mpu925x-driver

Release 0.1

Ceyhun Şen

Aug 12, 2022

CONTENTS:

Welcome to the documentation for MPU-925X driver. This driver supports MPU-9250 and MPU-9255 sensors.

Source code is available at Github.

CHAPTER

ONE

UNITS

- Acceleration from accelerometer: G
- Rotation from gyroscope: Degrees per second
- Magnetic field from magnetometer: Micro Gauss
- Temperature from thermometer: Celsius degree

CHAPTER

TWO

COMPATIBILITY

This driver is designed to be portable. So, it can be used on any platform that has a C compiler which supports `C99` standard. See *porting guide* and `examples` directory for more information.

2.1 Getting Started

2.1.1 Adding to Your Project

1. Copy `mpu925x-driver` directory to your project's `drivers` directory.
2. Add `inc` directory to your toolchain's include path.
3. Add `src/mpu925x_core.c`, `src/mpu925x_settings.c` and `src/mpu925x_internals.c` source files to your project's build toolchain.
4. Provide bus handle, bus read, bus write and delay functions depending on your platform (see: *porting guide*).
5. Include `mpu925x.h` header to your desired source files.
6. [EXTRAS] Extra modules can be compiled with program if any of the extra functionalities needed. Extra modules are located in `extras` directory.

2.1.2 Simple Usage

Listing 1: Example Code

```
#include "mpu925x.h"

// Create mpu925x_t struct instance.
mpu925x_t mpu925x = {
        .master_specific = {
                .bus_handle = &my_bus_handle,

                // Bus functions
                .bus_read = my_bus_read_function_pointer,
                .bus_write = my_bus_write_function_pointer,
                .delay_ms = my_delay_function_pointer
        },

        .settings = {
                // Other settings
```

(continues on next page)

(continued from previous page)

```
                .accelerometer_scale = mpu925x_2g,
                .gyroscope_scale = mpu925x_250dps,
                .orientation = mpu925x_z_plus
        }
};

// Wait till' initializition is complete. Will be in endless loop if sensor
// is unreachable (wiring is not correct, sensor is damaged...).
while (mpu925x_init(&mpu925x, 0));

while (1) {
        // Get sensor data.
        mpu925x_get_all(&mpu925x);

        // Use sensor data (e.g. print).
        printf("Acceleration: %f, %f, %f\n"
               "Rotation: %f, %f, %f\n"
               "Magnetic field: %f, %f, %f\n",
               mpu925x.sensor_data.acceleration[0], mpu925x.sensor_data.acceleration[1],␣
↪mpu925x.sensor_data.acceleration[2],
               mpu925x.sensor_data.rotation[0], mpu925x.sensor_data.rotation[1], mpu925x.
↪sensor_data.rotation[2],
               mpu925x.sensor_data.magnetic_field[0], mpu925x.sensor_data.magnetic_
↪field[1], mpu925x.sensor_data.magnetic_field[2],);
}
```

See *advanced usage* for more functionalities (e.g. hardware offset cancellation, sensor ranges…).

2.2 Porting Guide

This driver needs:

- Bus read
- Bus write
- Delay microseconds
- Bus handle (optional, depending on your platform)

function and struct pointers. You must define your own functions and pass their pointers to `mpu925x_t` struct before initialization.

Given slave address is 7 bit, so if your bus interface needs an 8 bit address, shift slave address to 1 bit left (e.g. address << 1).

2.2.1 Bus Read Function

Bus read functions' prototype is like this:

```
uint8_t (*bus_read)(struct mpu925x_t *mpu925x, uint8_t slave_address, uint8_t reg, uint8_
↪t *buffer, uint8_t size);
```

This function wants to read `size` byte amount of data from bus slave's `reg` register which address is `slave_address` and stores it in `buffer` array.

