ID = 0xEA
- const uint8_t USU::STOP_CONTINUOUS = 0xFA
- const uint8_t USU::FIRMWARE_UPDATE = 0xFD
- const uint8_t USU::DEVICE_RESET = 0xFE

7.3.1 Detailed Description

File containing classes representing messages of the single byte protocol for the 3DM-GX3-25

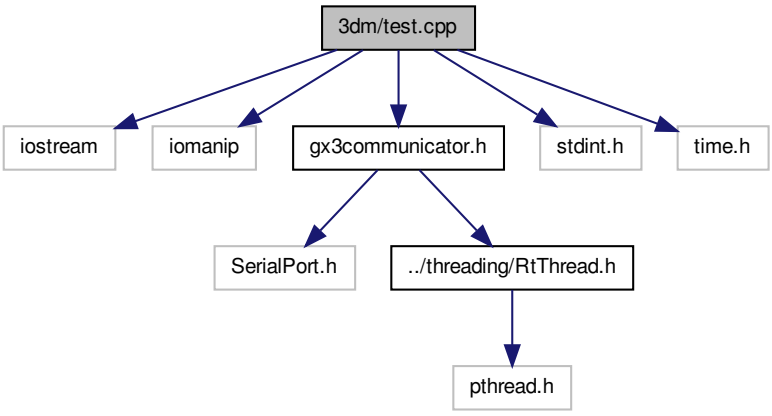
Author

Jan Sommer Created on: Apr 25, 2013

Definition in file messages.h.

7.4 3dm/test.cpp File Reference

`#include <iostream> #include <iomanip> #include "gx3communicator.h" #include <stdint.h> #include <time.h>` Include dependency graph for test.cpp:



Functions

- int main ()

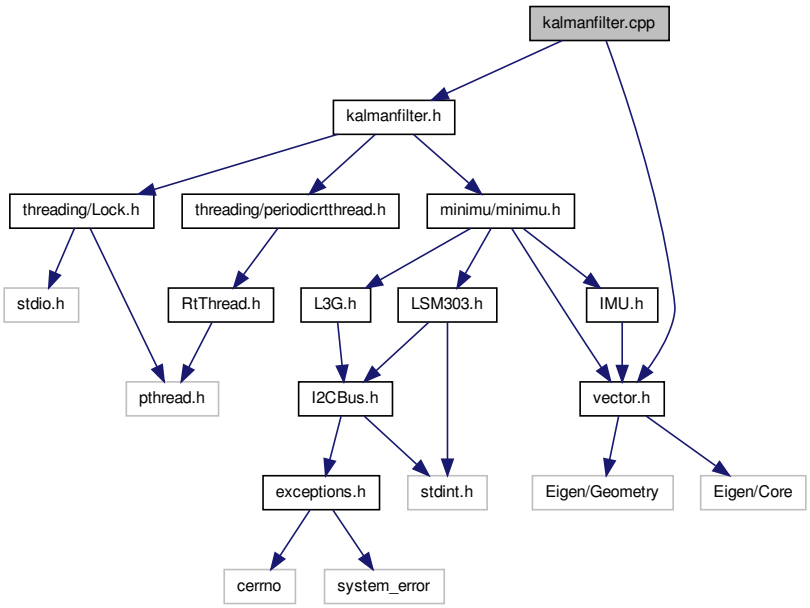
7.4.1 Function Documentation

7.4.1.1 int main ( )

Definition at line 9 of file test.cpp.

7.5 kalmanfilter.cpp File Reference

`#include "kalmanfilter.h" #include "minimu/vector.h"` Include dependency graph for kalmanfilter.cpp:



7.5.1 Detailed Description

C++ class for the sensor fusion and stated estimated. Based on the PeriodicRtThread class.

Author

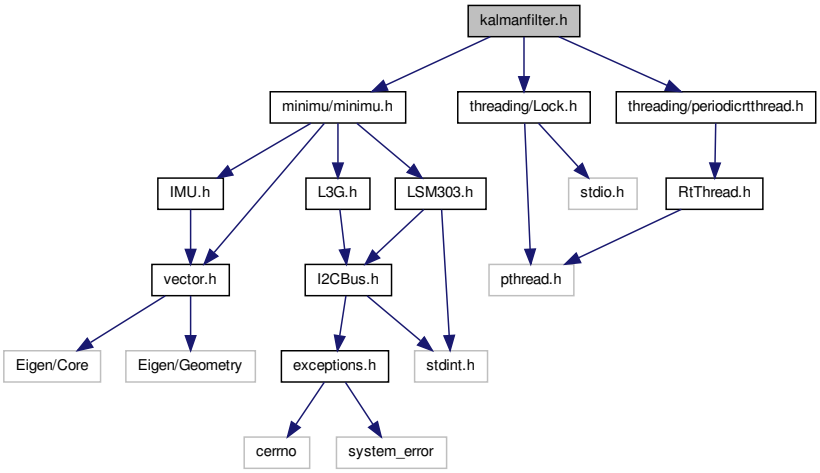
Jan Sommer Created on: Apr 20, 2013

Definition in file kalmanfilter.cpp.

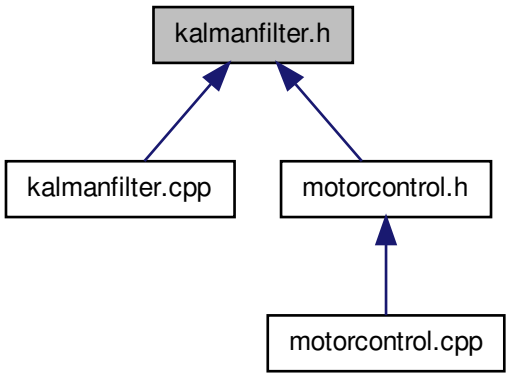
7.6 kalmanfilter.h File Reference

`#include "threading/periodicrtthread.h" #include "minimu/minimu.-h" #include "threading/Lock.h"` Include dependency graph for kalmanfilter.-

h:



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



Classes

- class USU::KalmanFilter

  Represents the Periodic Thread class for state estimation.

Namespaces

- namespace USU

  TODO: Make some proper exceptions.

7.6.1 Detailed Description

C++ class for the sensor fusion and stated estimated. Based on the PeriodicRtThread class.

Author

Jan Sommer Created on: Apr 20, 2013

Definition in file kalmanfilter.h.

7.7 main.cpp File Reference

Functions

- int main ()

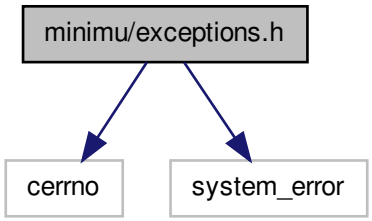
7.7.1 Function Documentation

7.7.1.1 int main ( )

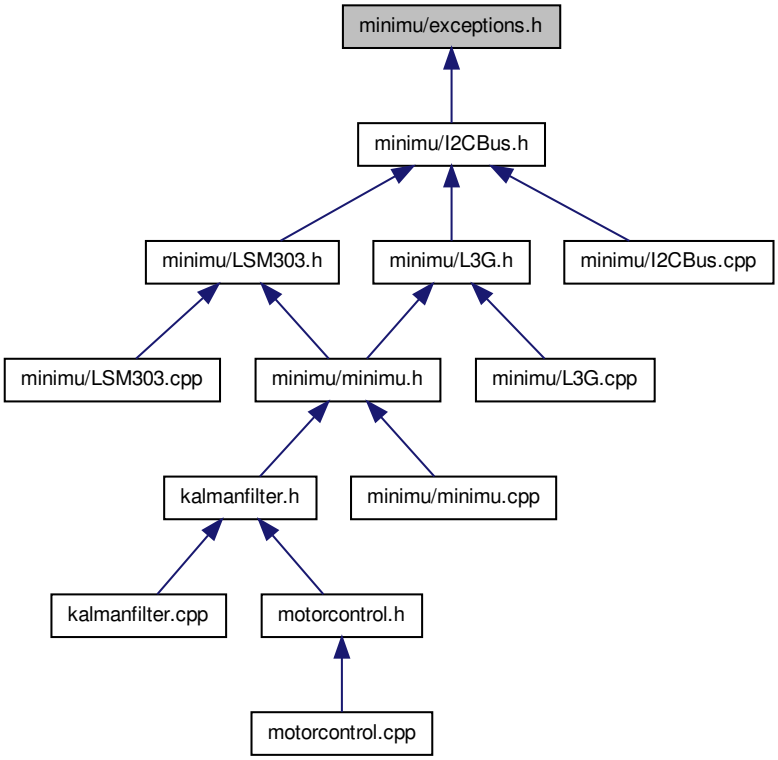
Definition at line 3 of file main.cpp.

