IMU Based Controller

Generated by Doxygen 1.9.4

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Module Documentation

4.1 CMSIS

Modules

- Stm32f0xx_system

4.1.1 Detailed Description

4.2 Stm32f0xx_system

Modules

- STM32F0xx_System_Private_Includes
- STM32F0xx_System_Private_TypesDefinitions
- STM32F0xx_System_Private_Defines
- STM32F0xx_System_Private_Macros
- STM32F0xx_System_Private_Variables
- STM32F0xx_System_Private_FunctionPrototypes
- STM32F0xx_System_Private_Functions

4.2.1 Detailed Description

4.3 STM32F0xx_System_Private_Includes

4.4 STM32F0xx_System_Private_TypesDefinitions

4.5 STM32F0xx_System_Private_Defines

Macros

- #define HSE_VALUE ((uint32_t)8000000)
- #define HSI_VALUE ((uint32_t)8000000)
- #define HSI48_VALUE ((uint32_t)48000000)

4.5.1 Detailed Description

4.5.2 Macro Definition Documentation

4.5.2.1 HSE_VALUE

```
#define HSE_VALUE ((uint32_t)8000000)
```

Default value of the External oscillator in Hz. This value can be provided and adapted by the user application.

4.5.2.2 HSI48_VALUE

```
#define HSI48_VALUE ((uint32_t)48000000)
```

Default value of the HSI48 Internal oscillator in Hz. This value can be provided and adapted by the user application.

4.5.2.3 HSI_VALUE

```
#define HSI_VALUE ((uint32_t)8000000)
```

Default value of the Internal oscillator in Hz. This value can be provided and adapted by the user application.

4.6 STM32F0xx_System_Private_Macros

4.7 STM32F0xx_System_Private_Variables

Variables

- uint32_t **SystemCoreClock** = 8000000
- const uint8_t **AHBPrescTable** [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t **APBPrescTable** [8] = {0, 0, 0, 0, 1, 2, 3, 4}

4.7.1 Detailed Description

4.8 STM32F0xx_System_Private_FunctionPrototypes

4.9 STM32F0xx_System_Private_Functions

Functions

- void SystemInit (void)

  Setup the microcontroller system.
- void SystemCoreClockUpdate (void)

  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

4.9.1 Detailed Description

4.9.2 Function Documentation

4.9.2.1 SystemCoreClockUpdate()

```
void SystemCoreClockUpdate (
             void  )
```

Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Note

Each time the core clock (HCLK) changes, this function must be called to update SystemCoreClock variable value. Otherwise, any configuration based on this variable will be incorrect.

- The system frequency computed by this function is not the real frequency in the chip. It is calculated based on the predefined constant and the selected clock source:

- If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(∗)
- If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(∗∗)
- If SYSCLK source is PLL, SystemCoreClock will contain the HSE_VALUE(∗∗) or HSI_VALUE(∗) multiplied/divided by the PLL factors.

(∗) HSI_VALUE is a constant defined in stm32f0xx_hal_conf.h file (default value 8 MHz) but the real value may vary depending on the variations in voltage and temperature.

(∗∗) HSE_VALUE is a constant defined in stm32f0xx_hal_conf.h file (its value depends on the application requirements), user has to ensure that HSE_VALUE is same as the real frequency of the crystal used. Otherwise, this function may have wrong result.

- The result of this function could be not correct when using fractional value for HSE crystal.

Parameters

None	

Return values

None	

4.9.2.2 SystemInit()

```
void SystemInit (
             void  )
```

Setup the microcontroller system.

Parameters

None	

Return values

None	

Chapter 5

Class Documentation

5.1 Kalman_t Struct Reference

Kalman structure.

`#include <mpu6050.h>`

Public Attributes

- double **Q_angle**
- double **Q_bias**
- double **R_measure**
- double **angle**
- double **bias**
- double **P** [2][2]

5.1.1 Detailed Description

Kalman structure.

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

- Inc/mpu6050.h

5.2 MPU6050_t Struct Reference

MPU6050 structure.

`#include <mpu6050.h>`

Public Attributes

- int16_t **Accel_X_RAW**
- int16_t **Accel_Y_RAW**
- int16_t **Accel_Z_RAW**
- double **Ax**
- double **Ay**
- double **Az**
- int16_t **Gyro_X_RAW**
- int16_t **Gyro_Y_RAW**
- int16_t **Gyro_Z_RAW**
- double **Gx**
- double **Gy**
- double **Gz**
- float **Temperature**
- double **KalmanAngleX**
- double **KalmanAngleY**

5.2.1 Detailed Description

MPU6050 structure.

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

- Inc/mpu6050.h

5.3 typHoverHandler Struct Reference

struct for hovercraft

`#include <controller.h>`

Public Attributes

- uint8_t **status**
- typMotorHandler **motorA**
  motor left
- typMotorHandler **motorB**
  motor right
- typMotorHandler **motorC**
  motor down
- typMotorHandler **motorD**
  motor down

5.3.1 Detailed Description

struct for hovercraft

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

- Inc/controller.h

5.4 typMotorHandler Struct Reference

individual motor

`#include <controller.h>`

Public Attributes

- char **motorCode**
  ascii code for a motor
- uint8_t **speed**
  0-255

5.4.1 Detailed Description

individual motor

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

- Inc/controller.h

5.5 typPWMInputHandler Struct Reference

unprossessed pwms for motors

`#include <controller.h>`

Public Attributes

- uint8_t **pwmInputA**
- uint8_t **pwmInputB**
- uint8_t **pwmInputC**
- uint8_t **pwmInputD**

5.5.1 Detailed Description

unprossessed pwms for motors

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

- Inc/controller.h

5.6 typPWMOutputHandler Struct Reference

obselete

`#include <controller.h>`

Public Attributes

- uint8_t **pwmOutputA**
- uint8_t **pwmOutputB**
- uint8_t **pwmOutputC**
- uint8_t **pwmOutputD**

5.6.1 Detailed Description

obselete

OBSOLETE output pwm for motors

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

- Inc/controller.h

5.7 typVector Struct Reference

vector indentifier

`#include <controller.h>`

Public Attributes

- uint8_t **forward**
- uint8_t **left**
- uint8_t **right**
- uint8_t **backward**

