DRIVER AND LIBRARY GUIDE

MPU6050

Prepared for:

PNR Group

Prepared by:

Majid Derhambakhsh

December 5, 2022

Content

1.1 MPU6050 library structures

1.1.1 MPU_INT_CFG_OPT_TypeDef

Data Fields

- **MPU_INT_CFG_TypeDef IntLevel**
- **MPU_INT_CFG_TypeDef IntOpen**
- **MPU_INT_CFG_TypeDef LatchIntEn**

Field Documentation

- ***MPU_INT_CFG_TypeDef MPU_INT_CFG_OPT_TypeDef:: IntLevel***

  This bit set the logic level for the INT pin
- ***MPU_INT_CFG_TypeDef MPU_INT_CFG_OPT_TypeDef:: IntOpen***

  This bit set the INT pin mode (Push pull / Open drain)
- ***MPU_INT_CFG_TypeDef MPU_INT_CFG_OPT_TypeDef:: LatchIntEn***

  This bit set the INT pin pulse mode

1.1.2 MPU_RawTypeDef

Data Fields

- **int16_t X**
- **int16_t Y**
- **int16_t Z**

Field Documentation

- ***int16_t MPU_RawTypeDef:: X***

  value of X position
- ***int16_t MPU_RawTypeDef:: Y***

  value of Y position
- ***int16_t MPU_RawTypeDef:: Z***

  value of Z position

1.1.3 MPU_XYZTypeDef

Data Fields

- **float X**
- **float Y**
- **float Z**

Field Documentation

- ***float MPU_XYZTypeDef:: X***

  value of X position
- ***float MPU_XYZTypeDef:: Y***

  value of Y position
- ***float MPU_XYZTypeDef:: Z***

  value of Z position

1.1.4 MPU_RPYTypeDef

Data Fields

- **float Roll**
- **float Pitch**
- **float Yaw**

Field Documentation

- ***float MPU_RPYTypeDef:: Roll***

  value of Roll angle
- ***float MPU_RPYTypeDef:: Pitch***

  value of Pitch angle
- ***float MPU_RPYTypeDef:: Yaw***

  value of Yaw angle

1.1.5 MPU_TypeDef

Data Fields

- **I2C_HandleTypeDef *I2Cx**
- **uint8_t SampleRateDivider**
- **MPU_AddTypeDef Address**
- **MPU_EXT_SYNC_SET_TypeDef ExtSync**
- **MPU_DLPF_CFG_TypeDef DigitalLowPassFilter**
- **MPU_GFS_SEL_TypeDef GyroFullScaleRange**
- **MPU_AFS_SEL_TypeDef AccelFullScaleRange**
- **MPU_INT_CFG_OPT_TypeDef InterruptConfig**
- **MPU_INT_EN_TypeDef InterruptEnable**
- **MPU_CLKSEL_TypeDef ClockSelection**
- **float GyroSensitivity**
- **float AccelSensitivity**

Field Documentation

- ***I2C_HandleTypeDef MPU_TypeDef:: *I2Cx***

  Specifies the I2C peripheral in STM32

- ***uint8_t MPU_TypeDef:: SampleRateDivider***

  This parameter specifies the divider from the gyroscope output rate used to generate the Sample Rate
- ***MPU_AddTypeDef MPU_TypeDef:: Address***

  The MPU6050 I2C device address
- ***MPU_EXT_SYNC_SET_TypeDef MPU_TypeDef:: ExtSync***

  This parameter configures the external Frame Synchronization (FSYNC) pin sampling for both the gyroscopes and accelerometers
- ***MPU_DLPF_CFG_TypeDef MPU_TypeDef:: DigitalLowPassFilter***

  This parameter configures the Digital Low Pass Filter (DLPF) setting for both the gyroscopes and accelerometers
- ***MPU_GFS_SEL_TypeDef MPU_TypeDef:: GyroFullScaleRange***

  This parameter configures the gyroscope full scale range
- ***MPU_AFS_SEL_TypeDef MPU_TypeDef:: AccelFullScaleRange***

  This parameter configures the accelerometer full scale range
- ***MPU_INT_CFG_OPT_TypeDef MPU_TypeDef:: InterruptConfig***

  This parameter configures the behavior of the interrupt signals at the INT pins
- ***MPU_INT_EN_TypeDef MPU_TypeDef:: InterruptEnable***