Arduino I2C example:

```
uint8_t arduino_i2c_read(mpu925x_t *mpu925x, uint8_t slave_address, uint8_t reg, uint8_t␣
↪*buffer, uint8_t size)
{
  Wire.begin();
  Wire.beginTransmission(slave_address);
  Wire.write(reg);
  Wire.endTransmission(false);
  Wire.requestFrom(slave_address, size);
  for (uint16_t i = 0; i < size; i++) {
        if (Wire.available()) {
          buffer[i] = (uint8_t)Wire.read();
        }
        else {
          return 1;
        }
  }
  return 0;
}
```

STM32 HAL I2C example:

```
uint8_t mpu925x_stm32_i2c_hal_read(mpu925x_t *mpu925x, uint8_t slave_address, uint8_t␣
↪reg, uint8_t *buffer, uint8_t size)
{
        return HAL_I2C_Mem_Read(mpu925x->master_specific.bus_handle, slave_address << 1,␣
↪reg, 1, buffer, size, HAL_MAX_DELAY);
}

mpu925x.master_specific.bus_read = mpu925x_stm32_i2c_hal_read;
```

2.2.2 Bus Write Function

Bus write function's prototype is like this:

```
uint8_t (*bus_write)(struct mpu925x_t *mpu925x, uint8_t slave_address, uint8_t reg,␣
↪uint8_t *buffer, uint8_t size);
```

This function wants to write `size` byte amount of data from `buffer` array to bus slave's `reg` register which address is `slave_address`.

Arduino I2C example:

```
uint8_t arduino_i2c_write(mpu925x_t *mpu925x, uint8_t slave_address, uint8_t reg, uint8_
↪t *buffer, uint8_t size)
{
  Wire.beginTransmission(slave_address);
  Wire.write(reg);
  for (uint16_t i = 0; i < size; i++) {
        Wire.write(buffer[i]);
  }
  Wire.endTransmission();
  return 0;
}
```

STM32 HAL I2C example:

```
uint8_t mpu925x_stm32_i2c_hal_write(mpu925x_t *mpu925x, uint8_t slave_address, uint8_t␣
↪reg, uint8_t *buffer, uint8_t size)
{
        return HAL_I2C_Mem_Write(mpu925x->master_specific.bus_handle, slave_address << 1,
↪ reg, 1, buffer, size, HAL_MAX_DELAY);
}

mpu925x.master_specific.bus_write = mpu925x_stm32_i2c_hal_write;
```

2.2.3 Delay Microseconds Function

Delay microseconds functions prototype is like this:

```
void (*delay_ms)(struct mpu925x_t *mpu925x, uint32_t delay);
```

This functions wants to wait at least `delay` microseconds.

Arduino I2C example:

```
void arduino_delay_ms(mpu925x_t *mpu925x, uint32_t dly)
{
  delay(dly);
}
```

STM32 HAL example:

```
void mpu925x_stm32_hal_delay_ms(mpu925x_t *mpu925x, uint32_t delay)
{
        HAL_Delay(delay);
}

mpu925x.master_specific.delay_ms = mpu925x_stm32_hal_delay_ms;
```

2.2.4 Bus Handle Struct

Bus handle is for platform specific structs or other types of data. If your platform needs some other information for bus or delay functions, you can pass the pointer and use it in bus or delay functions via `mpu925x.master_specific.bus_handle`.

STM32 HAL example:

```
I2C_HandleTypeDef hi2c1;

mpu925x.master_specific.bus_handle = &hi2c1;
```

2.3 Advanced Usage

2.3.1 Initialization

Initialization steps are following:

1. Define needed functions and pass their pointers (See: *porting guide*).
2. Create `mpu925x_t` struct instance with initial values (See: *accelerometer*, *gyroscope* and *magnetometer*).
3. Call mpu925x_init() function with created `mpu925x_t` struct and AD0 pin values.

mpu925x_init() Function

Init function takes `mpu925x_t` struct and AD0 pin values as parameters. AD0 pin value depends on physical sensor and is most probably 0. See *api reference* for more info.

Init function will return 0 on success, 1 on accelerometer and gyroscope fail and 2 on magnetometer fail. One can use accelerometer and gyroscope with return value of 2. But with return value of 1, nothing works, so check wiring and sensor damage.

2.3.2 General Settings

Sample Rate Divider

Internal sample rate divider is configurable. See MPU-9250 or MPU-9255's datasheet for more info.

void **mpu925x_set_sample_rate_divider**(*mpu925x_t* *mpu925x, uint8_t sample_rate_divider)

Set sample rate divider.

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **sample_rate_divider** – Sample rate divider sentence.

Clock Source

Internal clock source is configurable.

void **mpu925x_set_clock_source**(*mpu925x_t* *mpu925x, *mpu925x_clock* clock)

Set clock source.

