TM4C IMU

1

Generated by Doxygen 1.8.17

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 acc_t Struct Reference

Accelerometer data.

`#include <imu.h>`

Data Fields

- float x
  Accelerometer X component.
- float y
  Accelerometer Y component.
- float z
  Accelerometer X component.
- float magnitude
  Accelerometer vector magnitude.

3.1.1 Detailed Description

Accelerometer data.

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

- imu.h

3.2 calib_t Struct Reference

Calibration status.

`#include <imu.h>`

Data Fields

- uint8_t mag
  Magnetometer calibration status.
- uint8_t acc
  Accelerometer calibration status.
- uint8_t gyr
  Gyroscope calibration status.
- uint8_t sys
  System calibration status.

3.2.1 Detailed Description

Calibration status.

a 3 on any of the feilds indicates full calibration of that device.

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

- imu.h

3.3 imu_info_t Struct Reference

IMU Info packet.

`#include <imu.h>`

Data Fields

- uint8_t chipID
  Chip ID.
- uint8_t accID
  Accelerometer ID.
- uint8_t magID
  Magnetometer ID.
- uint8_t gyrID
  Gyroscope ID.
- uint16_t swRevID
  Software Revision ID.
- uint8_t blRevID
  Bootloader Revision.

3.3.1 Detailed Description

IMU Info packet.

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

- imu.h

3.4 mag_t Struct Reference

Magnetometer data.

`#include <imu.h>`

Data Fields

- int16_t x
  Magnetometer X component.
- int16_t y
  Magnetometer Y component.
- int16_t z
  Magnetometer Z component.
- float heading
  Magnetometer heading, given in degrees.

3.4.1 Detailed Description

Magnetometer data.

3.4.2 Field Documentation

3.4.2.1 heading

`float mag_t::heading`

Magnetometer heading, given in degrees.

0 Degrees represents north

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

- imu.h

Chapter 4

File Documentation

4.1 i2c.c File Reference

```
#include <i2c.h>
#include <stdlib.h>
```

Functions

- void i2c_init ()

  Initialize I2C1 module.
- uint8_t i2c_requestByte (uint8_t addr, uint8_t request)

  Transmits a request, then receives with a repeated START.
- size_t i2c_sendByte (uint8_t addr, uint8_t data)

  Sends one byte of data.
- size_t i2c_sendBytes (uint8_t addr, uint8_t *data, size_t dataLen)

  Sends multiple bytes of data.
- uint8_t i2c_recByte (uint8_t addr)

  Receives a byte of data from a slave.
- uint8_t * i2c_recBytes (uint8_t addr, size_t dataLen)

  Receives multiple bytes of data from a slave.

4.1.1 Detailed Description

Library for I2C on TM4C123GH6PM Microcontroller

Author

Braedon Giblin bgiblin@iastate.edu

4.1.2 Function Documentation

4.1.2.1 i2c_init()

```
void i2c_init ( )
```

Initialize I2C1 module.

Initializes I2C1 module as a master at standard speed.

4.1.2.2 i2c_recByte()

```
uint8_t i2c_recByte (
            uint8_t addr )
```

Receives a byte of data from a slave.

Parameters

addr	Address to read from

Returns

byte received

4.1.2.3 i2c_recBytes()

```
uint8_t* i2c_recBytes (
            uint8_t addr,
            size_t dataLen )
```

Receives multiple bytes of data from a slave.

Parameters

addr	Address to read from
dataLen	number of bytes to read

Returns

bytes received

4.1.2.4 i2c_requestByte()

```
uint8_t i2c_requestByte (
            uint8_t addr,
            uint8_t request )
```

Transmits a request, then receives with a repeated START.

Parameters

addr	Address to request from
request	Request byte

4.1.2.5 i2c_sendByte()

```
size_t i2c_sendByte (
            uint8_t addr,
            uint8_t data )
```

Sends one byte of data.

Automatically sends start and stop bits.