5.7.1 Detailed Description

vector indentifier

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

- Inc/controller.h

Chapter 6

File Documentation

6.1 controller.h

```
1 /*
2  *  @file controller.h
3  *  @author: Anil Ozrenk
4  *  @date 05/13/2022
5  *
6  */
7 #ifndef _CONTROL_H_
8 #define _CONTROL_H_
9
10 #include <stdint.h>
11
12 #define OK 'F'
13
14 #define MAX_POWER 255
16 enum vector_state{
17     forward,
18     turn_left,
19     turn_right,
20     forward_left,
21     forward_right,
22     idle,
23     halt,
24
25 };
27 typedef struct{
28     char motorCode;
29     uint8_t speed;
30
31 }typMotorHandler;
33 typedef struct{
34     uint8_t status;
35     typMotorHandler motorA;
36     typMotorHandler motorB;
37     typMotorHandler motorC;
38     typMotorHandler motorD;
39 }typHoverHandler;
41 typedef struct{
42     uint8_t pwmInputA;
43     uint8_t pwmInputB;
44     uint8_t pwmInputC;
45     uint8_t pwmInputD;
46 }typPWMInputHandler;
52 typedef struct{   //0-255
53     uint8_t pwmOutputA;
54     uint8_t pwmOutputB;
55     uint8_t pwmOutputC;
56     uint8_t pwmOutputD;
57 }typPWMOutputHandler;
59 typedef struct{
60     uint8_t forward;
61     uint8_t left;
62     uint8_t right;
63     uint8_t backward;
64 }typVector;
65
66 void vectorToPwm(typVector *hVec, typPWMInputHandler *pwmInput);
67 void angleToVector(typVector *hVec,double curr_angle_x,double start_angle_x,double curr_angle_y,double
       start_angle_y ,double death_zone);
```

```
68 void deathzone(double *delta_x,double pmax,double nmax,double death_zone);
69 void hoverInit(typHoverHandler *hhov);
70 void pwmSmooting(typHoverHandler *hHov,typPWMInputHandler *input,double kf);
71 void deathzonefit(double *delta_x,double pmax,double nmax,double death_zone);
72 void command(typHoverHandler *hHov,char *buff);
73
74
75
76
77
78 #endif
79
80
81
82
83
```

6.2 Inc/main.h File Reference

: Header for main.c file. This file contains the common defines of the application.

```
#include "stm32f0xx_hal.h"
```

Macros

- #define **B1_Pin** GPIO_PIN_13
- #define **B1_GPIO_Port** GPIOC
- #define **B1_EXTI_IRQn** EXTI4_15_IRQn
- #define **USART_TX_Pin** GPIO_PIN_2
- #define **USART_TX_GPIO_Port** GPIOA
- #define **USART_RX_Pin** GPIO_PIN_3
- #define **USART_RX_GPIO_Port** GPIOA
- #define **LD2_Pin** GPIO_PIN_5
- #define **LD2_GPIO_Port** GPIOA
- #define **TMS_Pin** GPIO_PIN_13
- #define **TMS_GPIO_Port** GPIOA
- #define **TCK_Pin** GPIO_PIN_14
- #define **TCK_GPIO_Port** GPIOA

Functions

- void Error_Handler (void)

  This function is executed in case of error occurrence.

6.2.1 Detailed Description

: Header for main.c file. This file contains the common defines of the application.

Attention

6.2.2 Function Documentation

6.2.2.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

6.3 main.h

Go to the documentation of this file.

```
1 /* USER CODE BEGIN Header */
19 /* USER CODE END Header */
20
21 /* Define to prevent recursive inclusion -------------------------------------*/
22 #ifndef __MAIN_H
23 #define __MAIN_H
24
25 #ifdef __cplusplus
26 extern "C" {
27 #endif
28
29 /* Includes ------------------------------------------------------------------*/
30 #include "stm32f0xx_hal.h"
31
32 /* Private includes ----------------------------------------------------------*/
33 /* USER CODE BEGIN Includes */
34
35 /* USER CODE END Includes */
36
37 /* Exported types ------------------------------------------------------------*/
38 /* USER CODE BEGIN ET */
39
40 /* USER CODE END ET */
41
42 /* Exported constants --------------------------------------------------------*/
43 /* USER CODE BEGIN EC */
44
45 /* USER CODE END EC */
46
47 /* Exported macro ------------------------------------------------------------*/
48 /* USER CODE BEGIN EM */
49
50 /* USER CODE END EM */
51
52 /* Exported functions prototypes ---------------------------------------------*/
53 void Error_Handler(void);
54
55 /* USER CODE BEGIN EFP */
56
57 /* USER CODE END EFP */
58
59 /* Private defines -----------------------------------------------------------*/
60 #define B1_Pin GPIO_PIN_13
61 #define B1_GPIO_Port GPIOC
62 #define B1_EXTI_IRQn EXTI4_15_IRQn
63 #define USART_TX_Pin GPIO_PIN_2
64 #define USART_TX_GPIO_Port GPIOA
65 #define USART_RX_Pin GPIO_PIN_3
66 #define USART_RX_GPIO_Port GPIOA
67 #define LD2_Pin GPIO_PIN_5
68 #define LD2_GPIO_Port GPIOA
```

```
69 #define TMS_Pin GPIO_PIN_13
70 #define TMS_GPIO_Port GPIOA
71 #define TCK_Pin GPIO_PIN_14
72 #define TCK_GPIO_Port GPIOA
73 /* USER CODE BEGIN Private defines */
74
75 /* USER CODE END Private defines */
76
77 #ifdef __cplusplus
78 }
79 #endif
80
81 #endif /* __MAIN_H */
```

6.4 Inc/mpu6050.h File Reference

```
#include <stdint.h>
#include "main.h"
```

Classes

- struct MPU6050_t

  MPU6050 structure.
- struct Kalman_t

  Kalman structure.

Functions

- uint8_t MPU6050_Init (I2C_HandleTypeDef ∗I2Cx)
- void MPU6050_Read_Accel (I2C_HandleTypeDef ∗I2Cx, MPU6050_t ∗DataStruct)
- void MPU6050_Read_Gyro (I2C_HandleTypeDef ∗I2Cx, MPU6050_t ∗DataStruct)
- void MPU6050_Read_Temp (I2C_HandleTypeDef ∗I2Cx, MPU6050_t ∗DataStruct)
- void MPU6050_Read_All (I2C_HandleTypeDef ∗I2Cx, MPU6050_t ∗DataStruct)
- double **Kalman_getAngle** (Kalman_t ∗Kalman, double newAngle, double newRate, double dt)

6.4.1 Detailed Description

Date

11/13/2019

Author

Bulanov Konstantin

6.4.2 Function Documentation

6.4.2.1 MPU6050_Init()

```
uint8_t MPU6050_Init (
            I2C_HandleTypeDef * I2Cx )
```

check device ID WHO_AM_I

power management register 0X6B we should write all 0's to wake the sensor up

Set DATA RATE of 1KHz by writing SMPLRT_DIV register

6.4.2.2 MPU6050_Read_Accel()

```
void MPU6050_Read_Accel (
            I2C_HandleTypeDef * I2Cx,
            MPU6050_t * DataStruct )
```

Read 6 BYTES of data starting from ACCEL_XOUT_H register

convert the RAW values into acceleration in 'g' we have to divide according to the Full scale value set in FS_SEL have configured FS_SEL = 0. So I am dividing by 16384.0 for more details check ACCEL_CONFIG Register

6.4.2.3 MPU6050_Read_All()

```
void MPU6050_Read_All (
            I2C_HandleTypeDef * I2Cx,
            MPU6050_t * DataStruct )
```

Read 14 BYTES of data starting from ACCEL_XOUT_H register

Kalman angle solve

6.4.2.4 MPU6050_Read_Gyro()

```
void MPU6050_Read_Gyro (
            I2C_HandleTypeDef * I2Cx,
            MPU6050_t * DataStruct )
```