  This parameter enables interrupt generation by interrupt sources
- ***MPU_CLKSEL_TypeDef MPU_TypeDef:: ClockSelection***

  This parameter allows the user to configure the clock source
- ***float MPU_TypeDef:: GyroSensitivity***

  The gyroscope sensitivity of MPU6050
- ***float MPU_TypeDef:: AccelSensitivity***

  The accelerometer sensitivity of MPU6050

1.2 MPU6050 library API description

1.2.1 How to use this library

This library can be used as follows:

1. Config MCU I2C and initialize it
2. Add library Header and Source file in your project
3. Config the library in ***"mpu6050_conf.h"***
4. Create MPU6050 object with ***MPU_TypeDef*** type and set parameters
5. Initialize MPU6050 with ***MPU6050_Init***
6. Use MPU6050 operation functions

1.2.2 Initialization and de-initialization functions

This section provides functions allowing to:

- Initialize and configure the MPU-60X0 IMU

This section contains the following APIs:

- *MPU6050_Init()*
- *MPU6050_AutoInit()*
- *MPU6050_DefInit()*

1.2.3 Operation functions

This section contains the following APIs:

- *MPU6050_IsReady()*
- *MPU6050_Reset()*
- *MPU6050_SetDeviceID()*
- *MPU6050_GetDeviceID()*
- *MPU6050_GetRawAccel()*
- *MPU6050_GetRawGyro()*
- *MPU6050_GetRawTemp()*
- *MPU6050_GetAccel()*
- *MPU6050_GetGyro()*
- *MPU6050_GetTemp()*
- *MPU6050_GetRoll()*
- *MPU6050_GetPitch()*
- *MPU6050_GetYaw()*
- *MPU6050_GetRPY()*

1.2.4 Detailed description of functions

MPU6050_Init

Function name	**MPU_StatusTypeDef MPU6050_Init (MPU_TypeDef *MPUx, uint16_t Timeout)**
Function description	This function is used to initialize MPU-60X0 IMU
Parameters	• **MPUx:** pointer to MPU struct • **Timeout:** timeout duration
Return values	• Status of command transmission

MPU6050_AutoInit

Function name	**MPU_StatusTypeDef MPU6050_AutoInit (MPU_TypeDef *MPUx, uint16_t Timeout)**

Function description This function is used to initialize automatically MPU-60X0 IMU

Parameters

- **MPUx:** pointer to MPU struct
- **Timeout:** timeout duration

Return values

- Status of command transmission

MPU6050_DefInit

Function name **MPU_StatusTypeDef MPU6050_DefInit (MPU_TypeDef *MPUx, uint16_t Timeout)**

Function description This function is used to initialize MPU-60X0 IMU by default configuration

Parameters

- **MPUx:** pointer to MPU struct
- **Timeout:** timeout duration

Return values

- Status of command transmission

MPU6050_IsReady

Function name **MPU_StatusTypeDef MPU6050_IsReady (MPU_TypeDef *MPUx, uint16_t Timeout)**

Function description This function is used to connection status of MPU-60X0

Parameters

- **MPUx:** pointer to MPU struct
- **Timeout:** timeout duration

Return values

- Status of device connection

MPU6050_Reset

Function name **MPU_StatusTypeDef MPU6050_Reset (MPU_TypeDef *MPUx, uint16_t Timeout)**

Function description This function is used to reset MPU-60X0

Parameters

- **MPUx:** pointer to MPU struct
- **Timeout:** timeout duration

Return values

- Status of command transmission

MPU6050_SetDeviceID

Function name **MPU_StatusTypeDef MPU6050_SetDeviceID (MPU_TypeDef *MPUx, uint8_t ID, uint16_t Timeout)**

Function description This function is used to set MPU-60X0 device ID

Parameters

- **MPUx:** pointer to MPU struct
- **ID:** ID of MPU-60X0
- **Timeout:** timeout duration

Return values

- Status of command transmission

MPU6050_GetDeviceID

Function name **MPU_StatusTypeDef MPU6050_GetDeviceID (MPU_TypeDef *MPUx, uint8_t *ID, uint16_t Timeout)**

Function description This function is used to get device ID of MPU-60X0

Parameters

- **MPUx:** pointer to MPU struct
- **ID:** pointer to store device ID
- **Timeout:** timeout duration

Return values

- Status of command transmission

MPU6050_GetRawAccel

Function name **MPU_StatusTypeDef MPU6050_GetRawAccel (MPU_TypeDef *MPUx, MPU_RawTypeDef *AccelRaw, uint16_t Timeout)**