Parameters

addr	7 bit address to send data to
data	Byte of data to transmit
sendStop	Whether or not to send a stop sequence

Returns

Number of bytes sent (1 if successful, else 0)

4.1.2.6 i2c_sendBytes()

```
size_t i2c_sendBytes (
            uint8_t addr,
            uint8_t * data,
            size_t dataLen )
```

Sends multiple bytes of data.

Automatically sends start and stop bits.

Parameters

addr	7 bit address to send data to
byte	Byte of data to transmit
dataLen	Number of data bits to send

Returns

Number of bytes sent (if successful, should match dataLen

4.2 i2c.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include <stddef.h>
#include <inc/tm4c123gh6pm.h>
```

Functions

- void i2c_init ()
  Initialize I2C1 module.
- size_t i2c_sendByte (uint8_t addr, uint8_t data)
  Sends one byte of data.
- size_t i2c_sendBytes (uint8_t addr, uint8_t *data, size_t dataLen)
  Sends multiple bytes of data.
- uint8_t i2c_recByte (uint8_t addr)
  Receives a byte of data from a slave.
- uint8_t * i2c_recBytes (uint8_t addr, size_t dataLen)
  Receives multiple bytes of data from a slave.
- uint8_t i2c_requestByte (uint8_t addr, uint8_t request)
  Transmits a request, then receives with a repeated START.

4.2.1 Detailed Description

Library for I2C on TM4C123GH6PM Microcontroller.

Author

Braedon Giblin bgiblin@iastate.edu

4.2.2 Function Documentation

4.2.2.1 i2c_init()

```
void i2c_init ( )
```

Initialize I2C1 module.

Initializes I2C1 module as a master at standard speed.

4.2.2.2 i2c_recByte()

```
uint8_t i2c_recByte (
            uint8_t addr )
```

Receives a byte of data from a slave.

Parameters

addr	Address to read from

Returns

byte received

4.2.2.3 i2c_recBytes()

```
uint8_t* i2c_recBytes (
            uint8_t addr,
            size_t dataLen )
```

Receives multiple bytes of data from a slave.

Parameters

addr	Address to read from
dataLen	number of bytes to read

Returns

bytes received

4.2.2.4 i2c_requestByte()

```
uint8_t i2c_requestByte (
            uint8_t addr,
            uint8_t request )
```

Transmits a request, then receives with a repeated START.

Parameters

addr	Address to request from
request	Request byte

4.2.2.5 i2c_sendByte()

```
size_t i2c_sendByte (
```

```
            uint8_t addr,
            uint8_t data )
```

Sends one byte of data.

Automatically sends start and stop bits.

Parameters

addr	7 bit address to send data to
data	Byte of data to transmit
sendStop	Whether or not to send a stop sequence

Returns

Number of bytes sent (1 if successful, else 0)

4.2.2.6 i2c_sendBytes()

```
size_t i2c_sendBytes (
            uint8_t addr,
            uint8_t * data,
            size_t dataLen )
```

Sends multiple bytes of data.

Automatically sends start and stop bits.

Parameters

addr	7 bit address to send data to
byte	Byte of data to transmit
dataLen	Number of data bits to send

Returns

Number of bytes sent (if successful, should match dataLen

4.3 imu.c File Reference

```
#include <stdlib.h>
#include <math.h>
#include "i2c.h"
#include "imu.h"
#include "timer.h"
```

Functions

- void imu_init ()
  Initialize IMU.
- void imu_setMode (imu_mode_t mode)
  sets the mode of the IMU.
- imu_mode_t **imu_getMode** ()
- mag_t * imu_getMag ()
  Gets the current magnetometer data from the IMU.
- acc_t * imu_getAcc ()
  Gets the current acceleration data from the IMU.
- acc_t * **imu_getLinAcc** ()
- void imu_setDefaultUnits ()
  Sets the default units.
- bool imu_connected ()
  Check if IMU is connected.
- uint8_t imu_getStatus ()
  Gets the current status from the IMU.
- int imu_getTemp ()
  Gets the current temperature reading from IMU.
- imu_info_t * imu_getChipInfo ()
  gets the current Chip Info.
- void imu_reset ()
  Resets the IMU.
- void imu_writeReg (uint8_t regAddr, uint8_t val)
  Writes to a specified register.
- uint8_t imu_readRegByte (uint8_t regAddr)
  Reads a byte from the IMU.
- uint8_t * imu_readRegBytes (uint8_t regAddr, size_t dataLen)
  Reads a multiple bytes from the IMU.

4.3.1 Detailed Description

Driver for BNO055 IMU.

Author

Braedon Giblin bgiblin@iastate.edu

4.3.2 Function Documentation

4.3.2.1 imu_connected()