7.8 minimu/exceptions.h File Reference

`#include <cerrno>` `#include <system_error>` Include dependency graph for exceptions.h:



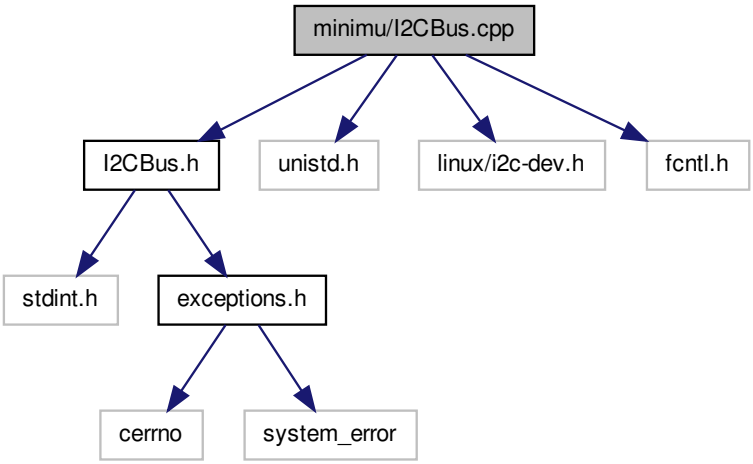
This graph shows which files directly or indirectly include this file:



7.9 minimu/I2CBus.cpp File Reference

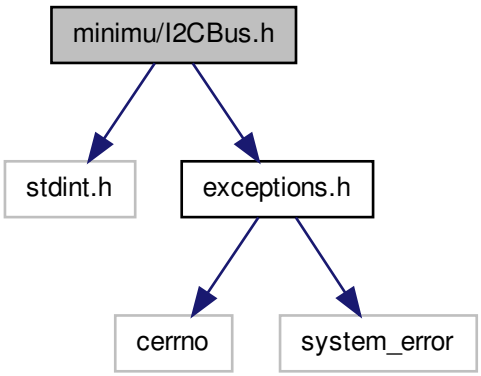
```
#include "I2CBus.h" #include <unistd.h> #include <linux/i2c-dev.-
```

h> #include <fcntl.h> Include dependency graph for I2CBus.cpp:

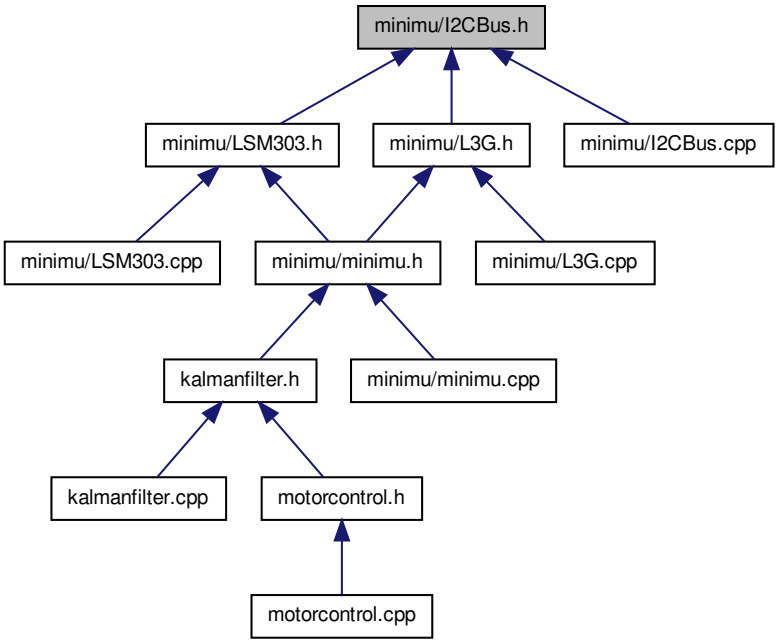


7.10 minimu/I2CBus.h File Reference

#include <stdint.h> #include "exceptions.h" Include dependency graph for I2CBus.h:



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

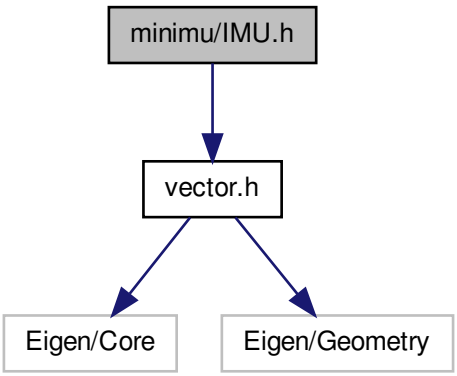


Classes

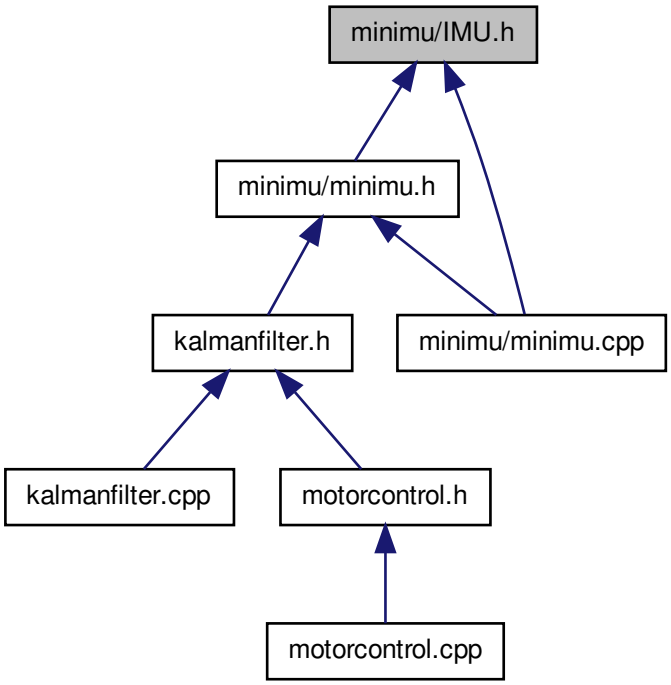
- class I2CBus

7.11 minimu/IMU.h File Reference

`#include "vector.h"` Include dependency graph for IMU.h:



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



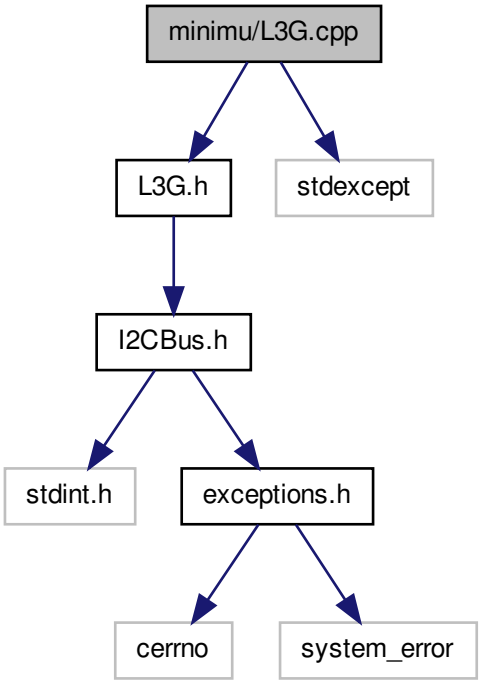
Classes

- class IMU

7.12 minimu/L3G.cpp File Reference

`#include "L3G.h"` `#include <stdexcept>` Include dependency graph for

L3G.cpp:



Defines

- #define L3G4200D_ADDRESS_SA0_LOW (0xD0 >> 1)
- #define L3G4200D_ADDRESS_SA0_HIGH (0xD2 >> 1)
- #define L3GD20_ADDRESS_SA0_LOW (0xD4 >> 1)
- #define L3GD20_ADDRESS_SA0_HIGH (0xD6 >> 1)

7.12.1 Define Documentation

7.12.1.1 #define L3G4200D_ADDRESS_SA0_HIGH (0xD2 >> 1)

Definition at line 5 of file L3G.cpp.

7.12.1.2 #define L3G4200D_ADDRESS_SA0_LOW (0xD0 >> 1)

Definition at line 4 of file L3G.cpp.

7.12.1.3 #define L3GD20_ADDRESS_SA0_HIGH (0xD6 >> 1)

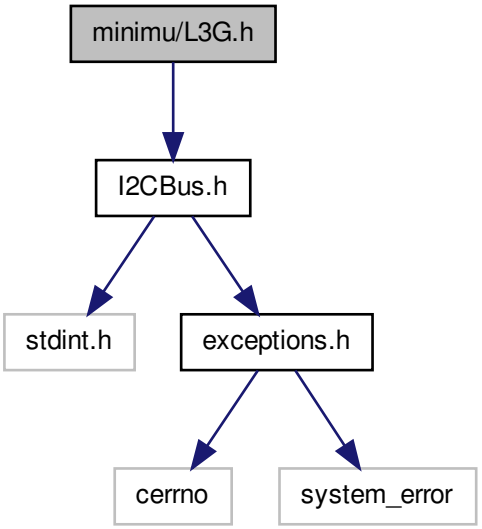
Definition at line 7 of file L3G.cpp.