See also:

mpu925x_clock

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **clock** – Clock select option.

enum **mpu925x_clock**

Clock settings for MPU-925X.

Values:

enumerator **mpu925x_internal_20_hz_clock**

enumerator **mpu925x_auto_select_pll**

2.3.3 Accelerometer Advanced Usage

Raw And Translated Data

Raw and translated acceleration datas are available. Calling `mpu925x_get_acceleration` function also calls `mpu925x_get_acceleration_raw` function internally. Both functions stores data in given `mpu925x_t` struct.

void **mpu925x_get_acceleration**(*mpu925x_t* *mpu925x)

Get acceleration in G's.

See also:

mpu925x_get_acceleration_raw

Parameters

mpu925x – MPU-925X struct pointer.

void **mpu925x_get_acceleration_raw**(*mpu925x_t* *mpu925x)

Get raw acceleration data.

See also:

mpu925x_get_acceleration

Parameters

mpu925x – MPU-925X struct pointer.

Full-Scale Select

One can set accelerometer full-scale range of 2g, 4g, 8g and 16g.

void **mpu925x_set_accelerometer_scale**(*mpu925x_t* *mpu925x, *mpu925x_accelerometer_scale* scale)

Set accelerometer full-scale range.

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **scale** – Accelerometer full-scale range to be set.

enum **mpu925x_accelerometer_scale**

Accelerometer full-scale ranges.

Values:

enumerator **mpu925x_2g**

enumerator **mpu925x_4g**

enumerator **mpu925x_8g**

enumerator **mpu925x_16g**

Digital Low Pass Filter

One can enable hardware digital low pass filter. Look datasheet for more info.

void **mpu925x_set_accelerometer_dlpf**(*mpu925x_t* *mpu925x, uint8_t a_fchoice, uint8_t dlpf)

Set accelerometer digital low pass filter setting.

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **a_fchoice** – Accelerometer fchoice bit.
- **dlpf** – Digital low pass filter choice.

Offset Cancellation

Offset cancellation consist 2 steps: Getting and setting offset cancellation values. While getting offset cancellation values, sensor must stand still. Also orientation of sensor is very important. Set orientation of sensor with `.sensor_settings.orientation` variable in `mpu925x_t` struct.

enum **mpu925x_orientation**

Orientation of the sensor.

Values:

enumerator **mpu925x_x_plus**

enumerator **mpu925x_x_minus**

enumerator **mpu925x_y_plus**

enumerator **mpu925x_y_minus**

enumerator **mpu925x_z_plus**

enumerator **mpu925x_z_minus**

void **mpu925x_get_accelerometer_offset**(*mpu925x_t* *mpu925x, uint16_t sampling_amount, int16_t *offset)

Get accelerometer offset cancellation value.

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **sampling_amount** – Sampling amount for acceleration values.
- **offset** – 3d array which will hold accelerometer offset cancellation values.

void **mpu925x_set_accelerometer_offset**(*mpu925x_t* *mpu925x, int16_t *offset)

Set accelerometer offset cancellation value.

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **offset** – 3d array which holds accelerometer offset cancellation values.

One can use these two methods to get and set offset values or call one unified function which does 2 of them at once.

void **mpu925x_accelerometer_offset_cancellation**(*mpu925x_t* *mpu925x, uint16_t sampling_amount)

Get and set accelerometer offset cancellation values.

See also:

mpu925x_get_accelerometer_offset

See also:

mpu925x_set_accelerometer_offset

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **sampling_amount** – Sampling amount for acceleration values.

Listing 2: Example Code

```
mpu925x.settings.orientation = mpu925x_y_plus; // Depends on how sensor is mounted.
// Sensor must stand still while offset cancellation.
mpu925x_accelerometer_offset_cancellation(&mpu925x, 200);
```

2.3.4 Gyroscope Advanced Usage

Raw And Translated Data

Raw and translated rotation datas are available. Calling `mpu925x_get_rotation` function also calls `mpu925x_get_rotation_raw` function internally. Both functions stores data in given `mpu925x_t` struct.

void **mpu925x_get_rotation**(*mpu925x_t* *mpu925x)

Get rotation in degrees per second.