```
bool imu_connected ( )
```

Check if IMU is connected.

Returns

True if connected, else false

4.3.2.2 imu_getAcc()

```
acc_t* imu_getAcc ( )
```

Gets the current acceleration data from the IMU.

Returns

Accelerometer data packet

4.3.2.3 imu_getChipInfo()

```
imu_info_t* imu_getChipInfo ( )
```

gets the current Chip Info.

Returns

Chip info packet.

4.3.2.4 imu_getMag()

```
mag_t* imu_getMag ( )
```

Gets the current magnetometer data from the IMU.

Returns

Magnetometer data packet

4.3.2.5 imu_getStatus()

```
uint8_t imu_getStatus ( )
```

Gets the current status from the IMU.

Returns

status. See data sheet.

4.3.2.6 imu_getTemp()

```
int imu_getTemp ( )
```

Gets the current temperature reading from IMU.

Returns

Temperature in specified units (default fahrenheit)

4.3.2.7 imu_readRegByte()

```
uint8_t imu_readRegByte (
            uint8_t regAddr )
```

Reads a byte from the IMU.

Parameters

regAddr	Address to read from

Returns

Byte read

4.3.2.8 imu_readRegBytes()

```
uint8_t* imu_readRegBytes (
            uint8_t regAddr,
            size_t dataLen )
```

Reads a multiple bytes from the IMU.

NOTE This allocates memory. Allocated array **MUST** be freed.

Parameters

regAddr	Address to begin read from
dataLen	Number of bytes to read

Returns

Byte array of bytes read.

4.3.2.9 imu_reset()

```
void imu_reset ( )
```

Resets the IMU.

Poll connected to wait until connected.

4.3.2.10 imu_setDefaultUnits()

```
void imu_setDefaultUnits ( )
```

Sets the default units.

Default units are - M/s^2, degrees, fahrenheit

4.3.2.11 imu_setMode()

```
void imu_setMode (
            imu_mode_t mode )
```

sets the mode of the IMU.

See imu_mode_t struct for different modes.

4.3.2.12 imu_writeReg()

```
void imu_writeReg (
            uint8_t regAddr,
            uint8_t val )
```

Writes to a specified register.

Parameters

regAddr	Register address to write
val	Value to write

4.4 imu.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include <stddef.h>
#include <inc/tm4c123gh6pm.h>
```

Data Structures

- struct calib_t

  Calibration status.
- struct mag_t

  Magnetometer data.
- struct acc_t

  Accelerometer data.
- struct imu_info_t

  IMU Info packet.

Macros

- #define BNO055_ADDRESS_A (0x28)

  BNO055 Address A.
- #define BNO055_ADDRESS_B (0x29)

  BNO055 Address B.

- #define BNO055_ID (0xA0)