Read 6 BYTES of data starting from GYRO_XOUT_H register

convert the RAW values into dps (°/s) we have to divide according to the Full scale value set in FS_SEL I have configured FS_SEL = 0. So I am dividing by 131.0 for more details check GYRO_CONFIG Register

6.4.2.5 MPU6050_Read_Temp()

```
void MPU6050_Read_Temp (
            I2C_HandleTypeDef * I2Cx,
            MPU6050_t * DataStruct )
```

Read 2 BYTES of data starting from TEMP_OUT_H_REG register

6.5 mpu6050.h

Go to the documentation of this file.

```
1
7 #ifndef INC_GY521_H_
8 #define INC_GY521_H_
9
10 #endif /* INC_GY521_H_ */
11
12 #include <stdint.h>
13 #include "main.h"
14
16 typedef struct
17 {
18
19     int16_t Accel_X_RAW;
20     int16_t Accel_Y_RAW;
21     int16_t Accel_Z_RAW;
22     double Ax;
23     double Ay;
24     double Az;
25
26     int16_t Gyro_X_RAW;
27     int16_t Gyro_Y_RAW;
28     int16_t Gyro_Z_RAW;
29     double Gx;
30     double Gy;
31     double Gz;
32
33     float Temperature;
34
35     double KalmanAngleX;
36     double KalmanAngleY;
37
38
39 } MPU6050_t;
40
42 typedef struct
43 {
44     double Q_angle;
45     double Q_bias;
46     double R_measure;
47     double angle;
48     double bias;
49     double P[2][2];
50 } Kalman_t;
51
52 uint8_t MPU6050_Init(I2C_HandleTypeDef *I2Cx);
53
54 void MPU6050_Read_Accel(I2C_HandleTypeDef *I2Cx, MPU6050_t *DataStruct);
55
56 void MPU6050_Read_Gyro(I2C_HandleTypeDef *I2Cx, MPU6050_t *DataStruct);
57
58 void MPU6050_Read_Temp(I2C_HandleTypeDef *I2Cx, MPU6050_t *DataStruct);
59
60 void MPU6050_Read_All(I2C_HandleTypeDef *I2Cx, MPU6050_t *DataStruct);
61
62 double Kalman_getAngle(Kalman_t *Kalman, double newAngle, double newRate, double dt);
```

6.6 Inc/stm32f0xx_hal_conf.h File Reference

HAL configuration file.

```
#include "stm32f0xx_hal_rcc.h"
#include "stm32f0xx_hal_gpio.h"
#include "stm32f0xx_hal_exti.h"
#include "stm32f0xx_hal_dma.h"
#include "stm32f0xx_hal_cortex.h"
#include "stm32f0xx_hal_flash.h"
#include "stm32f0xx_hal_i2c.h"
#include "stm32f0xx_hal_pwr.h"
#include "stm32f0xx_hal_uart.h"
```

Macros

- #define **HAL_MODULE_ENABLED**

  This is the list of modules to be used in the HAL driver.
- #define **HAL_UART_MODULE_ENABLED**
- #define **HAL_CORTEX_MODULE_ENABLED**
- #define **HAL_DMA_MODULE_ENABLED**
- #define **HAL_FLASH_MODULE_ENABLED**
- #define **HAL_GPIO_MODULE_ENABLED**
- #define **HAL_EXTI_MODULE_ENABLED**
- #define **HAL_PWR_MODULE_ENABLED**
- #define **HAL_RCC_MODULE_ENABLED**
- #define **HAL_I2C_MODULE_ENABLED**
- #define HSE_VALUE ((uint32_t)8000000)

  Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).
- #define HSE_STARTUP_TIMEOUT ((uint32_t)100)

  In the following line adjust the External High Speed oscillator (HSE) Startup Timeout value.
- #define HSI_VALUE ((uint32_t)8000000)

  Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).
- #define HSI_STARTUP_TIMEOUT ((uint32_t)5000)

  In the following line adjust the Internal High Speed oscillator (HSI) Startup Timeout value.
- #define HSI14_VALUE ((uint32_t)14000000)

  Internal High Speed oscillator for ADC (HSI14) value.
- #define HSI48_VALUE ((uint32_t)48000000)

  Internal High Speed oscillator for USB (HSI48) value.
- #define **LSI_VALUE** ((uint32_t)40000)

  Internal Low Speed oscillator (LSI) value.
- #define LSE_VALUE ((uint32_t)32768)

  External Low Speed oscillator (LSI) value.
- #define LSE_STARTUP_TIMEOUT ((uint32_t)5000)

  Time out for LSE start up value in ms.
- #define VDD_VALUE ((uint32_t)3300)

  This is the HAL system configuration section.
- #define TICK_INT_PRIORITY ((uint32_t)0)
- #define **USE_RTOS** 0
- #define **PREFETCH_ENABLE** 1
- #define **INSTRUCTION_CACHE_ENABLE** 0
- #define **DATA_CACHE_ENABLE** 0
- #define **USE_SPI_CRC** 0U
- #define **USE_HAL_ADC_REGISTER_CALLBACKS** 0U /∗ ADC register callback disabled ∗/
- #define **USE_HAL_CAN_REGISTER_CALLBACKS** 0U /∗ CAN register callback disabled ∗/
- #define **USE_HAL_COMP_REGISTER_CALLBACKS** 0U /∗ COMP register callback disabled ∗/
- #define **USE_HAL_CEC_REGISTER_CALLBACKS** 0U /∗ CEC register callback disabled ∗/
- #define **USE_HAL_DAC_REGISTER_CALLBACKS** 0U /∗ DAC register callback disabled ∗/
- #define **USE_HAL_I2C_REGISTER_CALLBACKS** 0U /∗ I2C register callback disabled ∗/
- #define **USE_HAL_SMBUS_REGISTER_CALLBACKS** 0U /∗ SMBUS register callback disabled ∗/
- #define **USE_HAL_UART_REGISTER_CALLBACKS** 0U /∗ UART register callback disabled ∗/
- #define **USE_HAL_USART_REGISTER_CALLBACKS** 0U /∗ USART register callback disabled ∗/
- #define **USE_HAL_IRDA_REGISTER_CALLBACKS** 0U /∗ IRDA register callback disabled ∗/
- #define **USE_HAL_SMARTCARD_REGISTER_CALLBACKS** 0U /∗ SMARTCARD register callback disabled ∗/

- #define **USE_HAL_WWDG_REGISTER_CALLBACKS** 0U /* WWDG register callback disabled */
- #define **USE_HAL_RTC_REGISTER_CALLBACKS** 0U /* RTC register callback disabled */
- #define **USE_HAL_SPI_REGISTER_CALLBACKS** 0U /* SPI register callback disabled */
- #define **USE_HAL_I2S_REGISTER_CALLBACKS** 0U /* I2S register callback disabled */
- #define **USE_HAL_TIM_REGISTER_CALLBACKS** 0U /* TIM register callback disabled */
- #define **USE_HAL_TSC_REGISTER_CALLBACKS** 0U /* TSC register callback disabled */
- #define **USE_HAL_PCD_REGISTER_CALLBACKS** 0U /* PCD register callback disabled */
- #define assert_param(expr) ((void)0U)

  Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.