See also:

mpu925x_get_rotation_raw

Parameters

mpu925x – MPU-925X struct pointer.

void **mpu925x_get_rotation_raw**(*mpu925x_t* *mpu925x)

Get raw rotation data.

See also:

mpu925x_get_rotation

Parameters

mpu925x – MPU-925X struct pointer.

Full-Scale Select

One can set gyroscope full-scale range of 250 dps, 500 dps, 1000 dps and 2000 dps.

void **mpu925x_set_gyroscope_scale**(*mpu925x_t* *mpu925x, *mpu925x_gyroscope_scale* scale)

Set gyroscope full-scale range.

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **scale** – Gyroscope full-scale range to be set.

enum **mpu925x_gyroscope_scale**

Gyroscope full-scale ranges for gyroscope.

Values:

enumerator **mpu925x_250dps**

enumerator **mpu925x_500dps**

enumerator **mpu925x_1000dps**

enumerator **mpu925x_2000dps**

Digital Low Pass Filter

One can enable hardware digital low pass filter. Look datasheet for more info.

void **mpu925x_set_gyroscope_dlpf**(*mpu925x_t* *mpu925x, uint8_t a_fchoice, uint8_t dlpf)

Set gyroscope digital low pass filter setting.

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **g_fchoice** – Gyroscope f_choice bits.
- **dlpf** – Digital low pass filter setting.

Offset Cancellation

Offset cancellation consist 2 steps: Getting and setting offset cancellation values. While getting offset cancellation values, sensor must stand still.

void **mpu925x_get_gyroscope_offset**(*mpu925x_t* *mpu925x, uint16_t sampling_amount, int16_t *offset)

Get gyroscope offset cancellation values.

See also:

mpu925x_set_gyroscope_offset

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **sampling_amount** – Sampling amount for rotation values.
- **offset** – 3d array which holds gyroscope offset cancellation values.

void **mpu925x_set_gyroscope_offset**(*mpu925x_t* *mpu925x, int16_t *offset)

Set gyroscope offset cancellation values.

See also:

mpu925x_get_gyroscope_offset

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **offset** – 3d array which holds gyroscope offset cancellation values.

One can use these two methods to get and set offset values or call one unified function which does 2 of them at once.

void **mpu925x_gyroscope_offset_cancellation**(*mpu925x_t* *mpu925x, uint16_t sampling_amount)

Get and set gyroscope offset cancellation values.

See also:

mpu925x_get_gyroscope_offset

See also:

mpu925x_set_gyroscope_offset

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **sampling_amount** – Sampling amount for rotation values.

2.3.5 Magnetometer Advanced Usage

Raw And Translated Data

Raw and translated magnetic field datas are available. Calling `mpu925x_get_magnetic_field` function also calls `mpu925x_get_magnetic_field_raw` function internally. Both functions stores data in given `mpu925x_t` struct.

void **mpu925x_get_magnetic_field**(*mpu925x_t* *mpu925x)

Get magnetic field in micro Gauss.

Parameters
mpu925x – MPU-925X struct pointer.

void **mpu925x_get_magnetic_field_raw**(*mpu925x_t* *mpu925x)

Get raw magnetic field data.

Parameters
mpu925x – MPU-925X struct pointer.

Measurement Mode

Operation mode is configurable. See AK8963's datasheet for more info.

Warning: Other than `continuous_measurement_mode_1` and `continuous_measurement_mode_2`, operating modes are not tested. Wait for full version of this driver to use them properly.

void **mpu925x_set_magnetometer_measurement_mode**(*mpu925x_t* *mpu925x,
mpu925x_magnetometer_measurement_mode
measurement_mode)

Set magnetometer measurement mode.

See also:

mpu925x_magnetometer_measurement_mode

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **measurement_mode** – Measurement mode for magnetometer to be set.

enum **mpu925x_magnetometer_measurement_mode**

Measurement modes for AK8963.

Values:

enumerator **mpu925x_power_down_mode**

enumerator **mpu925x_single_measurement_mode**

enumerator **mpu925x_continuous_measurement_mode_1**

enumerator **mpu925x_continuous_measurement_mode_2**

enumerator **mpu925x_external_trigger_measurement_mode**

enumerator **mpu925x_self_test_mode**

enumerator **mpu925x_fuse_rom_access_mode**

Bit Mode

Bit mode is configurable. See AK8963's datasheet for more info.

void **mpu925x_set_magnetometer_bit_mode**(*mpu925x_t* *mpu925x, *mpu925x_magnetometer_bit_mode* bit_mode)

Set magnetometer bit mode.