  BNO055 ID.
- #define **IMU_CHIP_ID** 0x00
- #define **IMU_ACC_ID** 0x01
- #define **IMU_MAG_ID** 0x02
- #define **IMU_GYR_ID_R** 0x03
- #define **IMU_SW_REV_MSB** 0x04
- #define **IMU_SW_REV_LSB** 0x05
- #define **IMU_BL_VER_R** 0x06
- #define **IMU_PAGE_ID** 0x07
- #define **IMU_ACC_DATAX_LSB** 0x08
- #define **IMU_ACC_DATAX_MSB** 0x09
- #define **IMU_ACC_DATAY_LSB** 0x0A
- #define **IMU_ACC_DATAY_MSB** 0x0B
- #define **IMU_ACC_DATAZ_LSB** 0x0C
- #define **IMU_ACC_DATAZ_MSB** 0x0D
- #define **IMU_MAG_DATAX_LSB** 0x0E
- #define **IMU_MAG_DATAX_MSB** 0x0F
- #define **IMU_MAG_DATAY_LSB** 0x10
- #define **IMU_MAG_DATAY_MSB** 0x11
- #define **IMU_MAG_DATAZ_LSB** 0x12
- #define **IMU_MAG_DATAZ_MSB** 0x13
- #define **IMU_GYR_DATAX_LSB** 0x14
- #define **IMU_GYR_DATAX_MSB** 0x15
- #define **IMU_GYR_DATAY_LSB** 0x16
- #define **IMU_GYR_DATAY_MSB** 0x17
- #define **IMU_GYR_DATAZ_LSB** 0x18
- #define **IMU_GYR_DATAZ_MSB** 0x19
- #define **IMU_EUL_HEAD_LSB** 0x1A
- #define **IMU_EUL_HEAD_MSB** 0x1B
- #define **IMU_EUL_ROLL_LSB** 0x1C
- #define **IMU_EUL_ROLL_MSB** 0x1D
- #define **IMU_EUL_PTCH_LSB** 0x1E
- #define **IMU_EUL_PTCH_MSB** 0x1F
- #define **IMU_QUA_DATAW_LSB** 0x20
- #define **IMU_QUA_DATAW_MSB** 0x21
- #define **IMU_QUA_DATAX_LSB** 0x22
- #define **IMU_QUA_DATAX_MSB** 0x23
- #define **IMU_QUA_DATAY_LSB** 0x24
- #define **IMU_QUA_DATAY_MSB** 0x25
- #define **IMU_QUA_DATAZ_LSB** 0x26
- #define **IMU_QUA_DATAZ_MSB** 0x27
- #define **IMU_LIA_DATAX_LSB** 0x28
- #define **IMU_LIA_DATAX_MSB** 0x29
- #define **IMU_LIA_DATAY_LSB** 0x2A
- #define **IMU_LIA_DATAY_MSB** 0x2B
- #define **IMU_LIA_DATAZ_LSB** 0x2C
- #define **IMU_LIA_DATAZ_MSB** 0x2D
- #define **IMU_GRV_DATAX_LSB** 0x2E
- #define **IMU_GRV_DATAX_MSB** 0x2F
- #define **IMU_GRV_DATAY_LSB** 0x30
- #define **IMU_GRV_DATAY_MSB** 0x31
- #define **IMU_GRV_DATAZ_LSB** 0x32
- #define **IMU_GRV_DATAZ_MSB** 0x33

- #define **IMU_TEMP** 0x34
- #define **IMU_CALIB_STAT** 0x35
- #define **IMU_SYS_STATUS** 0x39
- #define **IMU_UNIT_SEL** 0x3B
- #define **IMU_OPR_MODE** 0x3D

Enumerations

- enum imu_mode_t {
  CONFIG, ACCONLY, MAGONLY, GYROONLY,
  ACCMAG, ACCGYRO, MAGGYRO, AMG,
  IMU, COMPASS, M4G, NDOF_FMC_OFF,
  NDOF }

  IMU Operating Modes.

Functions

- void imu_init ()

  Initialize IMU.
- bool imu_connected ()

  Check if IMU is connected.
- void imu_reset ()

  Resets the IMU.
- int imu_getTemp ()

  Gets the current temperature reading from IMU.
- mag_t * imu_getMag ()

  Gets the current magnetometer data from the IMU.
- acc_t * imu_getAcc ()

  Gets the current acceleration data from the IMU.
- acc_t * imu_getLinAcc ()

  Gets the linear acceleration data from IMU.
- uint8_t imu_getStatus ()

  Gets the current status from the IMU.
- void imu_writeReg (uint8_t regAddr, uint8_t val)

  Writes to a specified register.
- uint8_t imu_readRegByte (uint8_t regAddr)

  Reads a byte from the IMU.
- uint8_t * imu_readRegBytes (uint8_t regAddr, size_t dataLen)

  Reads a multiple bytes from the IMU.
- imu_info_t * imu_getChipInfo ()

  gets the current Chip Info.
- void imu_setDefaultUnits ()

  Sets the default units.
- void imu_setMode (imu_mode_t mode)

  sets the mode of the IMU.