6.6.1 Detailed Description

HAL configuration file.

Attention

6.6.2 Macro Definition Documentation

6.6.2.1 assert_param

```
#define assert_param(
              expr ) ((void)0U)
```

Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.

Include module's header file

6.6.2.2 HSE_STARTUP_TIMEOUT

```
#define HSE_STARTUP_TIMEOUT ((uint32_t)100)
```

In the following line adjust the External High Speed oscillator (HSE) Startup Timeout value.

Time out for HSE start up, in ms

6.6.2.3 HSE_VALUE

```
#define HSE_VALUE ((uint32_t)8000000)
```

Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).

Value of the External oscillator in Hz

6.6.2.4 HSI14_VALUE

```
#define HSI14_VALUE ((uint32_t)14000000)
```

Internal High Speed oscillator for ADC (HSI14) value.

Value of the Internal High Speed oscillator for ADC in Hz. The real value may vary depending on the variations in voltage and temperature.

6.6.2.5 HSI48_VALUE

```
#define HSI48_VALUE ((uint32_t)48000000)
```

Internal High Speed oscillator for USB (HSI48) value.

Value of the Internal High Speed oscillator for USB in Hz. The real value may vary depending on the variations in voltage and temperature.

6.6.2.6 HSI_STARTUP_TIMEOUT

```
#define HSI_STARTUP_TIMEOUT ((uint32_t)5000)
```

In the following line adjust the Internal High Speed oscillator (HSI) Startup Timeout value.

Time out for HSI start up

6.6.2.7 HSI_VALUE

```
#define HSI_VALUE ((uint32_t)8000000)
```

Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).

Value of the Internal oscillator in Hz

6.6.2.8 LSE_STARTUP_TIMEOUT

```
#define LSE_STARTUP_TIMEOUT ((uint32_t)5000)
```

Time out for LSE start up value in ms.

Time out for LSE start up, in ms

6.6.2.9 LSE_VALUE

```
#define LSE_VALUE ((uint32_t)32768)
```

External Low Speed oscillator (LSI) value.

< Value of the Internal Low Speed oscillator in Hz The real value may vary depending on the variations in voltage and temperature.
Value of the External Low Speed oscillator in Hz

6.6.2.10 TICK_INT_PRIORITY

```
#define TICK_INT_PRIORITY ((uint32_t)0)
```

tick interrupt priority (lowest by default)

6.6.2.11 VDD_VALUE

```
#define VDD_VALUE ((uint32_t)3300)
```

This is the HAL system configuration section.