7.12.1.4 #define L3GD20_ADDRESS_SA0_LOW (0xD4 >> 1)

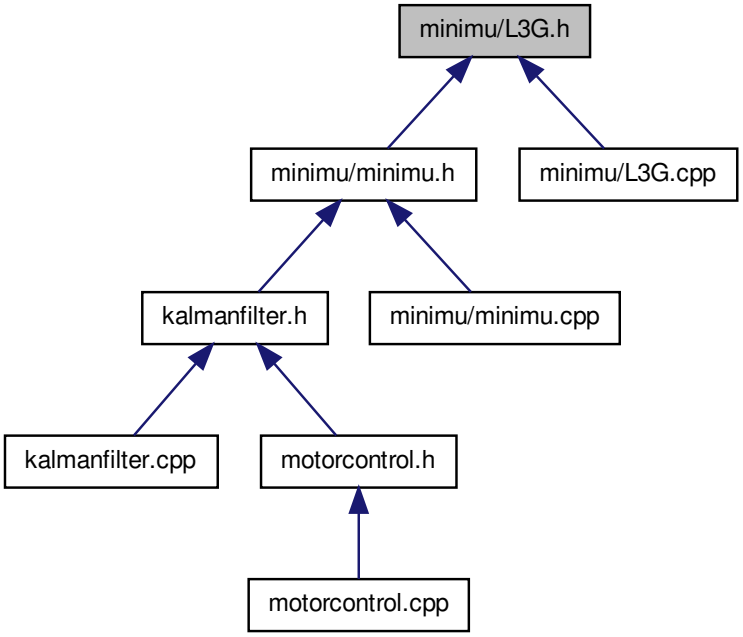
Definition at line 6 of file L3G.cpp.

7.13 minimu/L3G.h File Reference

`#include "I2CBus.h"` Include dependency graph for L3G.h:



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



Classes

- class L3G

Defines

- #define L3G_WHO_AM_I 0x0F
- #define L3G_CTRL_REG1 0x20
- #define L3G_CTRL_REG2 0x21
- #define L3G_CTRL_REG3 0x22
- #define L3G_CTRL_REG4 0x23
- #define L3G_CTRL_REG5 0x24
- #define L3G_REFERENCE 0x25
- #define L3G_OUT_TEMP 0x26
- #define L3G_STATUS_REG 0x27
- #define L3G_OUT_X_L 0x28
- #define L3G_OUT_X_H 0x29

- #define L3G_OUT_Y_L 0x2A
- #define L3G_OUT_Y_H 0x2B
- #define L3G_OUT_Z_L 0x2C
- #define L3G_OUT_Z_H 0x2D
- #define L3G_FIFO_CTRL_REG 0x2E
- #define L3G_FIFO_SRC_REG 0x2F
- #define L3G_INT1_CFG 0x30
- #define L3G_INT1_SRC 0x31
- #define L3G_INT1_THS_XH 0x32
- #define L3G_INT1_THS_XL 0x33
- #define L3G_INT1_THS_YH 0x34
- #define L3G_INT1_THS_YL 0x35
- #define L3G_INT1_THS_ZH 0x36
- #define L3G_INT1_THS_ZL 0x37
- #define L3G_INT1_DURATION 0x38

7.13.1 Define Documentation

7.13.1.1 #define L3G_CTRL_REG1 0x20

Definition at line 8 of file L3G.h.

7.13.1.2 #define L3G_CTRL_REG2 0x21

Definition at line 9 of file L3G.h.

7.13.1.3 #define L3G_CTRL_REG3 0x22

Definition at line 10 of file L3G.h.

7.13.1.4 #define L3G_CTRL_REG4 0x23

Definition at line 11 of file L3G.h.

7.13.1.5 #define L3G_CTRL_REG5 0x24

Definition at line 12 of file L3G.h.

7.13.1.6 #define L3G_FIFO_CTRL_REG 0x2E

Definition at line 24 of file L3G.h.

7.13.1.7 #define L3G_FIFO_SRC_REG 0x2F

Definition at line 25 of file L3G.h.

7.13.1.8 #define L3G_INT1_CFG 0x30

Definition at line 27 of file L3G.h.

7.13.1.9 #define L3G_INT1_DURATION 0x38

Definition at line 35 of file L3G.h.

7.13.1.10 #define L3G_INT1_SRC 0x31

Definition at line 28 of file L3G.h.

7.13.1.11 #define L3G_INT1_THS_XH 0x32

Definition at line 29 of file L3G.h.

7.13.1.12 #define L3G_INT1_THS_XL 0x33

Definition at line 30 of file L3G.h.

7.13.1.13 #define L3G_INT1_THS_YH 0x34

Definition at line 31 of file L3G.h.

7.13.1.14 #define L3G_INT1_THS_YL 0x35

Definition at line 32 of file L3G.h.

7.13.1.15 #define L3G_INT1_THS_ZH 0x36

Definition at line 33 of file L3G.h.

7.13.1.16 #define L3G_INT1_THS_ZL 0x37

Definition at line 34 of file L3G.h.

7.13.1.17 #define L3G_OUT_TEMP 0x26

Definition at line 14 of file L3G.h.

7.13.1.18 #define L3G_OUT_X_H 0x29

Definition at line 18 of file L3G.h.

7.13.1.19 #define L3G_OUT_X_L 0x28

Definition at line 17 of file L3G.h.

7.13.1.20 #define L3G_OUT_Y_H 0x2B

Definition at line 20 of file L3G.h.

7.13.1.21 #define L3G_OUT_Y_L 0x2A

Definition at line 19 of file L3G.h.

7.13.1.22 #define L3G_OUT_Z_H 0x2D

Definition at line 22 of file L3G.h.

7.13.1.23 #define L3G_OUT_Z_L 0x2C

Definition at line 21 of file L3G.h.

7.13.1.24 #define L3G_REFERENCE 0x25

Definition at line 13 of file L3G.h.

7.13.1.25 #define L3G_STATUS_REG 0x27

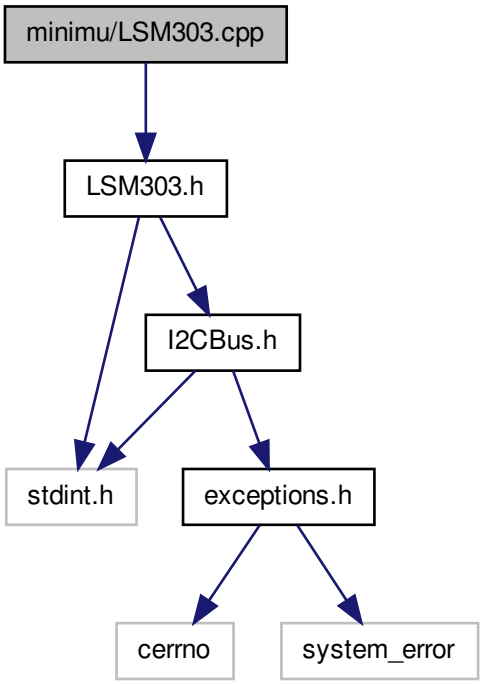
Definition at line 15 of file L3G.h.

7.13.1.26 #define L3G_WHO_AM_I 0x0F

Definition at line 6 of file L3G.h.

7.14 minimu/LSM303.cpp File Reference

`#include "LSM303.h"` Include dependency graph for LSM303.cpp:



Defines

- #define MAG_ADDRESS (0x3C >> 1)
- #define ACC_ADDRESS_SA0_A_LOW (0x30 >> 1)
- #define ACC_ADDRESS_SA0_A_HIGH (0x32 >> 1)

7.14.1 Define Documentation

7.14.1.1 #define ACC_ADDRESS_SA0_A_HIGH (0x32 >> 1)

Definition at line 20 of file LSM303.cpp.

7.14.1.2 #define ACC_ADDRESS_SA0_A_LOW (0x30 >> 1)

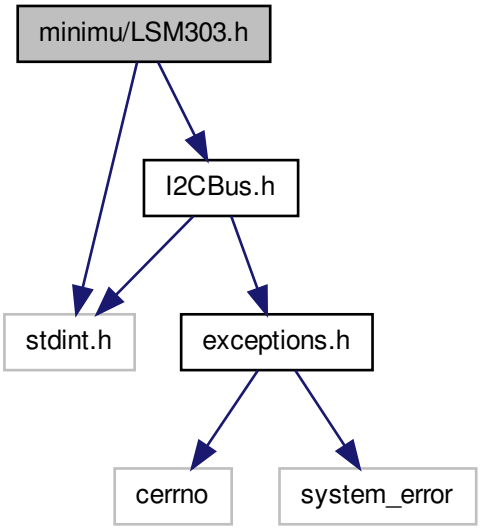
Definition at line 19 of file LSM303.cpp.

7.14.1.3 #define MAG_ADDRESS (0x3C >> 1)

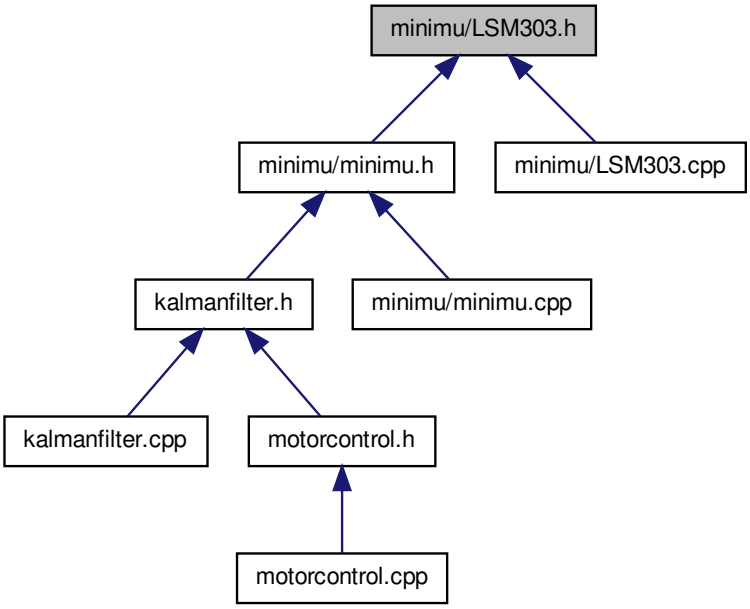
Definition at line 18 of file LSM303.cpp.

