STM32F3-Discovery BSP User Manual

Private Types

STM32F3-Discovery ACCELEROMETER

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by doxygen 1.7.6.1
Private Constants

STM32F3-DISCOVERY ACCELEROMETER

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP
User Manual by doxygen 1.7.6.1
# Private Macros

STM32F3-DISCOVERY ACCELEROMETER

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by [doxygen](http://www.stackoverflow.com) 1.7.6.1
STM32F3-Discovery BSP User Manual

Exported Types

STM32F3-DISCOVERY ACCELEROMETER

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by docxosgen 1.7.6.1
STM32F3-Discovery BSP User Manual

Exported Macros

STM32F3-DISCOVERY ACCELEROMETER

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by doxygen 1.7.6.1
STM32F3-Discovery BSP User Manual

Private Functions

STM32F3-DISCOVERY ACCELEROMETER

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by doxygen 1.7.6.1
STM32F3-Discovery BSP User Manual

Private Types

STM32F3-DISCOVERY GYROSCOPE

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by doxygen 1.7.6.1
Private Constants
Private Macros

STM32F3-DISCOVERY GYROSCOPE
### Private Functions

**STM32F3-Discovery GYROSCOPE**

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by [doxygen](http://www.stackoverflow.com) 1.7.6.1
STM32F3-Discovery BSP User Manual

Exported Constants

STM32F3-DISCOVERY GYROSCOPE

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by doxygen 1.7.6.1
Exported Macros

STM32F3-DISCOVERY_GYROSCOPE

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by doxygen 1.7.6.1
STM32F3-Discovery BSP User Manual

Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- _ _ -

- __STM32F3_DISCO_BSP_VERSION : stm32f3_discovery.c
- __STM32F3_DISCO_BSP_VERSION_MAIN : stm32f3_discovery.c
- __STM32F3_DISCO_BSP_VERSION_RC : stm32f3_discovery.c
- __STM32F3_DISCO_BSP_VERSION_SUB1 : stm32f3_discovery.c
- __STM32F3_DISCO_BSP_VERSION_SUB2 : stm32f3_discovery.c

- a -

- ACCELERO_DRDY_EXTI_IRQn : stm32f3_discovery.h
- ACCELERO_DRDY_GPIO_CLK_DISABLE : stm32f3_discovery.h
- ACCELERO_DRDY_GPIO_CLK_ENABLE : stm32f3_discovery.h
- ACCELERO_DRDY_GPIO_PORT : stm32f3_discovery.h
- ACCELERO_DRDY_PIN : stm32f3_discovery.h
- ACCELERO_ERROR : stm32f3_discovery_accelerometer.h
- ACCELERO_I2C_ADDRESS : stm32f3_discovery.h
- ACCELERO_INT1_EXTI_IRQn : stm32f3_discovery.h
- ACCELERO_INT1_PIN : stm32f3_discovery.h
- ACCELERO_INT2_EXTI_IRQn : stm32f3_discovery.h
- b -

```plaintext
- ACCELERO_INT2_PIN : stm32f3_discovery.h
- ACCELERO_INT_GPIO_CLK_DISABLE : stm32f3_discovery.h
- ACCELERO_INT_GPIO_CLK_ENABLE : stm32f3_discovery.h
- ACCELERO_INT_GPIO_PORT : stm32f3_discovery.h
- ACCELERO_OK : stm32f3_discovery_accelerometer.h
- ACCELERO_StatusTypeDef : stm32f3_discovery_accelerometer.h
- ACCELERO_TIMEOUT : stm32f3_discovery_accelerometer.h
- AccelerometerDrv : stm32f3_discovery_accelerometer.c

- BSP_ACCELERO_GetXYZ() : 
  stm32f3_discovery_accelerometer.c
- BSP_ACCELERO_Init() : stm32f3_discovery_accelerometer.c
- BSP_ACCELERO_Reset() : 
  stm32f3_discovery_accelerometer.c
- BSP_GetVersion() : stm32f3_discovery.c
- BSP_GYRO_DisableIT() : stm32f3_discovery_gyroscope.c
- BSP_GYRO_EnableIT() : stm32f3_discovery_gyroscope.c
- BSP_GYRO_GetXYZ() : stm32f3_discovery_gyroscope.c
- BSP_GYRO_Init() : stm32f3_discovery_gyroscope.c
- BSP_GYRO_ITConfig() : stm32f3_discovery_gyroscope.c
- BSP_GYRO_ReadID() : stm32f3_discovery_gyroscope.c
- BSP_GYRO_Reset() : stm32f3_discovery_gyroscope.c
- BSP_LED_Init() : stm32f3_discovery.c
- BSP_LED_Off() : stm32f3_discovery.c
- BSP_LED_On() : stm32f3_discovery.c
- BSP_LED_Toggle() : stm32f3_discovery.c
- BSP_PB_GetState() : stm32f3_discovery.c
- BSP_PB_Init() : stm32f3_discovery.c
- BUTTON_IRQn : stm32f3_discovery.c
- BUTTON_MODE_EXTI : stm32f3_discovery.h
- BUTTON_MODE_GPIO : stm32f3_discovery.h
- BUTTON_PIN : stm32f3_discovery.c
- BUTTON_PORT : stm32f3_discovery.c
- Button_TypeDef : stm32f3_discovery.h
```
- c -