Value of VDD in mv

6.7 stm32f0xx_hal_conf.h

Go to the documentation of this file.

```
1 /* USER CODE BEGIN Header */
18 /* USER CODE END Header */
19
20 /* Define to prevent recursive inclusion -------------------------------------*/
21 #ifndef __STM32F0xx_HAL_CONF_H
22 #define __STM32F0xx_HAL_CONF_H
23
24 #ifdef __cplusplus
25  extern "C" {
26 #endif
27
28 /* Exported types ------------------------------------------------------------*/
29 /* Exported constants --------------------------------------------------------*/
30
31 /* ########################## Module Selection ############################## */
35 #define HAL_MODULE_ENABLED
36   /*#define HAL_ADC_MODULE_ENABLED   */
37 /*#define HAL_CRYP_MODULE_ENABLED   */
38 /*#define HAL_CAN_MODULE_ENABLED   */
39 /*#define HAL_CEC_MODULE_ENABLED   */
40 /*#define HAL_COMP_MODULE_ENABLED   */
```

```
41 /*#define HAL_CRC_MODULE_ENABLED   */
42 /*#define HAL_CRYP_MODULE_ENABLED   */
43 /*#define HAL_TSC_MODULE_ENABLED   */
44 /*#define HAL_DAC_MODULE_ENABLED   */
45 /*#define HAL_I2S_MODULE_ENABLED   */
46 /*#define HAL_IWDG_MODULE_ENABLED   */
47 /*#define HAL_LCD_MODULE_ENABLED   */
48 /*#define HAL_LPTIM_MODULE_ENABLED   */
49 /*#define HAL_RNG_MODULE_ENABLED   */
50 /*#define HAL_RTC_MODULE_ENABLED   */
51 /*#define HAL_SPI_MODULE_ENABLED   */
52 /*#define HAL_TIM_MODULE_ENABLED   */
53 #define HAL_UART_MODULE_ENABLED
54 /*#define HAL_USART_MODULE_ENABLED   */
55 /*#define HAL_IRDA_MODULE_ENABLED   */
56 /*#define HAL_SMARTCARD_MODULE_ENABLED   */
57 /*#define HAL_SMBUS_MODULE_ENABLED   */
58 /*#define HAL_WWDG_MODULE_ENABLED   */
59 /*#define HAL_PCD_MODULE_ENABLED   */
60 #define HAL_CORTEX_MODULE_ENABLED
61 #define HAL_DMA_MODULE_ENABLED
62 #define HAL_FLASH_MODULE_ENABLED
63 #define HAL_GPIO_MODULE_ENABLED
64 #define HAL_EXTI_MODULE_ENABLED
65 #define HAL_PWR_MODULE_ENABLED
66 #define HAL_RCC_MODULE_ENABLED
67 #define HAL_I2C_MODULE_ENABLED
68
69 /* ########################## HSE/HSI Values adaptation ##################### */
75 #if !defined  (HSE_VALUE)
76   #define HSE_VALUE    ((uint32_t)8000000)
77 #endif /* HSE_VALUE */
78
83 #if !defined  (HSE_STARTUP_TIMEOUT)
84   #define HSE_STARTUP_TIMEOUT    ((uint32_t)100)
85 #endif /* HSE_STARTUP_TIMEOUT */
86
92 #if !defined  (HSI_VALUE)
93   #define HSI_VALUE    ((uint32_t)8000000)
94 #endif /* HSI_VALUE */
95
100 #if !defined  (HSI_STARTUP_TIMEOUT)
101  #define HSI_STARTUP_TIMEOUT   ((uint32_t)5000)
102 #endif /* HSI_STARTUP_TIMEOUT */
103
107 #if !defined  (HSI14_VALUE)
108 #define HSI14_VALUE ((uint32_t)14000000)
111 #endif /* HSI14_VALUE */
112
116 #if !defined  (HSI48_VALUE)
117   #define HSI48_VALUE ((uint32_t)48000000)
120 #endif /* HSI48_VALUE */
121
125 #if !defined  (LSI_VALUE)
126   #define LSI_VALUE  ((uint32_t)40000)
127 #endif /* LSI_VALUE */
133 #if !defined  (LSE_VALUE)
134   #define LSE_VALUE  ((uint32_t)32768)
135 #endif /* LSE_VALUE */
136
140 #if !defined  (LSE_STARTUP_TIMEOUT)
141   #define LSE_STARTUP_TIMEOUT    ((uint32_t)5000)
142 #endif /* LSE_STARTUP_TIMEOUT */
143
144 /* Tip: To avoid modifying this file each time you need to use different HSE,
145    ===  you can define the HSE value in your toolchain compiler preprocessor. */
146
147 /* ########################### System Configuration ######################### */
151 #define  VDD_VALUE                    ((uint32_t)3300)
152 #define  TICK_INT_PRIORITY            ((uint32_t)0)
153                                                                              /*  Warning: Must be set
    to higher priority for HAL_Delay()  */
154                                                                              /*  and HAL_GetTick()
    usage under interrupt context          */
155 #define  USE_RTOS                     0
156 #define  PREFETCH_ENABLE              1
157 #define  INSTRUCTION_CACHE_ENABLE     0
158 #define  DATA_CACHE_ENABLE            0
159 #define  USE_SPI_CRC                     0U
160
161 #define  USE_HAL_ADC_REGISTER_CALLBACKS         0U /* ADC register callback disabled       */
162 #define  USE_HAL_CAN_REGISTER_CALLBACKS         0U /* CAN register callback disabled       */
163 #define  USE_HAL_COMP_REGISTER_CALLBACKS        0U /* COMP register callback disabled      */
164 #define  USE_HAL_CEC_REGISTER_CALLBACKS         0U /* CEC register callback disabled       */
165 #define  USE_HAL_DAC_REGISTER_CALLBACKS         0U /* DAC register callback disabled       */
166 #define  USE_HAL_I2C_REGISTER_CALLBACKS         0U /* I2C register callback disabled       */
167 #define  USE_HAL_SMBUS_REGISTER_CALLBACKS       0U /* SMBUS register callback disabled     */
```

```
168 #define  USE_HAL_UART_REGISTER_CALLBACKS        0U /* UART register callback disabled      */
169 #define  USE_HAL_USART_REGISTER_CALLBACKS       0U /* USART register callback disabled     */
170 #define  USE_HAL_IRDA_REGISTER_CALLBACKS        0U /* IRDA register callback disabled      */
171 #define  USE_HAL_SMARTCARD_REGISTER_CALLBACKS   0U /* SMARTCARD register callback disabled */
172 #define  USE_HAL_WWDG_REGISTER_CALLBACKS        0U /* WWDG register callback disabled      */
173 #define  USE_HAL_RTC_REGISTER_CALLBACKS         0U /* RTC register callback disabled       */
174 #define  USE_HAL_SPI_REGISTER_CALLBACKS         0U /* SPI register callback disabled       */
175 #define  USE_HAL_I2S_REGISTER_CALLBACKS         0U /* I2S register callback disabled       */
176 #define  USE_HAL_TIM_REGISTER_CALLBACKS         0U /* TIM register callback disabled       */
177 #define  USE_HAL_TSC_REGISTER_CALLBACKS         0U /* TSC register callback disabled       */
178 #define  USE_HAL_PCD_REGISTER_CALLBACKS         0U /* PCD register callback disabled       */
179
180 /* ########################## Assert Selection ############################## */
185 /* #define USE_FULL_ASSERT   1U */
186
187 /* Includes ------------------------------------------------------------------*/
192 #ifdef HAL_RCC_MODULE_ENABLED
193  #include "stm32f0xx_hal_rcc.h"
194 #endif /* HAL_RCC_MODULE_ENABLED */
195
196 #ifdef HAL_GPIO_MODULE_ENABLED
197  #include "stm32f0xx_hal_gpio.h"
198 #endif /* HAL_GPIO_MODULE_ENABLED */
199
200 #ifdef HAL_EXTI_MODULE_ENABLED
201   #include "stm32f0xx_hal_exti.h"
202 #endif /* HAL_EXTI_MODULE_ENABLED */
203
204 #ifdef HAL_DMA_MODULE_ENABLED
205   #include "stm32f0xx_hal_dma.h"
206 #endif /* HAL_DMA_MODULE_ENABLED */
207
208 #ifdef HAL_CORTEX_MODULE_ENABLED
209  #include "stm32f0xx_hal_cortex.h"
210 #endif /* HAL_CORTEX_MODULE_ENABLED */
211
212 #ifdef HAL_ADC_MODULE_ENABLED
213  #include "stm32f0xx_hal_adc.h"
214 #endif /* HAL_ADC_MODULE_ENABLED */
215
216 #ifdef HAL_CAN_MODULE_ENABLED
217  #include "stm32f0xx_hal_can.h"
218 #endif /* HAL_CAN_MODULE_ENABLED */
219
220 #ifdef HAL_CEC_MODULE_ENABLED
221  #include "stm32f0xx_hal_cec.h"
222 #endif /* HAL_CEC_MODULE_ENABLED */
223
224 #ifdef HAL_COMP_MODULE_ENABLED
225  #include "stm32f0xx_hal_comp.h"
226 #endif /* HAL_COMP_MODULE_ENABLED */
227
228 #ifdef HAL_CRC_MODULE_ENABLED
229  #include "stm32f0xx_hal_crc.h"
230 #endif /* HAL_CRC_MODULE_ENABLED */
231
232 #ifdef HAL_DAC_MODULE_ENABLED
233  #include "stm32f0xx_hal_dac.h"
234 #endif /* HAL_DAC_MODULE_ENABLED */
235
236 #ifdef HAL_FLASH_MODULE_ENABLED
237  #include "stm32f0xx_hal_flash.h"
238 #endif /* HAL_FLASH_MODULE_ENABLED */
239
240 #ifdef HAL_I2C_MODULE_ENABLED
241  #include "stm32f0xx_hal_i2c.h"
242 #endif /* HAL_I2C_MODULE_ENABLED */
243
244 #ifdef HAL_I2S_MODULE_ENABLED
245  #include "stm32f0xx_hal_i2s.h"
246 #endif /* HAL_I2S_MODULE_ENABLED */
247
248 #ifdef HAL_IRDA_MODULE_ENABLED
249  #include "stm32f0xx_hal_irda.h"
250 #endif /* HAL_IRDA_MODULE_ENABLED */
251
252 #ifdef HAL_IWDG_MODULE_ENABLED
253  #include "stm32f0xx_hal_iwdg.h"
254 #endif /* HAL_IWDG_MODULE_ENABLED */
255
256 #ifdef HAL_PCD_MODULE_ENABLED
257  #include "stm32f0xx_hal_pcd.h"
258 #endif /* HAL_PCD_MODULE_ENABLED */
259
260 #ifdef HAL_PWR_MODULE_ENABLED
261  #include "stm32f0xx_hal_pwr.h"
262 #endif /* HAL_PWR_MODULE_ENABLED */
```

```
263
264 #ifdef HAL_RTC_MODULE_ENABLED
265  #include "stm32f0xx_hal_rtc.h"
266 #endif /* HAL_RTC_MODULE_ENABLED */
267
268 #ifdef HAL_SMARTCARD_MODULE_ENABLED
269  #include "stm32f0xx_hal_smartcard.h"
270 #endif /* HAL_SMARTCARD_MODULE_ENABLED */
271
272 #ifdef HAL_SMBUS_MODULE_ENABLED
273  #include "stm32f0xx_hal_smbus.h"
274 #endif /* HAL_SMBUS_MODULE_ENABLED */
275
276 #ifdef HAL_SPI_MODULE_ENABLED
277  #include "stm32f0xx_hal_spi.h"
278 #endif /* HAL_SPI_MODULE_ENABLED */
279
280 #ifdef HAL_TIM_MODULE_ENABLED
281  #include "stm32f0xx_hal_tim.h"
282 #endif /* HAL_TIM_MODULE_ENABLED */
283
284 #ifdef HAL_TSC_MODULE_ENABLED
285  #include "stm32f0xx_hal_tsc.h"
286 #endif /* HAL_TSC_MODULE_ENABLED */
287
288 #ifdef HAL_UART_MODULE_ENABLED
289  #include "stm32f0xx_hal_uart.h"
290 #endif /* HAL_UART_MODULE_ENABLED */
291
292 #ifdef HAL_USART_MODULE_ENABLED
293  #include "stm32f0xx_hal_usart.h"
294 #endif /* HAL_USART_MODULE_ENABLED */
295
296 #ifdef HAL_WWDG_MODULE_ENABLED
297  #include "stm32f0xx_hal_wwdg.h"
298 #endif /* HAL_WWDG_MODULE_ENABLED */
299
300 /* Exported macro ------------------------------------------------------------*/
301 #ifdef  USE_FULL_ASSERT
310   #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
311 /* Exported functions ------------------------------------------------------- */
312   void assert_failed(uint8_t* file, uint32_t line);
313 #else
314   #define assert_param(expr) ((void)0U)
315 #endif /* USE_FULL_ASSERT */
316
317 #ifdef __cplusplus
318 }
319 #endif
320
321 #endif /* __STM32F0xx_HAL_CONF_H */
322
```