See also:

mpu925x_magnetometer_bit_mode

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **bit_mode** – Bit mode for magnetometer to be set.

enum **mpu925x_magnetometer_bit_mode**

Bit modes for AK8963.

Values:

enumerator **mpu925x_14_bit**

enumerator **mpu925x_16_bit**

2.3.6 Extra Modules

Simple AHRS

Simple attitude and heading reference system module can return pitch and roll angles using acceleration value. Include `mpu925x_simple_ahrs.h` in desired source file and compile `mpu925x_simple_ahrs.c` source file with target program.

Listing 3: Example Code

```
#include "mpu925x.h"
#include "mpu925x_simple_ahrs.h"

// Create struct instances.
mpu925x_t mpu925x;
mpu925x_simple_ahrs ahrs;

// Initialize driver.
mpu925x_init(&mpu925x, 0);

// Get values.
mpu925x_get_acceleration(&mpu925x);
mpu925x_get_simple_ahrs(&mpu925x, &ahrs);

printf("Pitch: %f, Roll: %f\n", ahrs.pitch, ahrs.roll);
```

API Reference

Simple AHRS header file for MPU-925X driver.

Author
Ceyhun Şen

Typedefs

typedef struct *mpu925x_simple_ahrs* **mpu925x_simple_ahrs**

Pitch and roll angles for simple AHRS.

Functions

void **mpu925x_get_simple_ahrs**(*mpu925x_t* *mpu925x, *mpu925x_simple_ahrs* *ahrs)

Get pitch and roll angles.

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **ahrs** – Simple AHRS struct pointer.

struct **mpu925x_simple_ahrs**

#include <mpu925x_simple_ahrs.h> Pitch and roll angles for simple AHRS.

Public Members

float **pitch**

float **roll**

project
mpu925x-driver

2.4 API Reference

Header file for MPU-925X driver.

Author
Ceyhun Şen

Typedefs

typedef enum *mpu925x_clock* **mpu925x_clock**

typedef enum *mpu925x_orientation* **mpu925x_orientation**

typedef enum *mpu925x_accelerometer_scale* **mpu925x_accelerometer_scale**

typedef enum *mpu925x_gyroscope_scale* **mpu925x_gyroscope_scale**

typedef enum *mpu925x_magnetometer_measurement_mode* **mpu925x_magnetometer_measurement_mode**

typedef enum *mpu925x_magnetometer_bit_mode* **mpu925x_magnetometer_bit_mode**

typedef struct *mpu925x_t* **mpu925x_t**

Enums

enum **mpu925x_clock**

Clock settings for MPU-925X.

Values:

enumerator **mpu925x_internal_20_hz_clock**

enumerator **mpu925x_auto_select_pll**

enum **mpu925x_orientation**

Orientation of the sensor.

Values:

enumerator **mpu925x_x_plus**

enumerator **mpu925x_x_minus**

enumerator **mpu925x_y_plus**

enumerator **mpu925x_y_minus**

enumerator **mpu925x_z_plus**

enumerator **mpu925x_z_minus**

enum **mpu925x_accelerometer_scale**

Accelerometer full-scale ranges.

Values:

enumerator **mpu925x_2g**

enumerator **mpu925x_4g**

enumerator **mpu925x_8g**

enumerator **mpu925x_16g**

enum **mpu925x_gyroscope_scale**

Gyroscope full-scale ranges for gyroscope.

Values:

enumerator **mpu925x_250dps**

enumerator **mpu925x_500dps**

enumerator **mpu925x_1000dps**

enumerator **mpu925x_2000dps**

enum **mpu925x_magnetometer_measurement_mode**

Measurement modes for AK8963.

Values:

enumerator **mpu925x_power_down_mode**

enumerator **mpu925x_single_measurement_mode**

enumerator **mpu925x_continuous_measurement_mode_1**

enumerator **mpu925x_continuous_measurement_mode_2**

enumerator **mpu925x_external_trigger_measurement_mode**

enumerator **mpu925x_self_test_mode**

enumerator **mpu925x_fuse_rom_access_mode**

enum **mpu925x_magnetometer_bit_mode**

Bit modes for AK8963.