- d -

- g -
- i -

- LED10 : stm32f3_discovery.h
  LED10_GPIO_CLK_DISABLE : stm32f3_discovery.h
  LED10_GPIO_CLK_ENABLE : stm32f3_discovery.h
- LED10_GPIO_PORT : stm32f3_discovery.h
- LED10_PIN : stm32f3_discovery.h
- LED3 : stm32f3_discovery.h
- LED3_GPIO_CLK_DISABLE : stm32f3_discovery.h
- LED3_GPIO_CLK_ENABLE : stm32f3_discovery.h
- LED3_GPIO_PORT : stm32f3_discovery.h
- LED3_PIN : stm32f3_discovery.h
- LED4 : stm32f3_discovery.h
- LED4_GPIO_CLK_DISABLE : stm32f3_discovery.h
- LED4_GPIO_CLK_ENABLE : stm32f3_discovery.h
- LED4_GPIO_PORT : stm32f3_discovery.h
- LED4_PIN : stm32f3_discovery.h
- LED5 : stm32f3_discovery.h
- LED5_GPIO_CLK_DISABLE : stm32f3_discovery.h
- LED5_GPIO_CLK_ENABLE : stm32f3_discovery.h
- LED5_GPIO_PORT : stm32f3_discovery.h
- LED5_PIN : stm32f3_discovery.h
- LED6 : stm32f3_discovery.h
- LED6_GPIO_CLK_DISABLE : stm32f3_discovery.h
- LED6_GPIO_CLK_ENABLE : stm32f3_discovery.h
- LED6_GPIO_PORT : stm32f3_discovery.h
- LED6_PIN : stm32f3_discovery.h
- LED7 : stm32f3_discovery.h
- LED7_GPIO_CLK_DISABLE : stm32f3_discovery.h
- LED7_GPIO_CLK_ENABLE : stm32f3_discovery.h
- LED7_GPIO_PORT : stm32f3_discovery.h
- LED7_PIN : stm32f3_discovery.h
- LED8 : stm32f3_discovery.h
- LED8_GPIO_CLK_DISABLE : stm32f3_discovery.h
- LED8_GPIO_CLK_ENABLE : stm32f3_discovery.h
- LED8_GPIO_PORT : stm32f3_discovery.h
- LED8_PIN : stm32f3_discovery.h
- LED9 : stm32f3_discovery.h
- LED9_GPIO_CLK_DISABLE : stm32f3_discovery.h
- LED9_GPIO_CLK_ENABLE : stm32f3_discovery.h
- LED9_GPIO_PORT : stm32f3_discovery.h
- LED9_PIN : stm32f3_discovery.h
• LED_BLUE : stm32f3_discovery.h
• LED_BLUE_2 : stm32f3_discovery.h
• LED_GREEN : stm32f3_discovery.h
• LED_GREEN_2 : stm32f3_discovery.h
• LED_ORANGE : stm32f3_discovery.h
• LED_ORANGE_2 : stm32f3_discovery.h
• LED_PIN : stm32f3_discovery.c
• LED_PORT : stm32f3_discovery.c
• LED_RED : stm32f3_discovery.h
• LED_RED_2 : stm32f3_discovery.h
• Led_TypeDef : stm32f3_discovery.h
• LEDn : stm32f3_discovery.h
• LEDx_GPIO_CLK_DISABLE : stm32f3_discovery.h
• LEDx_GPIO_CLK_ENABLE : stm32f3_discovery.h