6.8 Inc/stm32f0xx_it.h File Reference

This file contains the headers of the interrupt handlers.

Functions

- void **NMI_Handler** (void)

  This function handles Non maskable interrupt.
- void **HardFault_Handler** (void)

  This function handles Hard fault interrupt.
- void **SVC_Handler** (void)

  This function handles System service call via SWI instruction.
- void **PendSV_Handler** (void)

  This function handles Pendable request for system service.
- void **SysTick_Handler** (void)

  This function handles System tick timer.
- void **EXTI4_15_IRQHandler** (void)

This function handles EXTI line 4 to 15 interrupts.

- void **USART1_IRQHandler** (void)

  This function handles USART1 global interrupt / USART1 wake-up interrupt through EXTI line 25.

- void **USART2_IRQHandler** (void)

  This function handles USART2 global interrupt / USART2 wake-up interrupt through EXTI line 26.

6.8.1 Detailed Description

This file contains the headers of the interrupt handlers.

Attention

6.9 stm32f0xx_it.h

Go to the documentation of this file.

```
1 /* USER CODE BEGIN Header */
18 /* USER CODE END Header */
19
20 /* Define to prevent recursive inclusion -------------------------------------*/
21 #ifndef __STM32F0xx_IT_H
22 #define __STM32F0xx_IT_H
23
24 #ifdef __cplusplus
25  extern "C" {
26 #endif
27
28 /* Private includes ----------------------------------------------------------*/
29 /* USER CODE BEGIN Includes */
30
31 /* USER CODE END Includes */
32
33 /* Exported types ------------------------------------------------------------*/
34 /* USER CODE BEGIN ET */
35
36 /* USER CODE END ET */
37
38 /* Exported constants --------------------------------------------------------*/
39 /* USER CODE BEGIN EC */
40
41 /* USER CODE END EC */
42
43 /* Exported macro ------------------------------------------------------------*/
44 /* USER CODE BEGIN EM */
45
46 /* USER CODE END EM */
47
48 /* Exported functions prototypes ---------------------------------------------*/
49 void NMI_Handler(void);
50 void HardFault_Handler(void);
51 void SVC_Handler(void);
52 void PendSV_Handler(void);
53 void SysTick_Handler(void);
54 void EXTI4_15_IRQHandler(void);
55 void USART1_IRQHandler(void);
56 void USART2_IRQHandler(void);
57 /* USER CODE BEGIN EFP */
58
59 /* USER CODE END EFP */
60
61 #ifdef __cplusplus
62 }
63 #endif
64
65 #endif /* __STM32F0xx_IT_H */
```

6.10 Src/controller.c File Reference

all process angle data to sendable buffer

```
#include "controller.h"
```

Functions

- void pwmToAscii (typMotorHandler ∗hmotor, char ∗buff)
  8 bit unsigned integer value to string
- void deathzonefit (double ∗delta_x, double pmax, double nmax, double death_zone)
  limit the angle and add deathzone
- uint8_t vectorState (typVector ∗vector)
  decide the state of vectors
- void hoverInit (typHoverHandler ∗hhov)
  intiate hovercraft give motors ascii characters set all motors speeds to 0
- void pwmSmooting (typHoverHandler ∗hHov, typPWMInputHandler ∗input, double kf)
  exponantiel filter for smoothing motor pwm
- void vectorToPwm (typVector ∗hVec, typPWMInputHandler ∗pwmInput)
  generate pwm value corrosponding to vector states
- void angleToVector (typVector ∗hVec, double curr_angle_x, double start_angle_x, double curr_angle_y, double start_angle_y, double death_zone)
  angle values to vector calues
- void **command** (typHoverHandler ∗hHov, char ∗buff)

6.10.1 Detailed Description

all process angle data to sendable buffer

Date

05/13/2022

Author

Anil

6.10.2 Function Documentation

6.10.2.1 angleToVector()

```
void angleToVector (
            typVector * hVec,
            double curr_angle_x,
            double start_angle_x,
            double curr_angle_y,
            double start_angle_y,
            double death_zone )
```

angle values to vector calues

Parameters

typVector	
angle_x	
base_angle_x	
angle_y	
base_angle_y	
deathzone	

6.10.2.2 deathzonefit()

```
void deathzonefit (
            double * delta_x,
            double pmax,
            double nmax,
            double death_zone )
```

limit the angle and add deathzone

Parameters

delta_angle	
positive	max
negative	max
deathzone	

6.10.2.3 hoverInit()

```
void hoverInit (
            typHoverHandler * hhov )
```

intiate hovercraft give motors ascii characters set all motors speeds to 0

Parameters

typHoverHandler	

6.10.2.4 pwmSmooting()

```
void pwmSmooting (
            typHoverHandler * hHov,
```

```
            typPWMInputHandler * input,
            double kf )
```

exponantiel filter for smoothing motor pwm

Parameters

typHoverHandler	
typPWMInputHandler	

6.10.2.5 pwmToAscii()

```
void pwmToAscii (
            typMotorHandler * hmotor,
            char * buff )
```

8 bit unsigned integer value to string

Parameters

typMotorHandler	
char	∗buffer

6.10.2.6 vectorState()

```
uint8_t vectorState (
            typVector * vector )
```

decide the state of vectors

Parameters

typVector	

6.10.2.7 vectorToPwm()

```
void vectorToPwm (
            typVector * hVec,
            typPWMInputHandler * pwmInput )
```

generate pwm value corrosponding to vector states

Parameters

typPWMInputHandler	
typVector	

6.11 Src/main.c File Reference

Main program body.

```
#include "main.h"
#include "mpu6050.h"
#include "controller.h"
```

Macros

- #define **BUFFER_LEN** 13

  predefined lenght to main uart buffer
- #define **KF** 0.85

  exponantial filter coeficient

Functions

- void SystemClock_Config (void)

  System Clock Configuration.
- void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)
- int main (void)

  The application entry point.
- void Error_Handler (void)

  This function is executed in case of error occurrence.