7.15 minimu/LSM303.h File Reference

`#include <stdint.h>` `#include "I2CBus.h"` Include dependency graph for LSM303.h:



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



Classes

- class LSM303

Defines

- #define LSM303_CTRL_REG1_A 0x20
- #define LSM303_CTRL_REG2_A 0x21
- #define LSM303_CTRL_REG3_A 0x22
- #define LSM303_CTRL_REG4_A 0x23
- #define LSM303_CTRL_REG5_A 0x24
- #define LSM303_CTRL_REG6_A 0x25
- #define LSM303_HP_FILTER_RESET_A 0x25
- #define LSM303_REFERENCE_A 0x26
- #define LSM303_STATUS_REG_A 0x27
- #define LSM303_OUT_X_L_A 0x28
- #define LSM303_OUT_X_H_A 0x29
- #define LSM303_OUT_Y_L_A 0x2A

- #define LSM303_OUT_Y_H_A 0x2B
- #define LSM303_OUT_Z_L_A 0x2C
- #define LSM303_OUT_Z_H_A 0x2D
- #define LSM303_FIFO_CTRL_REG_A 0x2E
- #define LSM303_FIFO_SRC_REG_A 0x2F
- #define LSM303_INT1_CFG_A 0x30
- #define LSM303_INT1_SRC_A 0x31
- #define LSM303_INT1_THS_A 0x32
- #define LSM303_INT1_DURATION_A 0x33
- #define LSM303_INT2_CFG_A 0x34
- #define LSM303_INT2_SRC_A 0x35
- #define LSM303_INT2_THS_A 0x36
- #define LSM303_INT2_DURATION_A 0x37
- #define LSM303_CLICK_CFG_A 0x38
- #define LSM303_CLICK_SRC_A 0x39
- #define LSM303_CLICK_THS_A 0x3A
- #define LSM303_TIME_LIMIT_A 0x3B
- #define LSM303_TIME_LATENCY_A 0x3C
- #define LSM303_TIME_WINDOW_A 0x3D
- #define LSM303_CRA_REG_M 0x00
- #define LSM303_CRB_REG_M 0x01
- #define LSM303_MR_REG_M 0x02
- #define LSM303_OUT_X_H_M 0x03
- #define LSM303_OUT_X_L_M 0x04
- #define LSM303_OUT_Y_H_M -1
- #define LSM303_OUT_Y_L_M -2
- #define LSM303_OUT_Z_H_M -3
- #define LSM303_OUT_Z_L_M -4
- #define LSM303_SR_REG_M 0x09
- #define LSM303_IRA_REG_M 0x0A
- #define LSM303_IRB_REG_M 0x0B
- #define LSM303_IRC_REG_M 0x0C
- #define LSM303_WHO_AM_I_M 0x0F
- #define LSM303_TEMP_OUT_H_M 0x31
- #define LSM303_TEMP_OUT_L_M 0x32
- #define LSM303DLH_OUT_Y_H_M 0x05
- #define LSM303DLH_OUT_Y_L_M 0x06
- #define LSM303DLH_OUT_Z_H_M 0x07
- #define LSM303DLH_OUT_Z_L_M 0x08
- #define LSM303DLM_OUT_Z_H_M 0x05
- #define LSM303DLM_OUT_Z_L_M 0x06
- #define LSM303DLM_OUT_Y_H_M 0x07
- #define LSM303DLM_OUT_Y_L_M 0x08
- #define LSM303DLHC_OUT_Z_H_M 0x05
- #define LSM303DLHC_OUT_Z_L_M 0x06

7.15.1 Define Documentation

7.15.1.1 #define LSM303_CLICK_CFG_A 0x38

Definition at line 38 of file LSM303.h.

7.15.1.2 #define LSM303_CLICK_SRC_A 0x39

Definition at line 39 of file LSM303.h.

7.15.1.3 #define LSM303_CLICK_THS_A 0x3A

Definition at line 40 of file LSM303.h.

7.15.1.4 #define LSM303_CRA_REG_M 0x00

Definition at line 45 of file LSM303.h.

7.15.1.5 #define LSM303_CRB_REG_M 0x01

Definition at line 46 of file LSM303.h.

7.15.1.6 #define LSM303_CTRL_REG1_A 0x20

Definition at line 9 of file LSM303.h.

7.15.1.7 #define LSM303_CTRL_REG2_A 0x21

Definition at line 10 of file LSM303.h.

7.15.1.8 #define LSM303_CTRL_REG3_A 0x22

Definition at line 11 of file LSM303.h.

7.15.1.9 #define LSM303_CTRL_REG4_A 0x23

Definition at line 12 of file LSM303.h.

7.15.1.10 #define LSM303_CTRL_REG5_A 0x24

Definition at line 13 of file LSM303.h.

7.15.1.11 #define LSM303_CTRL_REG6_A 0x25

Definition at line 14 of file LSM303.h.

7.15.1.12 #define LSM303_FIFO_CTRL_REG_A 0x2E

Definition at line 26 of file LSM303.h.

7.15.1.13 #define LSM303_FIFO_SRC_REG_A 0x2F

Definition at line 27 of file LSM303.h.

7.15.1.14 #define LSM303_HP_FILTER_RESET_A 0x25

Definition at line 15 of file LSM303.h.

7.15.1.15 #define LSM303_INT1_CFG_A 0x30

Definition at line 29 of file LSM303.h.

7.15.1.16 #define LSM303_INT1_DURATION_A 0x33

Definition at line 32 of file LSM303.h.

7.15.1.17 #define LSM303_INT1_SRC_A 0x31

Definition at line 30 of file LSM303.h.

7.15.1.18 #define LSM303_INT1_THS_A 0x32

Definition at line 31 of file LSM303.h.

7.15.1.19 #define LSM303_INT2_CFG_A 0x34

Definition at line 33 of file LSM303.h.

7.15.1.20 #define LSM303_INT2_DURATION_A 0x37

Definition at line 36 of file LSM303.h.

7.15.1.21 #define LSM303_INT2_SRC_A 0x35

Definition at line 34 of file LSM303.h.

7.15.1.22 #define LSM303_INT2_THS_A 0x36

Definition at line 35 of file LSM303.h.

7.15.1.23 #define LSM303_IRA_REG_M 0x0A

Definition at line 57 of file LSM303.h.

7.15.1.24 #define LSM303_IRB_REG_M 0x0B

Definition at line 58 of file LSM303.h.

7.15.1.25 #define LSM303_IRC_REG_M 0x0C

Definition at line 59 of file LSM303.h.

7.15.1.26 #define LSM303_MR_REG_M 0x02

Definition at line 47 of file LSM303.h.

7.15.1.27 #define LSM303_OUT_X_H_A 0x29

Definition at line 20 of file LSM303.h.

7.15.1.28 #define LSM303_OUT_X_H_M 0x03

Definition at line 49 of file LSM303.h.

7.15.1.29 #define LSM303_OUT_X_L_A 0x28

Definition at line 19 of file LSM303.h.

7.15.1.30 #define LSM303_OUT_X_L_M 0x04

Definition at line 50 of file LSM303.h.

7.15.1.31 #define LSM303_OUT_Y_H_A 0x2B

Definition at line 22 of file LSM303.h.

7.15.1.32 #define LSM303_OUT_Y_H_M -1

Definition at line 51 of file LSM303.h.

7.15.1.33 #define LSM303_OUT_Y_L_A 0x2A

Definition at line 21 of file LSM303.h.

7.15.1.34 #define LSM303_OUT_Y_L_M -2

Definition at line 52 of file LSM303.h.

7.15.1.35 #define LSM303_OUT_Z_H_A 0x2D

Definition at line 24 of file LSM303.h.

7.15.1.36 #define LSM303_OUT_Z_H_M -3

Definition at line 53 of file LSM303.h.

7.15.1.37 #define LSM303_OUT_Z_L_A 0x2C

Definition at line 23 of file LSM303.h.

7.15.1.38 #define LSM303_OUT_Z_L_M -4

Definition at line 54 of file LSM303.h.

7.15.1.39 #define LSM303_REFERENCE_A 0x26

Definition at line 16 of file LSM303.h.

7.15.1.40 #define LSM303_SR_REG_M 0x09

Definition at line 56 of file LSM303.h.

7.15.1.41 #define LSM303_STATUS_REG_A 0x27

Definition at line 17 of file LSM303.h.

7.15.1.42 #define LSM303_TEMP_OUT_H_M 0x31

Definition at line 63 of file LSM303.h.

7.15.1.43 #define LSM303_TEMP_OUT_L_M 0x32

Definition at line 64 of file LSM303.h.

7.15.1.44 #define LSM303_TIME_LATENCY_A 0x3C

Definition at line 42 of file LSM303.h.