4.4.1 Detailed Description

Driver for BNO055 IMU.

Author

Braedon Giblin bgiblin@iastate.edu

4.4.2 Enumeration Type Documentation

4.4.2.1 imu_mode_t

```
enum imu_mode_t
```

IMU Operating Modes.

Enumerator

CONFIG	Config mode.
ACCONLY	Non-fusion Accelerometer only mode.
MAGONLY	Non-fusion Magnetometer only mode.
GYROONLY	Non-fusion Gyroscope only mode.
ACCMAG	Non-fusion Accelerometer and Magnetometer mode.
ACCGYRO	Non-fusion Accelerometer and Gyrosocpe mode.
MAGGYRO	Non-fusion Gyroscope and Magnetometer mode.
AMG	Non-fusion Accelerometer, Gyroscope, and Magnetometer mode.
IMU	Fusion Accelerometer and Gyrosocope – relative.
COMPASS	Fusion Accelerometer and Magnetometer – absolute.
M4G	Fusion Accelerometer and magnetometer – relative.
NDOF_FMC_OFF	Fusion, all devices, no fast calibration.
NDOF	Fusion, all devices.

4.4.3 Function Documentation

4.4.3.1 imu_connected()

```
bool imu_connected ( )
```

Check if IMU is connected.

Returns

True if connected, else false

4.4.3.2 imu_getAcc()

```
acc_t* imu_getAcc ( )
```

Gets the current acceleration data from the IMU.

Returns

Accelerometer data packet

4.4.3.3 imu_getChipInfo()

```
imu_info_t* imu_getChipInfo ( )
```

gets the current Chip Info.

Returns

Chip info packet.

4.4.3.4 imu_getLinAcc()

```
acc_t* imu_getLinAcc ( )
```

Gets the linear acceleration data from IMU.

IMU must be in Fusion mode to read this value.

Returns

linear acceration data.

4.4.3.5 imu_getMag()

```
mag_t* imu_getMag ( )
```

Gets the current magnetometer data from the IMU.

Returns

Magnetometer data packet

4.4.3.6 imu_getStatus()

```
uint8_t imu_getStatus ( )
```

Gets the current status from the IMU.

Returns

status. See data sheet.

4.4.3.7 imu_getTemp()

```
int imu_getTemp ( )
```

Gets the current temperature reading from IMU.

Returns

Temperature in specified units (default fahrenheit)

4.4.3.8 imu_readRegByte()

```
uint8_t imu_readRegByte (
            uint8_t regAddr )
```

Reads a byte from the IMU.

Parameters

regAddr	Address to read from

Returns

Byte read

4.4.3.9 imu_readRegBytes()

```
uint8_t* imu_readRegBytes (
            uint8_t regAddr,
            size_t dataLen )
```

Reads a multiple bytes from the IMU.

NOTE This allocates memory. Allocated array **MUST** be freed.

Parameters

regAddr	Address to begin read from
dataLen	Number of bytes to read

Returns

Byte array of bytes read.

4.4.3.10 imu_reset()

```
void imu_reset ( )
```

Resets the IMU.

Poll connected to wait until connected.

4.4.3.11 imu_setDefaultUnits()

```
void imu_setDefaultUnits ( )
```

Sets the default units.

Default units are - M/s^2, degrees, fahrenheit

4.4.3.12 imu_setMode()

```
void imu_setMode (
            imu_mode_t mode )
```

sets the mode of the IMU.

See imu_mode_t struct for different modes.

4.4.3.13 imu_writeReg()

```
void imu_writeReg (
            uint8_t regAddr,
            uint8_t val )
```

Writes to a specified register.

Parameters

regAddr	Register address to write
val	Value to write

Index