Variables

- I2C_HandleTypeDef **hi2c1**
- UART_HandleTypeDef **huart1**
- UART_HandleTypeDef **huart2**
- MPU6050_t **MPU6050**
- typHoverHandler **hHover**
- typPWMInputHandler **hInput**
- typPWMOutputHandler **hOutput**
- typVector **hVector**
- double **x_angle_base**

  holds gyro angle
- double **y_angle_base**

  holds gyro angle
- char **rx_buffer** [BUFFER_LEN] = {0}

  main rx buffer
- char **tx_buffer** [BUFFER_LEN] = {0}

  main tx buffer

6.11.1 Detailed Description

Main program body.

Author

Anil Ozrenk

Attention

6.11.2 Function Documentation

6.11.2.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

6.11.2.2 HAL_GPIO_EXTI_Callback()

```
void HAL_GPIO_EXTI_Callback (
             uint16_t GPIO_Pin )
```

external interrupt routine for base angle reset gpio pin 13

6.11.2.3 main()

```
int main (
             void  )
```

The application entry point.

Return values

int	

initialize mpu6050 module

initialize hover struct

initial x angle base

initial y angle base

transmit buffer to bluetooth

transmit buffer via usb

6.11.2.4 SystemClock_Config()

```
void SystemClock_Config (
             void  )
```

System Clock Configuration.

Return values

None	

Initializes the RCC Oscillators according to the specified parameters in the RCC_OscInitTypeDef structure.

Initializes the CPU, AHB and APB buses clocks

6.12 Src/mpu6050.c File Reference

```
#include <math.h>
#include "mpu6050.h"
```

Macros

- #define RAD_TO_DEG 57.295779513082320876798154814105
- #define **WHO_AM_I_REG** 0x75
- #define **PWR_MGMT_1_REG** 0x6B
- #define **SMPLRT_DIV_REG** 0x19
- #define **ACCEL_CONFIG_REG** 0x1C
- #define **ACCEL_XOUT_H_REG** 0x3B
- #define **TEMP_OUT_H_REG** 0x41
- #define **GYRO_CONFIG_REG** 0x1B
- #define **GYRO_XOUT_H_REG** 0x43
- #define **MPU6050_ADDR** (0x68<<1)

  Setup MPU6050.

Functions

- uint8_t MPU6050_Init (I2C_HandleTypeDef ∗I2Cx)
- void MPU6050_Read_Accel (I2C_HandleTypeDef ∗I2Cx, MPU6050_t ∗DataStruct)
- void MPU6050_Read_Gyro (I2C_HandleTypeDef ∗I2Cx, MPU6050_t ∗DataStruct)
- void MPU6050_Read_Temp (I2C_HandleTypeDef ∗I2Cx, MPU6050_t ∗DataStruct)
- void MPU6050_Read_All (I2C_HandleTypeDef ∗I2Cx, MPU6050_t ∗DataStruct)
- double **Kalman_getAngle** (Kalman_t ∗Kalman, double newAngle, double newRate, double dt)

Variables

- const uint16_t **i2c_timeout** = 500
- const double **Accel_Z_corrector** = 14418.0
- uint32_t **timer**
- Kalman_t KalmanX
- Kalman_t KalmanY

6.12.1 Detailed Description

Date

11/13/2019

Author

Bulanov Konstantin

6.12.1.1 Contact information

e-mail : leech001@gmail.com

6.12.2 Macro Definition Documentation

6.12.2.1 RAD_TO_DEG

```
#define RAD_TO_DEG 57.295779513082320876798154814105
```

|---------- | Copyright (C) Bulanov Konstantin,2021 | | This program is free software: you can redistribute it and/or modify | it under the terms of the GNU General Public License as published by | the Free Software Foundation, either version 3 of the License, or | any later version. | | This program is distributed in the hope that it will be useful, | but WITHOUT ANY WARRANTY; without even the implied warranty of | MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the | GNU General Public License for more details. | | You should have received a copy of the GNU General Public License | along with this program. If not, see http://www.gnu.org/licenses/. | | Kalman filter algorithm used from https://github.com/TKJElectronics/KalmanFilter |----------

6.12.3 Function Documentation

6.12.3.1 MPU6050_Init()

```
uint8_t MPU6050_Init (
            I2C_HandleTypeDef * I2Cx )
```

check device ID WHO_AM_I

power management register 0X6B we should write all 0's to wake the sensor up

Set DATA RATE of 1KHz by writing SMPLRT_DIV register

6.12.3.2 MPU6050_Read_Accel()

```
void MPU6050_Read_Accel (
            I2C_HandleTypeDef * I2Cx,
            MPU6050_t * DataStruct )
```

Read 6 BYTES of data starting from ACCEL_XOUT_H register

convert the RAW values into acceleration in 'g' we have to divide according to the Full scale value set in FS_SEL have configured FS_SEL = 0. So I am dividing by 16384.0 for more details check ACCEL_CONFIG Register

6.12.3.3 MPU6050_Read_All()

```
void MPU6050_Read_All (
            I2C_HandleTypeDef * I2Cx,
            MPU6050_t * DataStruct )
```

Read 14 BYTES of data starting from ACCEL_XOUT_H register

Kalman angle solve

6.12.3.4 MPU6050_Read_Gyro()

```
void MPU6050_Read_Gyro (
            I2C_HandleTypeDef * I2Cx,
            MPU6050_t * DataStruct )
```

Read 6 BYTES of data starting from GYRO_XOUT_H register

convert the RAW values into dps (°/s) we have to divide according to the Full scale value set in FS_SEL I have configured FS_SEL = 0. So I am dividing by 131.0 for more details check GYRO_CONFIG Register

6.12.3.5 MPU6050_Read_Temp()

```
void MPU6050_Read_Temp (
            I2C_HandleTypeDef * I2Cx,
            MPU6050_t * DataStruct )
```

Read 2 BYTES of data starting from TEMP_OUT_H_REG register

6.12.4 Variable Documentation

6.12.4.1 KalmanX

```
Kalman_t KalmanX
```

Initial value:

```
= {
    .Q_angle = 0.001f,
    .Q_bias = 0.003f,
    .R_measure = 0.03f}
```

6.12.4.2 KalmanY

```
Kalman_t KalmanY
```

Initial value:

```
= {
    .Q_angle = 0.001f,
    .Q_bias = 0.003f,
    .R_measure = 0.03f,
}
```

6.13 Src/stm32f0xx_hal_msp.c File Reference

This file provides code for the MSP Initialization and de-Initialization codes.

```
#include "main.h"
```

Functions

- void HAL_MspInit (void)
- void HAL_I2C_MspInit (I2C_HandleTypeDef *hi2c)

  I2C MSP Initialization This function configures the hardware resources used in this example.
- void HAL_I2C_MspDeInit (I2C_HandleTypeDef *hi2c)

  I2C MSP De-Initialization This function freeze the hardware resources used in this example.
- void HAL_UART_MspInit (UART_HandleTypeDef *huart)

  UART MSP Initialization This function configures the hardware resources used in this example.
- void HAL_UART_MspDeInit (UART_HandleTypeDef *huart)

  UART MSP De-Initialization This function freeze the hardware resources used in this example.