7.15.1.45 #define LSM303_TIME_LIMIT_A 0x3B

Definition at line 41 of file LSM303.h.

7.15.1.46 #define LSM303_TIME_WINDOW_A 0x3D

Definition at line 43 of file LSM303.h.

7.15.1.47 #define LSM303_WHO_AM_I_M 0x0F

Definition at line 61 of file LSM303.h.

7.15.1.48 #define LSM303DLH_OUT_Y_H_M 0x05

Definition at line 65 of file LSM303.h.

7.15.1.49 #define LSM303DLH_OUT_Y_L_M 0x06

Definition at line 66 of file LSM303.h.

7.15.1.50 #define LSM303DLH_OUT_Z_H_M 0x07

Definition at line 67 of file LSM303.h.

7.15.1.51 #define LSM303DLH_OUT_Z_L_M 0x08

Definition at line 68 of file LSM303.h.

7.15.1.52 #define LSM303DLHC_OUT_Z_H_M 0x05

Definition at line 75 of file LSM303.h.

7.15.1.53 #define LSM303DLHC_OUT_Z_L_M 0x06

Definition at line 76 of file LSM303.h.

7.15.1.54 #define LSM303DLM_OUT_Y_H_M 0x07

Definition at line 72 of file LSM303.h.

7.15.1.55 #define LSM303DLM_OUT_Y_L_M 0x08

Definition at line 73 of file LSM303.h.

7.15.1.56 #define LSM303DLM_OUT_Z_H_M 0x05

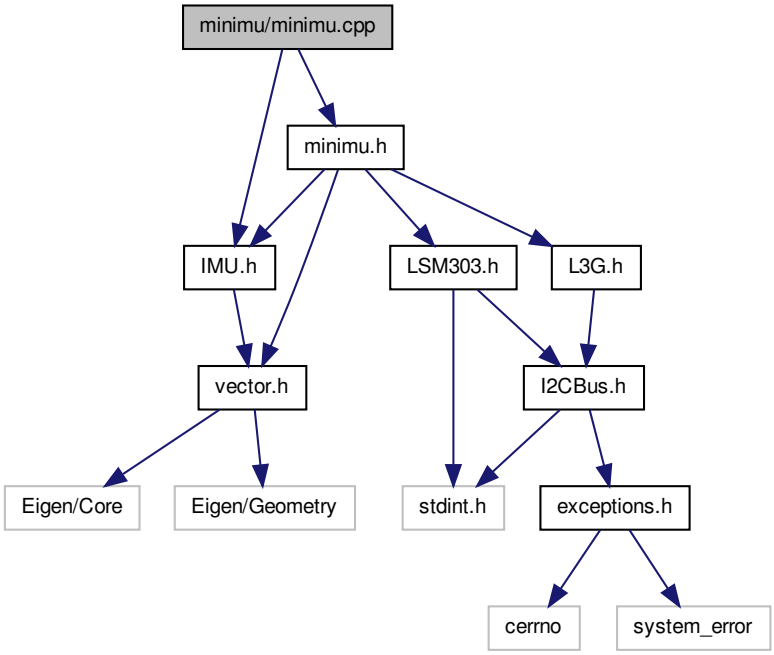
Definition at line 70 of file LSM303.h.

7.15.1.57 #define LSM303DLM_OUT_Z_L_M 0x06

Definition at line 71 of file LSM303.h.

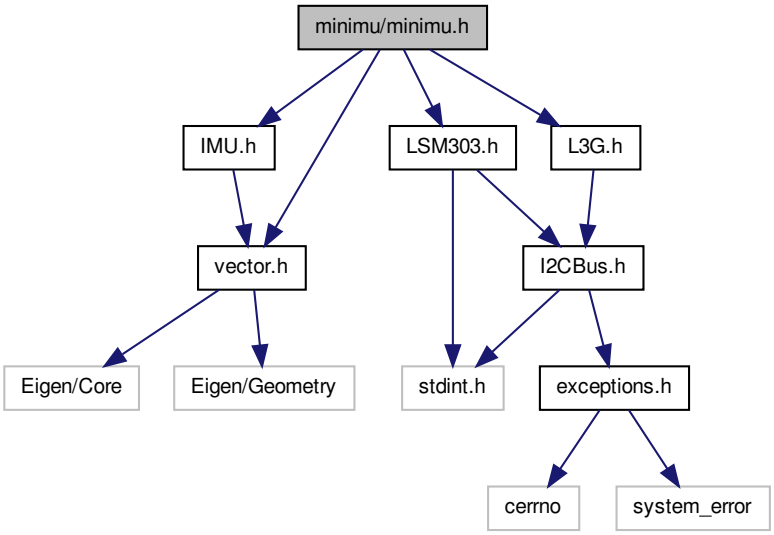
7.16 minimu/minimu.cpp File Reference

`#include "minimu.h"` `#include "IMU.h"` Include dependency graph for minimu.cpp:

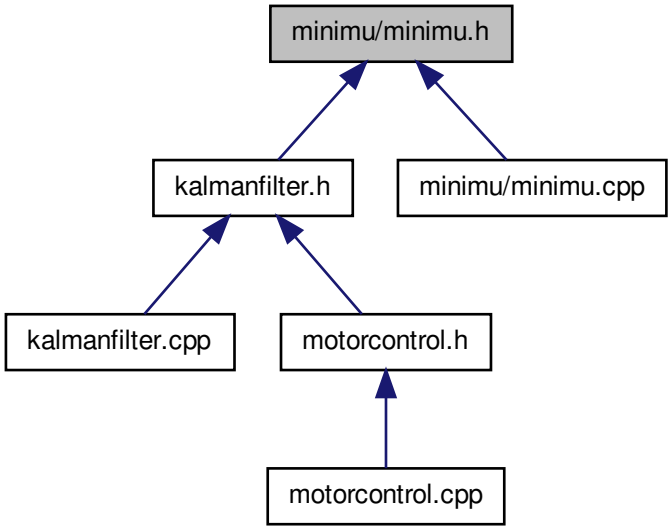


7.17 minimu/minimu.h File Reference

```
#include "IMU.h"   #include "LSM303.h"   #include "L3G.h" ×
#include "vector.h" Include dependency graph for minimu.h:
```



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



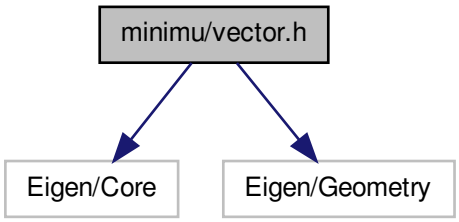
Classes

- class MinImu

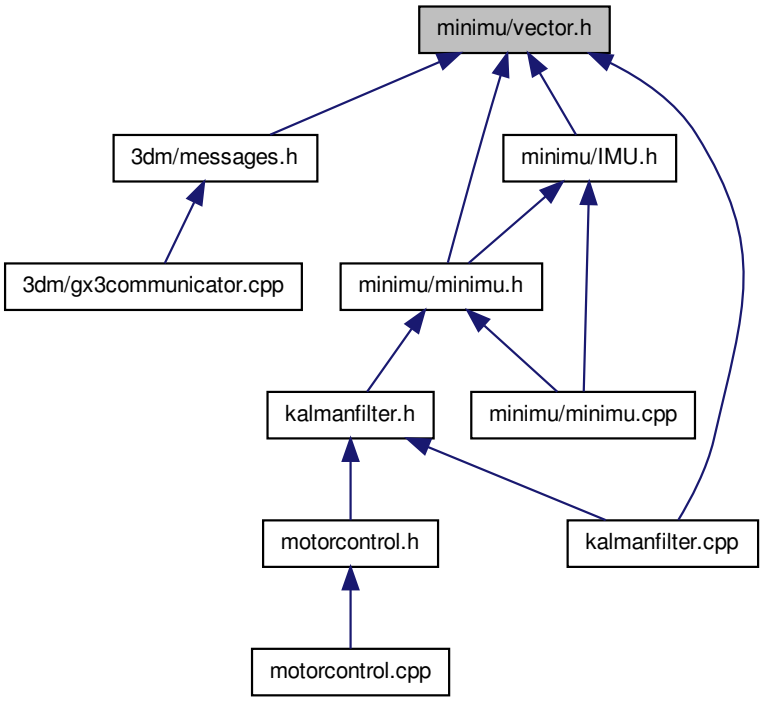
7.18 minimu/vector.h File Reference

`#include "Eigen/Core" #include "Eigen/Geometry"` Include depen-

dency graph for vector.h:



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



Typedefs

- typedef Eigen::Vector3f vector
- typedef Eigen::Vector3i int_vector
- typedef Eigen::Matrix3f matrix
- typedef Eigen::Quaternionf quaternion

7.18.1 Typedef Documentation

7.18.1.1 typedef Eigen::Vector3i int_vector

Definition at line 7 of file vector.h.

7.18.1.2 typedef Eigen::Matrix3f matrix

Definition at line 8 of file vector.h.