- m -

• MULTIPLEBYTE_CMD : stm32f3_discovery.h

- r -

• READWRITE_CMD : stm32f3_discovery.h

- s -

• SpiHandle : stm32f3_discovery.c
• SPIx_Error(): stm32f3_discovery.c
• SPIx_Init(): stm32f3_discovery.c
• SPIx_MspInit(): stm32f3_discovery.c
• SPIx_TIMEOUT_MAX : stm32f3_discovery.h
• SPIx_WriteRead() : stm32f3_discovery.c
• SpixTimeout : stm32f3_discovery.c

- u -

• USER_BUTTON_EXTI_IRQn : stm32f3_discovery.h
• USER_BUTTON_GPIO_CLK_DISABLE : stm32f3_discovery.h
• USER_BUTTON_GPIO_CLK_ENABLE : stm32f3_discovery.h
- USER_BUTTON_GPIO_PORT: `stm32f3_discovery.h`
- USER_BUTTON_PIN: `stm32f3_discovery.h`
STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>File List</td>
<td>Modules</td>
<td>Files</td>
<td>Directories</td>
</tr>
<tr>
<td>All</td>
<td>Functions</td>
<td>Variables</td>
<td>Enumerations</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>g</td>
<td>i</td>
</tr>
</tbody>
</table>

- **b** -

- BSP_ACCELERO_GetXYZ() :
  `stm32f3_discovery_accelerometer.c`
- BSP_ACCELERO_Init() : `stm32f3_discovery_accelerometer.c`
- BSP_ACCELERO_Reset() :
  `stm32f3_discovery_accelerometer.c`
- BSP_GetVersion() : `stm32f3_discovery.c`
- BSP_GYRO_DisableIT() : `stm32f3_discovery_gyroscope.c`
- BSP_GYRO_EnableIT() : `stm32f3_discovery_gyroscope.c`
- BSP_GYRO_GetXYZ() : `stm32f3_discovery_gyroscope.c`
- BSP_GYRO_Init() : `stm32f3_discovery_gyroscope.c`
- BSP_GYRO_ITConfig() : `stm32f3_discovery_gyroscope.c`
- BSP_GYRO_ReadID() : `stm32f3_discovery_gyroscope.c`
- BSP_GYRO_Reset() : `stm32f3_discovery_gyroscope.c`
- BSP_LED_Init() : `stm32f3_discovery.c`
- BSP_LED_Off() : `stm32f3_discovery.c`
- BSP_LED_On() : `stm32f3_discovery.c`
- BSP_LED_Toggle() : `stm32f3_discovery.c`
- BSP_PB_GetState() : `stm32f3_discovery.c`
- BSP_PB_Init() : `stm32f3_discovery.c`

- **c** -

- COMPASSACCELERO_IO_Init() : `stm32f3_discovery.c`
- COMPASSACCELERO_IO_ITConfig() : stm32f3_discovery.c
- COMPASSACCELERO_IO_Read() : stm32f3_discovery.c
- COMPASSACCELERO_IO_Write() : stm32f3_discovery.c

- G

  - GYRO_IO_Init() : stm32f3_discovery.c
  - GYRO_IO_Read() : stm32f3_discovery.c
  - GYRO_IO_Write() : stm32f3_discovery.c

- I

  - I2Cx_Error() : stm32f3_discovery.c
  - I2Cx_Init() : stm32f3_discovery.c
  - I2Cx_MspInit() : stm32f3_discovery.c
  - I2Cx_ReadData() : stm32f3_discovery.c
  - I2Cx_WriteData() : stm32f3_discovery.c