Values:

enumerator **mpu925x_14_bit**

enumerator **mpu925x_16_bit**

Functions

uint8_t **mpu925x_init**(*mpu925x_t* *mpu925x, uint8_t ad0)

Initialize MPU-925X sensor.

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **ad0** – Last bit of the slave address (depends on ad0 pin connection).

Returns
0 on success, 1 on failure on mpu925x, 2 on failure on AK8963.

void **mpu925x_get_all_raw**(*mpu925x_t* *mpu925x)

Get all raw sensor data at once.

Parameters
mpu925x – MPU-925X struct pointer.

void **mpu925x_get_all**(*mpu925x_t* *mpu925x)

Get all sensor data at once.

Parameters
mpu925x – MPU-925X struct pointer.

void **mpu925x_get_acceleration_raw**(*mpu925x_t* *mpu925x)

Get raw acceleration data.

See also:

mpu925x_get_acceleration

Parameters
mpu925x – MPU-925X struct pointer.

void **mpu925x_get_acceleration**(*mpu925x_t* *mpu925x)

Get acceleration in G's.

See also:

mpu925x_get_acceleration_raw

Parameters
mpu925x – MPU-925X struct pointer.

void **mpu925x_get_rotation_raw**(*mpu925x_t* *mpu925x)

Get raw rotation data.

See also:

mpu925x_get_rotation

Parameters
mpu925x – MPU-925X struct pointer.

void **mpu925x_get_rotation**(*mpu925x_t* *mpu925x)

Get rotation in degrees per second.

See also:

mpu925x_get_rotation_raw

Parameters
mpu925x – MPU-925X struct pointer.

void **mpu925x_get_magnetic_field_raw**(*mpu925x_t* *mpu925x)

Get raw magnetic field data.

Parameters
mpu925x – MPU-925X struct pointer.

void **mpu925x_get_magnetic_field**(*mpu925x_t* *mpu925x)

Get magnetic field in micro Gauss.

Parameters
mpu925x – MPU-925X struct pointer.

void **mpu925x_get_temperature_raw**(*mpu925x_t* *mpu925x)

Get raw temperature data.

Parameters
mpu925x – MPU-925X struct pointer.

void **mpu925x_get_temperature**(*mpu925x_t* *mpu925x)

Get temperature in celsius degree.

Parameters
mpu925x – MPU-925X struct pointer.

void **mpu925x_set_sample_rate_divider**(*mpu925x_t* *mpu925x, uint8_t sample_rate_divider)

Set sample rate divider.

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **sample_rate_divider** – Sample rate divider sentence.

void **mpu925x_set_clock_source**(*mpu925x_t* *mpu925x, *mpu925x_clock* clock)

Set clock source.

See also:

mpu925x_clock

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **clock** – Clock select option.

void **mpu925x_set_accelerometer_scale**(*mpu925x_t* *mpu925x, *mpu925x_accelerometer_scale* scale)

Set accelerometer full-scale range.

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **scale** – Accelerometer full-scale range to be set.

void **mpu925x_set_accelerometer_dlpf**(*mpu925x_t* *mpu925x, uint8_t a_fchoice, uint8_t dlpf)

Set accelerometer digital low pass filter setting.

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **a_fchoice** – Accelerometer fchoice bit.
- **dlpf** – Digital low pass filter choice.

void **mpu925x_accelerometer_offset_cancellation**(*mpu925x_t* *mpu925x, uint16_t sampling_amount)

Get and set accelerometer offset cancellation values.

See also:

mpu925x_get_accelerometer_offset

See also:

mpu925x_set_accelerometer_offset

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **sampling_amount** – Sampling amount for acceleration values.

void **mpu925x_get_accelerometer_offset**(*mpu925x_t* *mpu925x, uint16_t sampling_amount, int16_t *offset)

Get accelerometer offset cancellation value.

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **sampling_amount** – Sampling amount for acceleration values.
- **offset** – 3d array which will hold accelerometer offset cancellation values.

void **mpu925x_set_accelerometer_offset**(*mpu925x_t* *mpu925x, int16_t *offset)

Set accelerometer offset cancellation value.