7.18.1.3 typedef Eigen::Quaternionf quaternion

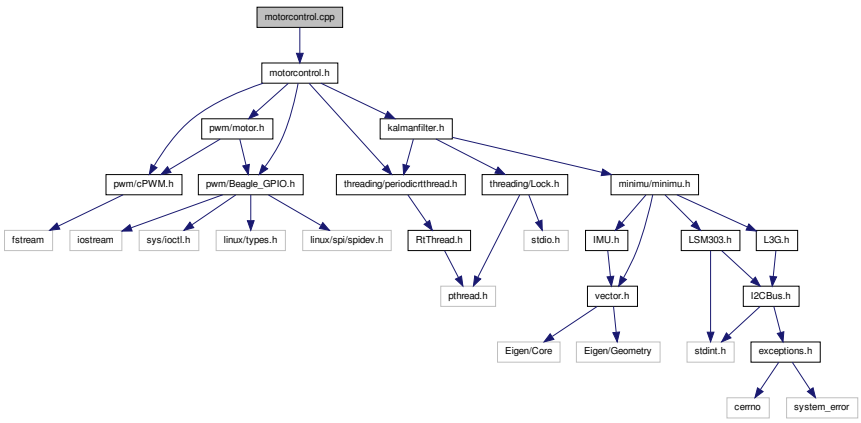
Definition at line 9 of file vector.h.

7.18.1.4 typedef Eigen::Vector3f vector

Definition at line 6 of file vector.h.

7.19 motorcontrol.cpp File Reference

`#include "motorcontrol.h"` Include dependency graph for motorcontrol.cpp:



7.19.1 Detailed Description

C++ class for the calculation of the control response. Based on the PeriodicRtThread class.

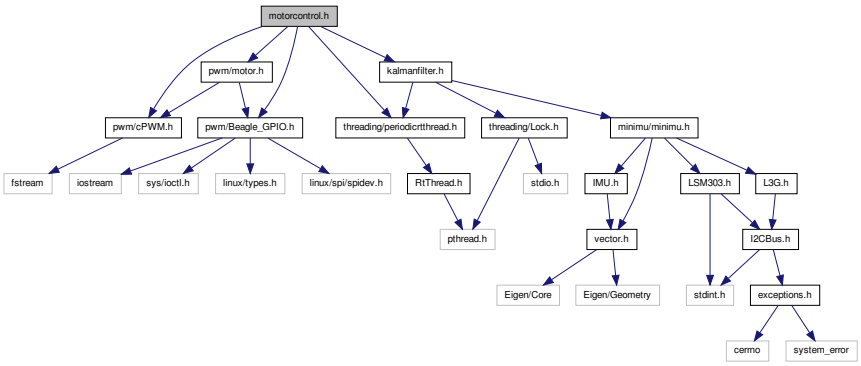
Author

Jan Sommer Created on: Apr 22, 2013

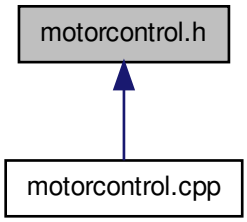
Definition in file motorcontrol.cpp.

7.20 motorcontrol.h File Reference

`#include "threading/periodicrtthread.h" #include "pwm/cPWM.h" #include "pwm/Beagle_GPIO.h" #include "pwm/motor.h" #include "kalmanfilter.h"` Include dependency graph for motorcontrol.h:



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



Classes

- class USU::MotorControl

  Represents the Periodic task for motor control.

Namespaces

- namespace USU

  TODO: Make some proper exceptions.

7.20.1 Detailed Description

C++ class for the calculation of the control response. Based on the PeriodicRtThread class.

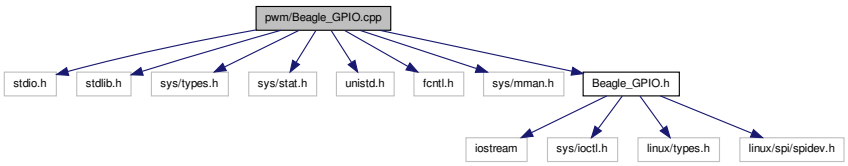
Author

Jan Sommer Created on: Apr 22, 2013

Definition in file motorcontrol.h.

7.21 pwm/Beagle_GPIO.cpp File Reference

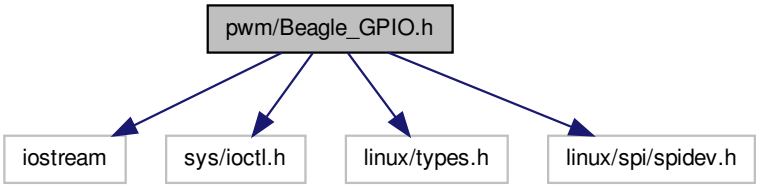
`#include <stdio.h> #include <stdlib.h> #include <sys/types.h> #include <sys/stat.h> #include <unistd.h> #include <fcntl.h> #include <sys/mman.h> #include "Beagle_GPIO.h"`
Include dependency graph for Beagle_GPIO.cpp:



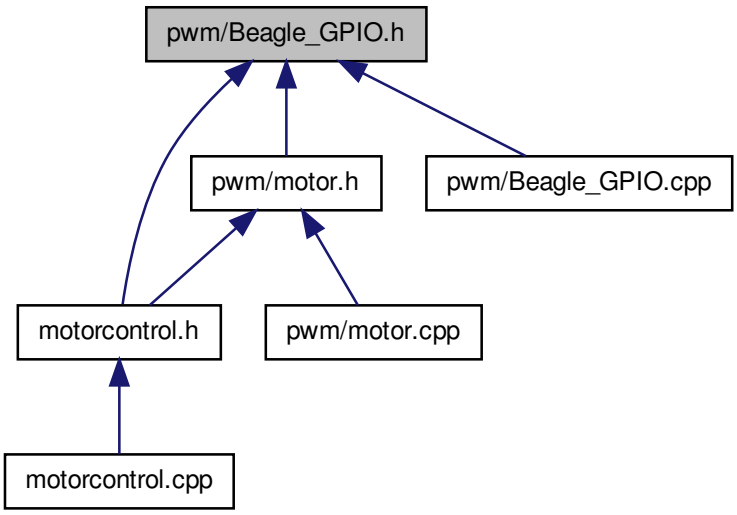
7.22 pwm/Beagle_GPIO.h File Reference

`#include <iostream> #include <sys/ioctl.h> #include <linux/types.h> #include <linux/spi/spidev.h>` Include dependency graph for Beagle-

_GPIO.h:



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



Classes

- class Beagle_GPIO

Defines

- #define GPIO_ERROR(msg) std::cout << "[GPIO] Error : " << msg << std::endl;
- #define BEAGLE_GPIO_DEBUG
- #define GPIO_PRINT(msg) std::cout << "[GPIO] : " << msg << std::endl;
- #define assert(condition)

7.22.1 Define Documentation

7.22.1.1 #define assert(*condition*)

Value:

```
if (!(condition))       \
                {                       \
                        GPIO_ERROR( "Assert Failed in file '" << __FILE__ << "'
        on line " << __LINE__ );        \
                        exit(0);        \
                }
```

Definition at line 32 of file Beagle_GPIO.h.

7.22.1.2 #define BEAGLE_GPIO_DEBUG

Definition at line 29 of file Beagle_GPIO.h.

7.22.1.3 #define GPIO_ERROR(*msg*) std::cout << "[GPIO] Error : " << msg << std::endl;

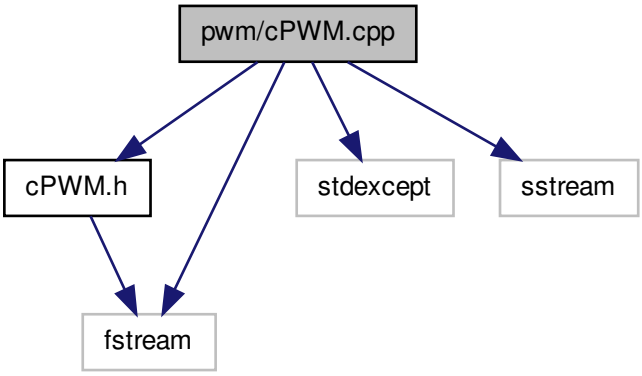
Definition at line 27 of file Beagle_GPIO.h.

7.22.1.4 #define GPIO_PRINT(*msg*) std::cout << "[GPIO] : " << msg << std::endl;

Definition at line 31 of file Beagle_GPIO.h.

7.23 pwm/cPWM.cpp File Reference

`#include "cPWM.h" #include <stdexcept> #include <fstream> #include <sstream>` Include dependency graph for cPWM.cpp:



Namespaces

- namespace cPWM

  Simple C++ class wrapper for beaglebone PWM eHRPWM interface.

7.23.1 Detailed Description

Simple C++ class wrapper for beaglebone PWM eHRPWM interface

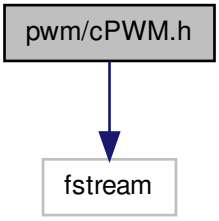
Author

claus Created on: Jun 13, 2012 Author: claus http://quadrotordiaries.-blogspot.com

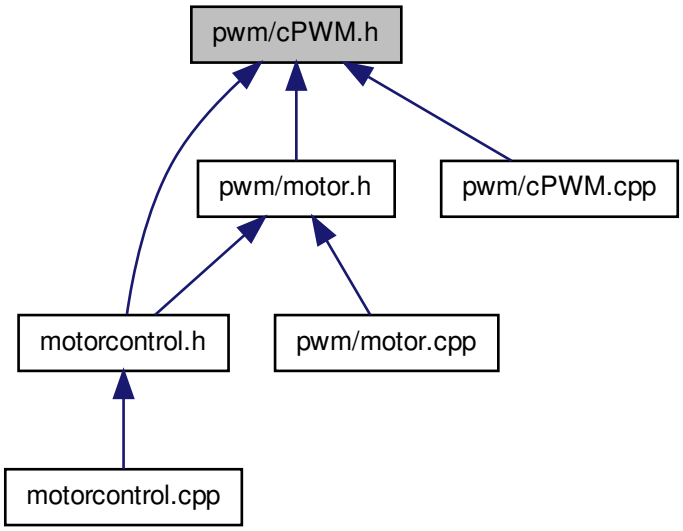
Definition in file cPWM.cpp.