- S

  - SPIx_Error() : stm32f3_discovery.c
  - SPIx_Init() : stm32f3_discovery.c
  - SPIx_MspInit() : stm32f3_discovery.c
  - SPIx_WriteRead() : stm32f3_discovery.c
### STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>File List</th>
<th>File</th>
<th>Global Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Functions</td>
<td>Variables</td>
</tr>
<tr>
<td></td>
<td>Enumerations</td>
<td>Defines</td>
</tr>
</tbody>
</table>

- AccelerometerDrv : `stm32f3_discovery_accelerometer.c`
- BUTTON_IRQn : `stm32f3_discovery.c`
- BUTTON_PIN : `stm32f3_discovery.c`
- BUTTON_PORT : `stm32f3_discovery.c`
- GyroscopeDrv : `stm32f3_discovery_gyroscope.c`
- I2cHandle : `stm32f3_discovery.c`
- I2cxTimeout : `stm32f3_discovery.c`
- LED_PIN : `stm32f3_discovery.c`
- LED_PORT : `stm32f3_discovery.c`
- SpiHandle : `stm32f3_discovery.c`
- SpixTimeout : `stm32f3_discovery.c`

---

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by [doxygen](https://www.doxygen.org) 1.7.6.1
### STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>File List</td>
<td>Globals</td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>Functions</td>
<td>Variables</td>
<td>Enumerations</td>
</tr>
</tbody>
</table>

- ACCELERO_StatusTypeDef : `stm32f3_discovery_accelerometer.h`
- Button_TypeDef : `stm32f3_discovery.h`
- ButtonMode_TypeDef : `stm32f3_discovery.h`
- GYRO_StatusTypeDef : `stm32f3_discovery_gyroscope.h`
- Led_TypeDef : `stm32f3_discovery.h`

---

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by [doxygen](http://www.stackoverflow.com) 1.7.6.1
### STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>File List</td>
<td>Globals</td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>Functions</td>
<td>Variables</td>
</tr>
</tbody>
</table>

- ACCELERO_ERROR : [stm32f3_discovery_accelerometer.h](#)
- ACCELERO_OK : [stm32f3_discovery_accelerometer.h](#)
- ACCELERO_TIMEOUT : [stm32f3_discovery_accelerometer.h](#)
- BUTTON_MODE_EXTI : [stm32f3_discovery.h](#)
- BUTTON_MODE_GPIO : [stm32f3_discovery.h](#)
- BUTTON_USER : [stm32f3_discovery.h](#)
- GYRO_ERROR : [stm32f3_discovery_gyroscope.h](#)
- GYRO_OK : [stm32f3_discovery_gyroscope.h](#)
- GYRO_TIMEOUT : [stm32f3_discovery_gyroscope.h](#)
- LED10 : [stm32f3_discovery.h](#)
- LED3 : [stm32f3_discovery.h](#)
- LED4 : [stm32f3_discovery.h](#)
- LED5 : [stm32f3_discovery.h](#)
- LED6 : [stm32f3_discovery.h](#)
- LED7 : [stm32f3_discovery.h](#)
- LED8 : [stm32f3_discovery.h](#)
- LED9 : [stm32f3_discovery.h](#)
- LED_BLUE : [stm32f3_discovery.h](#)
- LED_BLUE_2 : [stm32f3_discovery.h](#)
- LED_GREEN : [stm32f3_discovery.h](#)
- LED_GREEN_2 : [stm32f3_discovery.h](#)
- LED_ORANGE : [stm32f3_discovery.h](#)
- LED_ORANGE_2 : [stm32f3_discovery.h](#)
- LED_RED : [stm32f3_discovery.h](#)
- LED_RED_2 : [stm32f3_discovery.h](#)
STM32F3-Discovery BSP User Manual