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **offset** – 3d array which holds accelerometer offset cancellation values.

void **mpu925x_set_gyroscope_scale**(*mpu925x_t* *mpu925x, *mpu925x_gyroscope_scale* scale)

Set gyroscope full-scale range.

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **scale** – Gyroscope full-scale range to be set.

void **mpu925x_set_gyroscope_dlpf**(*mpu925x_t* *mpu925x, uint8_t a_fchoice, uint8_t dlpf)

Set gyroscope digital low pass filter setting.

Parameters

- **mpu925x** – MPU-925X struct pointer.

- **g_fchoice** – Gyroscope f_choice bits.
- **dlpf** – Digital low pass filter setting.

void **mpu925x_gyroscope_offset_cancellation**(*mpu925x_t* *mpu925x, uint16_t sampling_amount)

Get and set gyroscope offset cancellation values.

See also:

mpu925x_get_gyroscope_offset

See also:

mpu925x_set_gyroscope_offset

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **sampling_amount** – Sampling amount for rotation values.

void **mpu925x_get_gyroscope_offset**(*mpu925x_t* *mpu925x, uint16_t sampling_amount, int16_t *offset)

Get gyroscope offset cancellation values.

See also:

mpu925x_set_gyroscope_offset

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **sampling_amount** – Sampling amount for rotation values.
- **offset** – 3d array which holds gyroscope offset cancellation values.

void **mpu925x_set_gyroscope_offset**(*mpu925x_t* *mpu925x, int16_t *offset)

Set gyroscope offset cancellation values.

See also:

mpu925x_get_gyroscope_offset

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **offset** – 3d array which holds gyroscope offset cancellation values.

void **mpu925x_set_magnetometer_measurement_mode**(*mpu925x_t* *mpu925x, *mpu925x_magnetometer_measurement_mode* measurement_mode)

Set magnetometer measurement mode.

See also:

mpu925x_magnetometer_measurement_mode

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **measurement_mode** – Measurement mode for magnetometer to be set.

void **mpu925x_set_magnetometer_bit_mode**(*mpu925x_t* *mpu925x, *mpu925x_magnetometer_bit_mode* bit_mode)

Set magnetometer bit mode.

See also:

mpu925x_magnetometer_bit_mode

Parameters

- **mpu925x** – MPU-925X struct pointer.
- **bit_mode** – Bit mode for magnetometer to be set.

struct **mpu925x_t**

#include <mpu925x.h> Main struct for MPU-925X driver.

This structs includes sensor data, driver settings and master specific handle, bus and delay function pointers.

Public Members

struct *mpu925x_t*::*sensor_data* **sensor_data**

struct *mpu925x_t*::*settings* **settings**

struct *mpu925x_t*::*master_specific* **master_specific**

struct **master_specific**

#include <mpu925x.h> Holds master specific pointers.

Public Members

uint8_t (***bus_read**)(struct *mpu925x_t* *mpu925x, uint8_t slave_address, uint8_t reg, uint8_t *buffer, uint8_t size)

uint8_t (***bus_write**)(struct *mpu925x_t* *mpu925x, uint8_t slave_address, uint8_t reg, uint8_t *buffer, uint8_t size)

void (***delay_ms**)(struct *mpu925x_t* *mpu925x, uint32_t delay)

void ***bus_handle**

struct **sensor_data**

#include <mpu925x.h> Holds sensor data.

Public Members

int16_t **acceleration_raw**[3]

int16_t **rotation_raw**[3]

int16_t **magnet_raw**[3]

int16_t **temperature_raw**

float **acceleration**[3]

float **rotation**[3]

float **magnetic_field**[3]

float **temperature**

struct **settings**

#include <mpu925x.h> Holds sensor settings.

Public Members

mpu925x_orientation **orientation**

mpu925x_accelerometer_scale **accelerometer_scale**

mpu925x_gyroscope_scale **gyroscope_scale**

mpu925x_magnetometer_measurement_mode **measurement_mode**

mpu925x_magnetometer_bit_mode **bit_mode**

float **acceleration_lsb**

float **gyroscope_lsb**

float **magnetometer_lsb**

float **magnetometer_coefficient**[3]

uint8_t **address**

CHAPTER

THREE

LICENSE

This project is licensed under MIT license.

INDEX

M