Function description This function is used to get the raw value of the accelerometer in three axis

Parameters

- **MPUx:** pointer to MPU struct
- **AccelRaw:** pointer to the raw value of the accelerometer for three axis
- **Timeout:** timeout duration

Return values

- Status of command transmission

MPU6050_GetRawGyro

Function name **MPU_StatusTypeDef MPU6050_GetRawGyro (MPU_TypeDef *MPUx, MPU_RawTypeDef *GyroRaw, uint16_t Timeout)**

Function description This function is used to get the raw value of the gyroscope in three axis

Parameters

- **MPUx:** pointer to MPU struct
- **GyroRaw:** pointer to the raw value of the gyroscope for three axis
- **Timeout:** timeout duration

Return values

- Status of command transmission

MPU6050_GetRawTemp

Function name **MPU_StatusTypeDef MPU6050_GetRawTemp (MPU_TypeDef *MPUx, int16_t *Temp, uint16_t Timeout)**

Function description This function is used to get the raw value of the temperature

Parameters

- **MPUx:** pointer to MPU struct
- **Temp:** pointer to the raw value of the temperature
- **Timeout:** timeout duration

Return values

- Status of command transmission

MPU6050_GetAccel

Function name **MPU_StatusTypeDef MPU6050_GetAccel (MPU_TypeDef *MPUx, MPU_XYZTypeDef *Accel, uint16_t Timeout)**

Function description This function is used to get the value of the accelerometer in three axis

Parameters

- **MPUx:** pointer to MPU struct
- **Accel:** pointer to the value of the accelerometer for three axis
- **Timeout:** timeout duration

Return values

- Status of command transmission

MPU6050_GetGyro

Function name **MPU_StatusTypeDef MPU6050_GetGyro (MPU_TypeDef *MPUx, MPU_XYZTypeDef *Gyro, uint16_t Timeout)**

Function description This function is used to get the value of the gyroscope in three axis

Parameters

- **MPUx:** pointer to MPU struct
- **Gyro:** pointer to the value of the gyroscope for three axis
- **Timeout:** timeout duration

Return values

- Status of command transmission

MPU6050_GetTemp

Function name **MPU_StatusTypeDef MPU6050_GetTemp (MPU_TypeDef *MPUx, float *Temp, uint16_t Timeout)**

Function description This function is used to get the value of the temperature

Parameters

- **MPUx:** pointer to MPU struct
- **Temp:** pointer to the value of the temperature
- **Timeout:** timeout duration

Return values

- Status of command transmission

MPU6050_GetRoll

Function name **MPU_StatusTypeDef MPU6050_GetRoll (MPU_TypeDef *MPUx, float *Roll, uint16_t Timeout)**

Function description This function is used to get the value of the Roll angle

Parameters

- **MPUx:** pointer to MPU struct
- **Roll:** pointer to the value of the Roll angle
- **Timeout:** timeout duration

Return values

- Status of command transmission

MPU6050_GetPitch

Function name **MPU_StatusTypeDef MPU6050_GetPitch (MPU_TypeDef *MPUx, float *Pitch, uint16_t Timeout)**

Function description This function is used to get the value of the Pitch angle

Parameters

- **MPUx:** pointer to MPU struct
- **Pitch:** pointer to the value of the Pitch angle
- **Timeout:** timeout duration

Return values

- Status of command transmission

MPU6050_GetYaw

Function name **MPU_StatusTypeDef MPU6050_GetYaw (MPU_TypeDef *MPUx, float *Yaw, uint16_t Timeout)**

Function description This function is used to get the value of the Yaw angle

Parameters

- **MPUx:** pointer to MPU struct
- **Yaw:** pointer to the value of the Yaw angle
- **Timeout:** timeout duration

Return values

- Status of command transmission

MPU6050_GetRPY

Function name **MPU_StatusTypeDef MPU6050_GetRPY (MPU_TypeDef *MPUx, MPU_RPYTypeDef *RPY, uint16_t Timeout)**

Function description This function is used to get the value of the Roll, Pitch, and Yaw angle

Parameters

- **MPUx:** pointer to MPU struct
- **RPY:** pointer to the value of the Roll, Pitch, and Yaw angle
- **Timeout:** timeout duration