- _ _ -

- __STM32F3_DISCO_BSP_VERSION__ : `stm32f3_discovery.c`
- __STM32F3_DISCO_BSP_VERSION__ : `stm32f3_discovery.c`
- __STM32F3_DISCO_BSP_VERSION__ : `stm32f3_discovery.c`
- __STM32F3_DISCO_BSP_VERSION__ : `stm32f3_discovery.c`
- __STM32F3_DISCO_BSP_VERSION__ : `stm32f3_discovery.c`

- a -

- ACCELERO_DRDY_EXTI_IRQn : `stm32f3_discovery.h`
- ACCELERO_DRDY_GPIO_CLK_DISABLE : `stm32f3_discovery.h`
- ACCELERO_DRDY_GPIO_CLK_ENABLE : `stm32f3_discovery.h`
- ACCELERO_DRDY_GPIO_PORT : `stm32f3_discovery.h`
- ACCELERO_DRDY_PIN : `stm32f3_discovery.h`
- ACCELERO_I2C_ADDRESS : `stm32f3_discovery.h`
- ACCELERO_INT1_EXTI_IRQn : `stm32f3_discovery.h`
- ACCELERO_INT1_PIN : `stm32f3_discovery.h`
- ACCELERO_INT2_EXTI_IRQn : `stm32f3_discovery.h`
- ACCELERO_INT2_PIN : `stm32f3_discovery.h`
- ACCELERO_INT_GPIO_CLK_DISABLE : `stm32f3_discovery.h`
- b -

- d -

- g -
- i -

- l -

- i -

- l -

- i -

- l -
- m -

- MULTIPLEBYTE_CMD : stm32f3_discovery.h

- r -

- READWRITE_CMD : stm32f3_discovery.h

- s -

- SPIx_TIMEOUT_MAX : stm32f3_discovery.h

- u -

- USER_BUTTON_EXTI_IRQn : stm32f3_discovery.h
- USER_BUTTON_GPIO_CLK_DISABLE : stm32f3_discovery.h
- USER_BUTTON_GPIO_CLK_ENABLE : stm32f3_discovery.h
- USER_BUTTON_GPIO_PORT : stm32f3_discovery.h
- USER_BUTTON_PIN : stm32f3_discovery.h
stm32f3_discovery.c

File Reference

This file provides set of firmware functions to manage Leds and push-button available on STM32F3-DISCOVERY Kit from STMicroelectronics. More...

#include "stm32f3_discovery.h"