7.24 pwm/cPWM.h File Reference

`#include <fstream>` Include dependency graph for cPWM.h:



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



Classes

- class cPWM::cPWM

Namespaces

- namespace cPWM

  Simple C++ class wrapper for beaglebone PWM eHRPWM interface.

Defines

- #define SYSFS_EHRPWM_PREFIX "/sys/class/pwm/ehrpwm."
- #define SYSFS_EHRPWM_SUFFIX_A ":0"
- #define SYSFS_EHRPWM_SUFFIX_B ":1"
- #define SYSFS_EHRPWM_DUTY_NS "duty_ns"
- #define SYSFS_EHRPWM_DUTY_PERCENT "duty_percent"
- #define SYSFS_EHRPWM_PERIOD_NS "period_ns"
- #define SYSFS_EHRPWM_PERIOD_FREQ "period_freq"
- #define SYSFS_EHRPWM_POLARITY "polarity"
- #define SYSFS_EHRPWM_RUN "run"
- #define SYSFS_EHRPWM_REQUEST "request"

7.24.1 Detailed Description

Simple C++ class wrapper for beaglebone PWM eHRPWM interface header file

Author

claus Created on: Jun 13, 2012 Author: claus http://quadrotordiaries.-blogspot.com

Definition in file cPWM.h.

7.24.2 Define Documentation

7.24.2.1 #define SYSFS_EHRPWM_DUTY_NS "duty_ns"

Definition at line 63 of file cPWM.h.

7.24.2.2 #define SYSFS_EHRPWM_DUTY_PERCENT "duty_percent"

Definition at line 64 of file cPWM.h.

7.24.2.3 #define SYSFS_EHRPWM_PERIOD_FREQ "period_freq"

Definition at line 66 of file cPWM.h.

7.24.2.4 #define SYSFS_EHRPWM_PERIOD_NS "period_ns"

Definition at line 65 of file cPWM.h.

7.24.2.5 #define SYSFS_EHRPWM_POLARITY "polarity"

Definition at line 67 of file cPWM.h.

7.24.2.6 #define SYSFS_EHRPWM_PREFIX "/sys/class/pwm/ehrpwm."

Definition at line 60 of file cPWM.h.

7.24.2.7 #define SYSFS_EHRPWM_REQUEST "request"

Definition at line 69 of file cPWM.h.

7.24.2.8 #define SYSFS_EHRPWM_RUN "run"

Definition at line 68 of file cPWM.h.

7.24.2.9 #define SYSFS_EHRPWM_SUFFIX_A ":0"

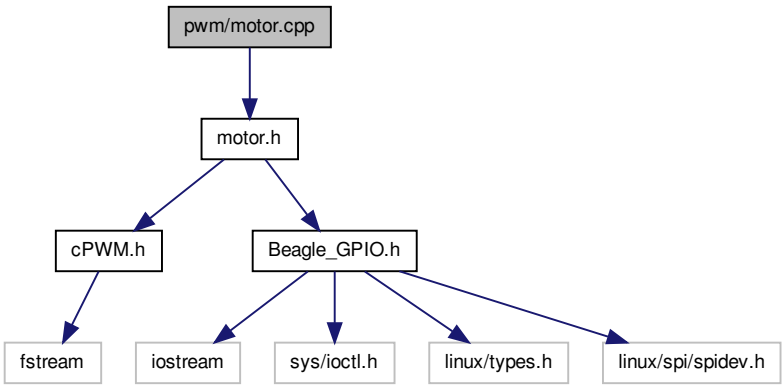
Definition at line 61 of file cPWM.h.

7.24.2.10 #define SYSFS_EHRPWM_SUFFIX_B ":1"

Definition at line 62 of file cPWM.h.

7.25 pwm/motor.cpp File Reference

`#include "motor.h"` Include dependency graph for motor.cpp:



7.25.1 Detailed Description

Class to represent a motor

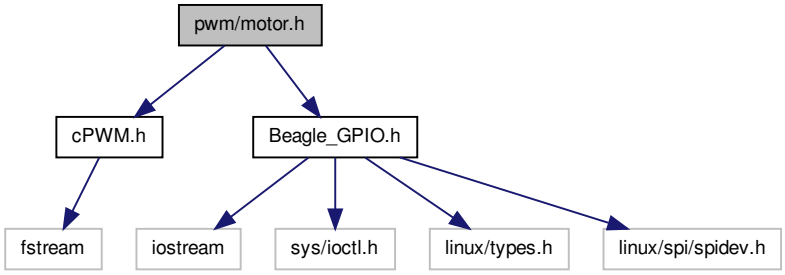
Author

Jan Sommer Created on: Apr 22, 2013

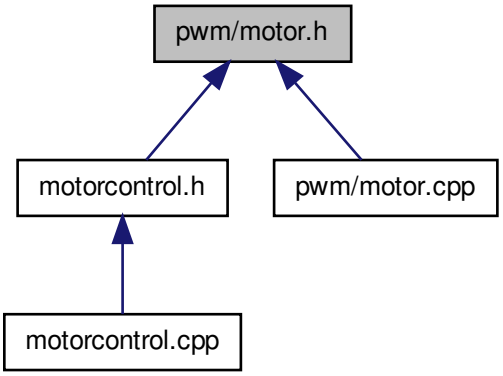
Definition in file motor.cpp.

7.26 pwm/motor.h File Reference

`#include "cPWM.h"` `#include "Beagle_GPIO.h"` Include dependency graph for motor.h:



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



Classes

- class USU::Motor

Namespaces

- namespace USU

  TODO: Make some proper exceptions.

Typedefs

- typedef void(cPWM::* SetDutyCyle)(unsigned int)

7.26.1 Detailed Description

Class to represent a motor

Author

Jan Sommer Created on: Apr 22, 2013

Definition in file motor.h.

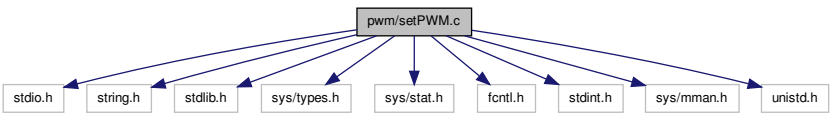
7.26.2 Typedef Documentation

7.26.2.1 typedef void(cPWM::* SetDutyCyle)(unsigned int)

Definition at line 18 of file motor.h.

7.27 pwm/setPWM.c File Reference

`#include <stdio.h> #include <string.h> #include <stdlib.h> #include <sys/types.h> #include <sys/stat.h> #include <fcntl.h> #include <stdint.h> #include <sys/mman.h> × #include <unistd.h>` Include dependency graph for setPWM.c:



Defines

- #define CM_PER_REG_START 0x44e00000
- #define CM_PER_REG_LENGTH 1024

- #define CM_PER_EPWMSS0_CLKCTRL_OFFSET 0xd4
- #define CM_PER_EPWMSS1_CLKCTRL_OFFSET 0xcc
- #define CM_PER_EPWMSS2_CLKCTRL_OFFSET 0xd8
- #define PWM_CLOCK_ENABLE 0x2
- #define PWM_CLOCK_DISABLE 0x0
- #define PWM_LIST_MAX 3

Functions

- void print_usage (const char ∗message)
- int main (int argc, char ∗∗argv)

Variables

- int PWM_OFFSETS [PWM_LIST_MAX]

7.27.1 Define Documentation

7.27.1.1 #define CM_PER_EPWMSS0_CLKCTRL_OFFSET 0xd4

Definition at line 13 of file setPWM.c.

7.27.1.2 #define CM_PER_EPWMSS1_CLKCTRL_OFFSET 0xcc

Definition at line 14 of file setPWM.c.

7.27.1.3 #define CM_PER_EPWMSS2_CLKCTRL_OFFSET 0xd8

Definition at line 15 of file setPWM.c.

7.27.1.4 #define CM_PER_REG_LENGTH 1024

Definition at line 12 of file setPWM.c.

7.27.1.5 #define CM_PER_REG_START 0x44e00000

Definition at line 11 of file setPWM.c.

7.27.1.6 #define PWM_CLOCK_DISABLE 0x0

Definition at line 18 of file setPWM.c.

7.27.1.7 #define PWM_CLOCK_ENABLE 0x2

Definition at line 17 of file setPWM.c.

7.27.1.8 #define PWM_LIST_MAX 3

Definition at line 20 of file setPWM.c.

7.27.2 Function Documentation

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

Definition at line 36 of file setPWM.c.