Return values

- Status of command transmission

1.3 MPU6050 library defines

1.3.1 MPU6050

MPU6050 Status

MPU_ERROR

MPU_OK

MPU6050 Address

MPU_ADD_LOW

MPU_ADD_HIGH

MPU6050 Timing

MPU_WAKEUP_TIME_MS

MPU6050 Digital Low Pass Filter

MPU_DLPF_CFG_260A_256G_HZ

MPU_DLPF_CFG_184A_188G_HZ

MPU_DLPF_CFG_94A_98G_HZ

MPU_DLPF_CFG_44A_42G_HZ

MPU_DLPF_CFG_21A_20G_HZ

MPU_DLPF_CFG_10_HZ

MPU_DLPF_CFG_5_HZ

MPU6050 External Sync

MPU_ES_INPUT_DISABLE

MPU_ES_TEMP_OUT_L

MPU_ES_GYRO_XOUT_L

MPU_ES_GYRO_YOUT_L

MPU_ES_GYRO_ZOUT_L

MPU_ES_ACCEL_XOUT_L

MPU_ES_ACCEL_YOUT_L

MPU_ES_ACCEL_ZOUT_L

MPU6050 Gyro Full Scale Range

MPU_GYRO_FULL_SCALE_RANGE_250

MPU_GYRO_FULL_SCALE_RANGE_500

MPU_GYRO_FULL_SCALE_RANGE_1000

MPU_GYRO_FULL_SCALE_RANGE_2000

MPU6050 Accelerometer Full Scale Range

MPU_ACCEL_FULL_SCALE_RANGE_2G

MPU_ACCEL_FULL_SCALE_RANGE_4G

MPU_ACCEL_FULL_SCALE_RANGE_8G

MPU_ACCEL_FULL_SCALE_RANGE_16G

MPU6050 Interrupt Pin Logic Level

MPU_INT_LEVEL_ACTIVE_HIGH

MPU_INT_LEVEL_ACTIVE_LOW

MPU6050 Interrupt Pin Mode

MPU_INT_OPEN_PUSH_PULL

MPU_INT_OPEN_OPEN_DRAIN

MPU6050 Interrupt Pin Pulse Mode

MPU_LATCH_INT_EN_50US_PULSE

MPU_LATCH_INT_EN_INTERRUPT_CLEARED

MPU6050 Interrupt Enable

MPU_INT_DISABLE

MPU_INT_DATA_RDY_EN

MPU_INT_I2C_MST_INT_EN

MPU_INT_FIFO_OFLOW_EN

MPU6050 Clock Select

MPU_CLKSEL_INTERNAL_8MHZ_OSCILLATOR

MPU_CLKSEL_X_AXIS_GYROSCOPE_REFERENCE

MPU_CLKSEL_Y_AXIS_GYROSCOPE_REFERENCE

MPU_CLKSEL_Z_AXIS_GYROSCOPE_REFERENCE

MPU_CLKSEL_EXTERNAL_32768HZ_REFERENCE

MPU_CLKSEL_EXTERNAL_19200KHZ_REFERENCE

MPU_CLKSEL_STOP

MPU6050 Exported Macros

__MPU_SET_ID

__MPU_GET_ID

1.4 MPU6050 Library Configuration

Open "stm32_i2c_conf.h" to configure I2C library (for STM32)

- Define MCU series, for example:

  + in this section, you should define the microcontroller of your project

  ```
  /* ---------- Configuration --------- */
  #define STM32XX
  ```

  + XX: microcontroller series, for example:

    - STM32F1 for F1 series

    - STM32L0 for L0 series

- Set buffer size in configuration section, for example:

  + in this section, you should set the buffer size to manage data

  ```
  /* ---------- Configuration --------- */
  #define _MEM_DEF_VAL_BUFF_LENGTH Size
  ```

  + Size: the buffer size (Range: 1 ~ x)

Open "mpu6050_conf.h" to configure library

- Add requirement driver / libraries, for example:

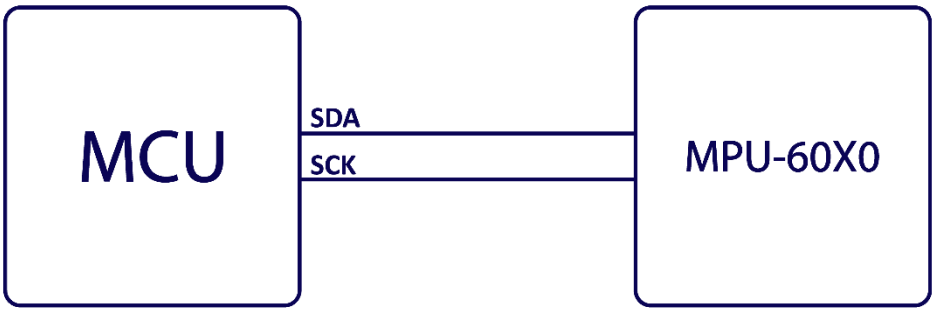
  + in this section, you should include the required libraries of the mpu6050 library

  ```
  /* ------ Required Driver.Library ------- */
  #include "math_ex.h"
  #include "i2c_library.h"
  ```

+ i2c_unit.h for AVR series

+ stm32_i2c.h for STM32 series

2. Circuit Design



3. Examples

- Example 1: Initialize and use MPU6050 with STM32

```
#include "main.h"
#include "i2c.h"
#include "usart.h"

#include "mpu6050.h"

char msg[50];

int main()
{

      /* MCU Configuration--------------------------------------------------------*/

      /*Reset of all peripherals, Initializes the Flash interface and the Systick*/
      HAL_Init();

      /* USER CODE BEGIN Init */

      /* USER CODE END Init */

      /* Configure the system clock */
      SystemClock_Config();

      /* USER CODE BEGIN SysInit */

      /* USER CODE END SysInit */

      /* Initialize all configured peripherals */
      MX_GPIO_Init();
      MX_I2C1_Init();
```

```
	MX_USART6_UART_Init();
	/* USER CODE BEGIN 2 */

	/* ---------- MPU6050 Setup --------- */
	MPU_TypeDef    IMU1;
	MPU_XYZTypeDef AccData;
	MPU_XYZTypeDef GyroData;

	IMU1.I2Cx                       = &hi2c1;
	IMU1.Address                    = MPU_ADD_LOW;
	IMU1.SampleRateDivider          = MPU_CLOCK_DIVIDER_8;
	IMU1.DigitalLowPassFilter       = MPU_DLPF_CFG_5_HZ;
	IMU1.InterruptEnable            = MPU_INT_DATA_RDY_EN;
	IMU1.ExtSync                    = MPU_ES_INPUT_DISABLE;
	IMU1.InterruptConfig.IntOpen    = MPU_INT_OPEN_PUSH_PULL;
	IMU1.InterruptConfig.IntLevel   = MPU_INT_LEVEL_ACTIVE_HIGH;
	IMU1.InterruptConfig.LatchIntEn = MPU_LATCH_INT_EN_50US_PULSE;
	IMU1.GyroFullScaleRange         = MPU_GYRO_FULL_SCALE_RANGE_2000;
	IMU1.AccelFullScaleRange        = MPU_ACCEL_FULL_SCALE_RANGE_16G;
	IMU1.ClockSelection             = MPU_CLKSEL_X_AXIS_GYROSCOPE_REFERENCE;

	MPU6050_Init(&IMU1, 100);

	/* Device Check */
	if (MPU6050_IsReady(&IMU1, 10, 100) == MPU_OK)
	{
		HAL_UART_Transmit(&huart6, (uint8_t *)"Is Ready\r\n", strlen("Is
				Ready\r\n"), 100);
	}
	else
	{
		HAL_UART_Transmit(&huart6, (uint8_t *)"Not Ready\r\n", strlen("Not
				Ready\r\n"), 100);
	}

	while(1)
	{
		/* :::::::::: Read Sensor Data :::::::::: */
		MPU6050_GetAccel(&IMU1, &AccData, 100);
		MPU6050_GetGyro(&IMU1, &GyroData, 100);
		HAL_Delay(100);

		sprintf(msg ,"AX:%f,AY:%f,AZ:%f,GX:%f,GY:%f,GZ:%f\r\n", AccData.X,
			AccData.Y, AccData.Z, GyroData.X, GyroData.Y, GyroData.Z);
		HAL_UART_Transmit(&huart6, (uint8_t *)msg, strlen(msg), 100);
	}
	/* Loop forever */
}
```

4. Requirements

1. HAL driver in STM32 series
2. stm32_i2c driver in STM32 (Included)
3. i2c_unit driver in AVR (Included)
4. math_ex header (Included)

5. Important tips

1. All functions are written as CamelCase

6. Error and Warning's

- **Error's**
  - None

- **Warning's**
  - None

7. License