Go to the source code of this file.
Defines

<table>
<thead>
<tr>
<th>Define</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__STM32F3_DISCO_BSP_VERSION_MAIN</code></td>
<td>0x02</td>
</tr>
<tr>
<td>STMicroelectronics STM32F3 DISCOVERY BSP Driver version number V2.1.5.</td>
<td></td>
</tr>
<tr>
<td><code>__STM32F3_DISCO_BSP_VERSION_SUB1</code></td>
<td>0x01</td>
</tr>
<tr>
<td><code>__STM32F3_DISCO_BSP_VERSION_SUB2</code></td>
<td>0x05</td>
</tr>
<tr>
<td><code>__STM32F3_DISCO_BSP_VERSION_RC</code></td>
<td>0x00</td>
</tr>
<tr>
<td><code>__STM32F3_DISCO_BSP_VERSION</code></td>
<td></td>
</tr>
</tbody>
</table>
### Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td><code>I2Cx_Init</code> (void)</td>
<td>Discovery I2Cx Bus initialization.</td>
</tr>
<tr>
<td>static void</td>
<td><code>I2Cx_WriteData</code> (uint16_t Addr, uint8_t Reg, uint8_t Value)</td>
<td>Write a value in a register of the device through BUS.</td>
</tr>
<tr>
<td>static uint8_t</td>
<td><code>I2Cx_ReadData</code> (uint16_t Addr, uint8_t Reg)</td>
<td>Read a value in a register of the device through BUS.</td>
</tr>
<tr>
<td>static void</td>
<td><code>I2Cx_Error</code> (void)</td>
<td>I2C3 error treatment function.</td>
</tr>
<tr>
<td>static void</td>
<td><code>I2Cx_MspInit</code> (I2C_HandleTypeDef *hi2c)</td>
<td>Discovery I2Cx MSP Initialization.</td>
</tr>
<tr>
<td>static void</td>
<td><code>SPIx_Init</code> (void)</td>
<td>SPIx Bus initialization.</td>
</tr>
<tr>
<td>static uint8_t</td>
<td><code>SPIx_WriteRead</code> (uint8_t Byte)</td>
<td>Sends a Byte through the SPI interface and return the Byte received from the SPI bus.</td>
</tr>
<tr>
<td>static void</td>
<td><code>SPIx_Error</code> (void)</td>
<td>SPIx error treatment function.</td>
</tr>
<tr>
<td>static void</td>
<td><code>SPIx_MspInit</code> (SPI_HandleTypeDef *hspi)</td>
<td>SPI MSP Init.</td>
</tr>
<tr>
<td>void</td>
<td><code>GYRO_IO_Init</code> (void)</td>
<td>Configures the GYROSCOPE SPI interface.</td>
</tr>
<tr>
<td>void</td>
<td><code>GYRO_IO_Write</code> (uint8_t *pBuffer, uint8_t WriteAddr, uint16_t NumByteToWrite)</td>
<td>Writes one byte to the GYROSCOPE.</td>
</tr>
<tr>
<td>void</td>
<td><code>GYRO_IO_Read</code> (uint8_t *pBuffer, uint8_t ReadAddr, uint16_t NumByteToRead)</td>
<td>Reads a block of data from the GYROSCOPE.</td>
</tr>
<tr>
<td>void</td>
<td><code>COMPASSACCELERO_IO_Init</code> (void)</td>
<td>Configures COMPASS / ACCELEROMETER I2C interface.</td>
</tr>
<tr>
<td>void</td>
<td>COMPASSACCELERO_IO_ITConfig (void)</td>
<td>Configures COMPASS / ACCELERO click IT.</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>void</td>
<td>COMPASSACCELERO_IO_Write (uint16_t DeviceAddr, uint8_t RegisterAddr, uint8_t Value)</td>
<td>Writes one byte to the COMPASS / ACCELEROMETER.</td>
</tr>
<tr>
<td>uint8_t</td>
<td>COMPASSACCELERO_IO_Read (uint16_t DeviceAddr, uint8_t RegisterAddr)</td>
<td>Reads a block of data from the COMPASS / ACCELEROMETER.</td>
</tr>
<tr>
<td>uint32_t</td>
<td>BSP_GetVersion (void)</td>
<td>This method returns the STM32F3-DISCOVERY BSP Driver revision.</td>
</tr>
<tr>
<td>void</td>
<td>BSP_LED_Init (Led_TypeDef Led)</td>
<td>Configures LED GPIO.</td>
</tr>
<tr>
<td>void</td>
<td>BSP_LED_On (Led_TypeDef Led)</td>
<td>Turns selected LED On.</td>
</tr>
<tr>
<td>void</td>
<td>BSP_LED_Off (Led_TypeDef Led)</td>
<td>Turns selected LED Off.</td>
</tr>
<tr>
<td>void</td>
<td>BSP_LED_Toggle (Led_TypeDef Led)</td>
<td>Toggles the selected LED.</td>
</tr>
<tr>
<td>void</td>
<td>BSP_PB_Init (Button_TypeDef Button, ButtonMode_TypeDef ButtonMode)</td>
<td>Configures Push Button GPIO and EXTI Line.</td>
</tr>
<tr>
<td>uint32_t</td>
<td>BSP_PB_GetState (Button_TypeDef Button)</td>
<td>Returns the selected Push Button state.</td>
</tr>
</tbody>
</table>
## Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GPIO_TypeDef *</code></td>
<td>LED Port [LEDn]</td>
</tr>
<tr>
<td><code>const uint16_t</code></td>
<td>LED Pin [LEDn]</td>
</tr>
<tr>
<td><code>GPIO_TypeDef *</code></td>
<td>BUTTON Port [BUTTONn] = {USER_BUTTON_GPIO_PORT}</td>
</tr>
<tr>
<td><code>const uint16_t</code></td>
<td>BUTTON Pin [BUTTONn] = {USER_BUTTON_PIN}</td>
</tr>
<tr>
<td><code>const uint8_t</code></td>
<td>BUTTON IRQn [BUTTONn] = {USER_BUTTON_EXTI_IRQn}</td>
</tr>
<tr>
<td><code>uint32_t</code></td>
<td>SPIxTimeout = SPIx_TIMEOUT_MAX</td>
</tr>
<tr>
<td><code>static SPI_HandleTypeDef</code></td>
<td>SpiHandle</td>
</tr>
<tr>
<td><code>static I2C_HandleTypeDef</code></td>
<td>I2cHandle</td>
</tr>
<tr>
<td><code>uint32_t</code></td>
<td>I2CxTimeout = I2Cx_TIMEOUT_MAX</td>
</tr>
</tbody>
</table>
Detailed Description