6.13.1 Detailed Description

This file provides code for the MSP Initialization and de-Initialization codes.

Attention

6.13.2 Function Documentation

6.13.2.1 HAL_I2C_MspDeInit()

```
void HAL_I2C_MspDeInit (
            I2C_HandleTypeDef * hi2c )
```

I2C MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

hi2c	I2C handle pointer

Return values

None	

I2C1 GPIO Configuration PB8 ----—> I2C1_SCL PB9 ----—> I2C1_SDA

6.13.2.2 HAL_I2C_MspInit()

```
void HAL_I2C_MspInit (
            I2C_HandleTypeDef * hi2c )
```

I2C MSP Initialization This function configures the hardware resources used in this example.

Parameters

hi2c	I2C handle pointer

Return values

None	

I2C1 GPIO Configuration PB8 ----—> I2C1_SCL PB9 ----—> I2C1_SDA

6.13.2.3 HAL_MspInit()

```
void HAL_MspInit (
             void  )
```

Initializes the Global MSP.

6.13.2.4 HAL_UART_MspDeInit()

```
void HAL_UART_MspDeInit (
             UART_HandleTypeDef * huart )
```

UART MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

huart	UART handle pointer

Return values

None	

USART1 GPIO Configuration PA9 ----—> USART1_TX PA10 ----—> USART1_RX

USART2 GPIO Configuration PA2 ----—> USART2_TX PA3 ----—> USART2_RX

6.13.2.5 HAL_UART_MspInit()

```
void HAL_UART_MspInit (
             UART_HandleTypeDef * huart )
```

UART MSP Initialization This function configures the hardware resources used in this example.

Parameters

huart	UART handle pointer

Return values

None	

USART1 GPIO Configuration PA9 ----—> USART1_TX PA10 ----—> USART1_RX

USART2 GPIO Configuration PA2 ----—> USART2_TX PA3 ----—> USART2_RX

6.14 Src/stm32f0xx_it.c File Reference

Interrupt Service Routines.

```
#include "main.h"
#include "stm32f0xx_it.h"
```

Functions

- void **NMI_Handler** (void)

  This function handles Non maskable interrupt.
- void **HardFault_Handler** (void)

  This function handles Hard fault interrupt.
- void **SVC_Handler** (void)

  This function handles System service call via SWI instruction.
- void **PendSV_Handler** (void)

  This function handles Pendable request for system service.
- void **SysTick_Handler** (void)

  This function handles System tick timer.
- void **EXTI4_15_IRQHandler** (void)

  This function handles EXTI line 4 to 15 interrupts.
- void **USART1_IRQHandler** (void)

  This function handles USART1 global interrupt / USART1 wake-up interrupt through EXTI line 25.
- void **USART2_IRQHandler** (void)

  This function handles USART2 global interrupt / USART2 wake-up interrupt through EXTI line 26.

Variables

- UART_HandleTypeDef **huart1**
- UART_HandleTypeDef **huart2**

6.14.1 Detailed Description

Interrupt Service Routines.

Attention

6.15 Src/syscalls.c File Reference

STM32CubeIDE Minimal System calls file.

```
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
```

Functions

- int **__io_putchar** (int ch) __attribute__((weak))
- int **__io_getchar** (void)
- void **initialise_monitor_handles** ()
- int **_getpid** (void)
- int **_kill** (int pid, int sig)
- void **_exit** (int status)
- **__attribute__** ((weak))
- int **_close** (int file)
- int **_fstat** (int file, struct stat ∗st)
- int **_isatty** (int file)
- int **_lseek** (int file, int ptr, int dir)
- int **_open** (char ∗path, int flags,...)
- int **_wait** (int ∗status)
- int **_unlink** (char ∗name)
- int **_times** (struct tms ∗buf)
- int **_stat** (char ∗file, struct stat ∗st)
- int **_link** (char ∗old, char ∗new)
- int **_fork** (void)
- int **_execve** (char ∗name, char ∗∗argv, char ∗∗env)

Variables

- char ∗∗ **environ** = __env

6.15.1 Detailed Description

STM32CubeIDE Minimal System calls file.

Author

Auto-generated by STM32CubeIDE

```
For more information about which c-functions
need which of these lowlevel functions
please consult the Newlib libc-manual
```

Attention

6.16 Src/sysmem.c File Reference

STM32CubeIDE System Memory calls file.

```
#include <errno.h>
#include <stdint.h>
```

Functions

- void * _sbrk (ptrdiff_t incr)

  _sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

6.16.1 Detailed Description

STM32CubeIDE System Memory calls file.

Author

Generated by STM32CubeIDE

```
For more information about which C functions
need which of these lowlevel functions
please consult the newlib libc manual
```

Attention

6.16.2 Function Documentation

6.16.2.1 _sbrk()

```
void * _sbrk (
            ptrdiff_t incr )
```

_sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

```
* ##########################################################################
* #  .data  #  .bss  #       newlib heap       #          MSP stack          #
* #         #        #                         # Reserved by _Min_Stack_Size #
* ##########################################################################
* ^-- RAM start      ^-- _end                             _estack, RAM end --^
*
```

This implementation starts allocating at the '_end' linker symbol The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack The implementation considers '_estack' linker symbol to be RAM end NOTE: If the MSP stack, at any point during execution, grows larger than the reserved size, please increase the '_Min_Stack_Size'.

Parameters

incr	Memory size

Returns

Pointer to allocated memory

6.17 Src/system_stm32f0xx.c File Reference

CMSIS Cortex-M0 Device Peripheral Access Layer System Source File.

```
#include "stm32f0xx.h"
```

Macros

- #define HSE_VALUE ((uint32_t)8000000)
- #define HSI_VALUE ((uint32_t)8000000)
- #define HSI48_VALUE ((uint32_t)48000000)

Functions

- void SystemInit (void)

  Setup the microcontroller system.
- void SystemCoreClockUpdate (void)

  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Variables

- uint32_t **SystemCoreClock** = 8000000
- const uint8_t **AHBPrescTable** [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t **APBPrescTable** [8] = {0, 0, 0, 0, 1, 2, 3, 4}

6.17.1 Detailed Description

CMSIS Cortex-M0 Device Peripheral Access Layer System Source File.

Author

MCD Application Team

1. This file provides two functions and one global variable to be called from user application:
   - SystemInit(): This function is called at startup just after reset and before branch to main program. This call is made inside the "startup_stm32f0xx.s" file.
   - SystemCoreClock variable: Contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.
   - SystemCoreClockUpdate(): Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.

Attention

Index