7.27.2.2 void print_usage (const char * *message*)

Definition at line 28 of file setPWM.c.

7.27.3 Variable Documentation

7.27.3.1 int PWM_OFFSETS[PWM_LIST_MAX]

Initial value:

```
 {
  CM_PER_EPWMSS0_CLKCTRL_OFFSET / sizeof (uint32_t),
  CM_PER_EPWMSS1_CLKCTRL_OFFSET / sizeof (uint32_t),
  CM_PER_EPWMSS2_CLKCTRL_OFFSET / sizeof (uint32_t)
}
```

Definition at line 22 of file setPWM.c.

7.28 pwm/setPWMReg.py File Reference

Namespaces

- namespace setPWMReg

Variables

- int setPWMReg.MMAP_OFFSET = 0x44c00000
- int setPWMReg.MMAP_SIZE = 0x48ffffff
- int setPWMReg.CM_PER_BASE = 0x44e00000
- int setPWMReg.CM_PER_EPWMSS1_CLKCTRL = 0xcc

- int setPWMReg.CM_PER_EPWMSS0_CLKCTRL = 0xd4

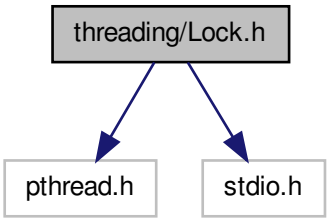
- int setPWMReg.CM_PER_EPWMSS2_CLKCTRL = 0xd8

- tuple setPWMReg.mem = mmap(f.fileno(), MMAP_SIZE, offset=MMAP_OFFSET)

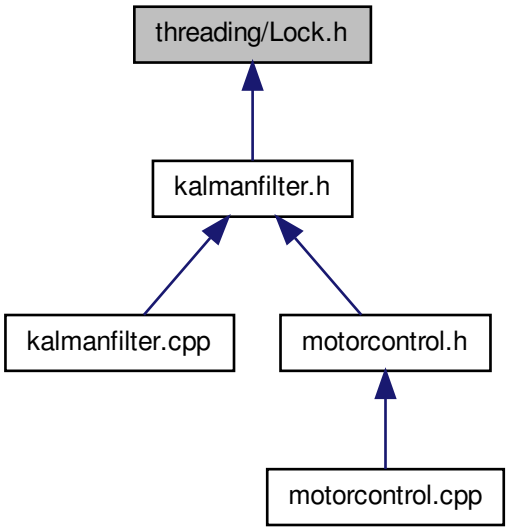
- tuple setPWMReg.val = _getReg(CM_PER_EPWMSS1_CLKCTRL)

7.29 threading/Lock.h File Reference

`#include <pthread.h>` `#include <stdio.h>` Include dependency graph for Lock.h:



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



Classes

- class USU::Lock

  Wrapper class for pthread mutexes.
- class USU::ScopedLock

  Provides a helper class for Scoped Mutexes.

Namespaces

- namespace USU

  TODO: Make some proper exceptions.

7.29.1 Detailed Description

Small C++ wrapper classes for pthread mutexes

Author

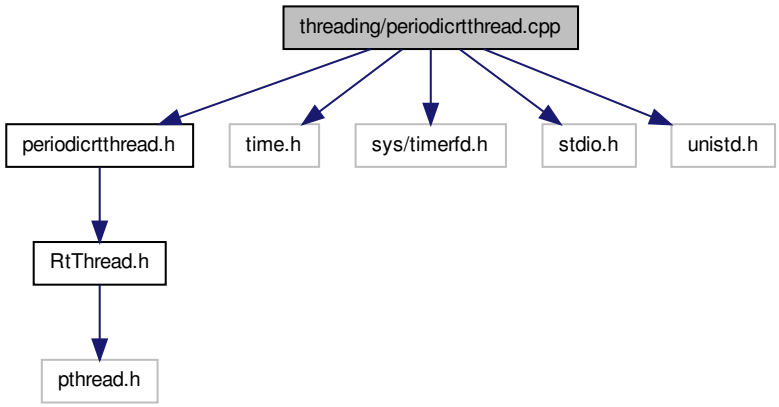
Jan Sommer Created on: Apr 10, 2013

Definition in file Lock.h.

7.30 threading/periodicrtthread.cpp File Reference

`#include "periodicrtthread.h" #include <time.h> #include <sys/timerfd.h> #include <stdio.h> #include <unistd.h>` ×

Include dependency graph for periodicrtthread.cpp:



7.30.1 Detailed Description

Small C++ wrapper class to create a realtime scheduled pthread with periodic timer events.

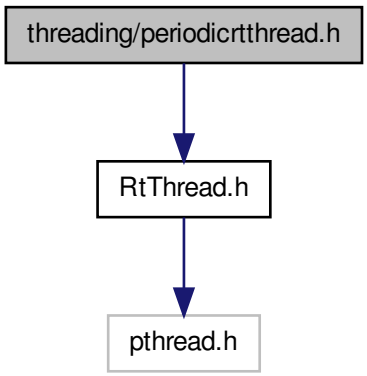
Author

Jan Sommer Created on: Apr 10, 2013

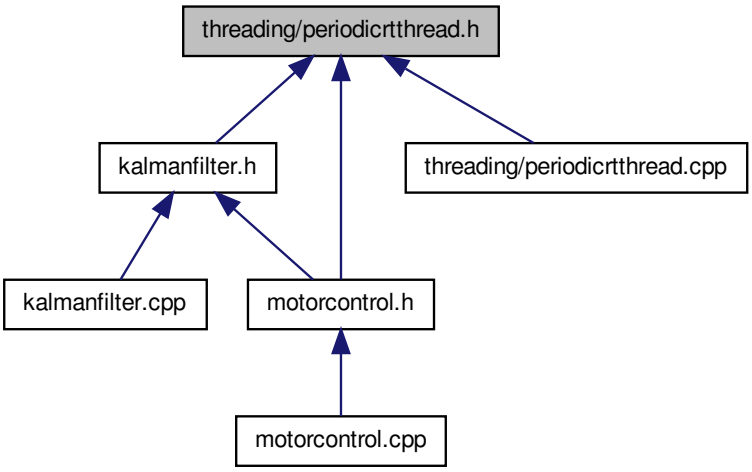
Definition in file periodicrtthread.cpp.

7.31 threading/periodicrtthread.h File Reference

`#include "RtThread.h"` Include dependency graph for periodicrtthread.h:



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



Classes

- class USU::PeriodicRtThread

  TODO: Make some proper exceptions.

Namespaces

- namespace USU

  TODO: Make some proper exceptions.

7.31.1 Detailed Description

Small C++ wrapper class to create a realtime scheduled pthread with periodic timer events.

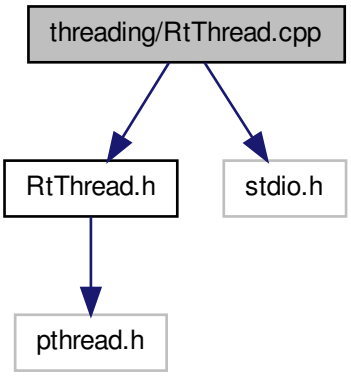
Author

Jan Sommer Created on: Apr 10, 2013

Definition in file periodicrtthread.h.

7.32 threading/RtThread.cpp File Reference

`#include "RtThread.h"` `#include <stdio.h>` Include dependency graph for RtThread.cpp:



7.32.1 Detailed Description

Small C++ wrapper class to create a realtime scheduled pthread

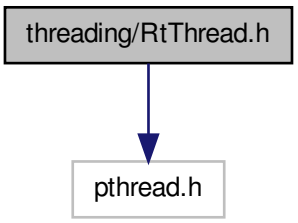
Author

Jan Sommer Created on: Apr 10, 2013

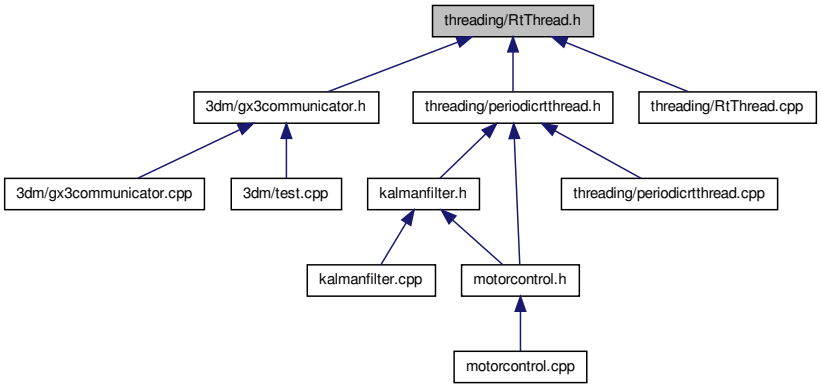
Definition in file RtThread.cpp.

7.33 threading/RtThread.h File Reference

`#include <pthread.h>` Include dependency graph for RtThread.h:



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



Classes

- class USU::RtThread

Abstract wrapper class for the pthread library with RT-priority.

Namespaces

- namespace USU

  TODO: Make some proper exceptions.

7.33.1 Detailed Description

Small C++ wrapper class to create a realtime scheduled pthread

Author

Jan Sommer Created on: Apr 10, 2013

Definition in file RtThread.h.