This file provides set of firmware functions to manage Leds and push-button available on STM32F3-DISCOVERY Kit from STMicroelectronics.

Author:
    MCD Application Team

Attention:
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file stm32f3_discovery.c.
stm32f3_discovery.h
File Reference

This file contains definitions for STM32F3-Discovery's Leds, push-buttons hardware resources. More...

#include "stm32f3xx_hal.h"

Go to the source code of this file.
Defines

#define LEDn 8
#define LED6_PIN GPIO_PIN_15
#define LED6_GPIO_PORT GPIOE
#define LED6_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED6_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define LED8_PIN GPIO_PIN_14
#define LED8_GPIO_PORT GPIOE
#define LED8_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED8_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define LED10_PIN GPIO_PIN_13
#define LED10_GPIO_PORT GPIOE
#define LED10_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED10_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define LED9_PIN GPIO_PIN_12
#define LED9_GPIO_PORT GPIOE
#define LED9_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED9_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define LED7_PIN GPIO_PIN_11
#define LED7_GPIO_PORT GPIOE
#define LED7_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED7_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define LED5_PIN GPIO_PIN_10
#define LED5_GPIO_PORT GPIOE
#define LED5_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED5_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define LED3_PIN GPIO_PIN_9
#define LED3_GPIO_PORT GPIOE
#define LED3_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED3_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define LED4_PIN GPIO_PIN_8
#define LED4_GPIO_PORT GPIOE
```c
#define LED4_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED4_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define LEDx_GPIO_CLK_ENABLE(__LED__) (__HAL_RCC_GPIOE_CLK_ENABLE())
#define LEDx_GPIO_CLK_DISABLE(__LED__) (__HAL_RCC_GPIOE_CLK_DISABLE())
#define BUTTONn 1
#define USER_BUTTON_PIN GPIO_PIN_0
  push-button
#define USER_BUTTON_GPIO_PORT GPIOA
#define USER_BUTTON_GPIO_CLK_ENABLE() __HAL_RCC_GPIOA_CLK_ENABLE()
#define USER_BUTTON_GPIO_CLK_DISABLE() __HAL_RCC_GPIOA_CLK_DISABLE()
#define USER_BUTTON_EXTIIRQn EXTI0_IRQn
#define BUTTONx_GPIO_CLK_ENABLE(__BUTTON__) do { if ((__BUTTON__) == BUTTON_USER) USER_BUTTON_GPIO_CLK_ENABLE(); } while (0)
#define BUTTONx_GPIO_CLK_DISABLE(__BUTTON__) ((__BUTTON__) == BUTTON_USER) ? USER_BUTTON_GPIO_CLK_DISABLE() : 0
#define DISCOVERY_SPIx SPI1
  Definition for SPI Interface pins (SPI1 used)
#define DISCOVERY_SPIx_CLK_ENABLE() __HAL_RCC_SPI1_CLK_ENABLE()
#define DISCOVERY_SPIx_GPIO_PORT GPIOA /* GPIOA */
#define DISCOVERY_SPIx_AF GPIO_AF5_SPI1
#define DISCOVERY_SPIx_GPIO_CLK_ENABLE() __HAL_RCC_GPIOA_CLK_ENABLE()
#define DISCOVERY_SPIx_GPIO_CLK_DISABLE() __HAL_RCC_GPIOA_CLK_DISABLE()
#define DISCOVERY_SPIx_SCK_PIN GPIO_PIN_5 /* PA.05 */
#define DISCOVERY_SPIx_MISO_PIN GPIO_PIN_6 /* PA.06 */
#define DISCOVERY_SPIx_MOSI_PIN GPIO_PIN_7 /* PA.07 */
#define SPIx_TIMEOUT_MAX ((uint32_t)0x1000)
#define DISCOVERY_I2Cx I2C1
  Definition for I2C Interface pins (I2C1 used)
#define DISCOVERY_I2Cx_CLK_ENABLE() __HAL_RCC_I2C1_CLK_ENABLE()
#define DISCOVERY_I2Cx_CLK_DISABLE() __HAL_RCC_I2C1_CLK_DISABLE()
#define DISCOVERY_I2Cx_FORCE_RESET() __HAL_RCC_I2C1_FORCE_RESET()
#define DISCOVERY_I2Cx_RELEASE_RESET() __HAL_RCC_I2C1_RELEASE_RESET()
#define DISCOVERY_I2Cx_SCL_PIN GPIO_PIN_6 /* PB.06 */
#define DISCOVERY_I2Cx_SDA_PIN GPIO_PIN_7 /* PB.07 */
```

```c
#define DISCOVERY_I2Cx_GPIO_PORT GPIOB /* GPIOB */
#define DISCOVERY_I2Cx_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_ENABLE()
#define DISCOVERY_I2Cx_GPIO_CLK_DISABLE() __HAL_RCC_GPIOB_CLK_DISABLE()
#define DISCOVERY_I2Cx_AF GPIO_AF4_I2C1
#define I2Cx_TIMEOUT_MAX 0x10000
#define READWRITE_CMD ((uint8_t)0x80)
#define MULTIPLEBYTE_CMD ((uint8_t)0x40)
#define DUMMY_BYTE ((uint8_t)0x00)
#define GYRO_CS_LOW() HAL_GPIO_WritePin(GYRO_CS_GPIO_PORT, GPIO_PIN_RESET)
#define GYRO_CS_HIGH() HAL_GPIO_WritePin(GYRO_CS_GPIO_PORT, GPIO_PIN_SET)
#define GYRO_CS_GPIO_PORT GPIOE /* GPIOE */
#define GYRO_CS_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define GYRO_CS_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define GYRO_CS_PIN GPIO_PIN_3 /* PE.03 */
#define GYRO_INT_GPIO_PORT GPIOE /* GPIOE */
#define GYRO_INT_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define GYRO_INT_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define GYRO_INT1_PIN GPIO_PIN_0 /* PE.00 */
#define GYRO_INT1_EXTI_IRQn EXTI0_IRQn
#define GYRO_INT2_PIN GPIO_PIN_1 /* PE.01 */
#define GYRO_INT2_EXTI_IRQn EXTI1_IRQn
#define GYRO_INT_GPIO_PORT GPIOE /* GPIOE */
#define GYRO_INT_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define GYRO_INT_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define ACCELERO_I2C_ADDRESS 0x32
#define ACCELERO_DRDY_PIN GPIO_PIN_2 /* PE.02 */
#define ACCELERO_DRDY_GPIO_PORT GPIOE /* GPIOE */
#define ACCELERO_DRDY_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define ACCELERO_DRDY_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define ACCELERO_DRDY_EXTI_IRQn EXTI2_TSC_IRQn /* TAMP_STAMP_IRQn */
#define ACCELERO_INT_GPIO_PORT GPIOE /* GPIOE */
#define ACCELERO_INT_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define ACCELERO_INT_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
```

GYRO SPI Interface pins.

ACCELEROMETER I2C1 Interface pins.
```c
#define ACCELERO_INT1_PIN  GPIO_PIN_4 /* PE.04 */
#define ACCELERO_INT1 EXTI_IRQn  EXTI4_IRQHandler
#define ACCELERO_INT2_PIN  GPIO_PIN_5 /* PE.05 */
#define ACCELERO_INT2 EXTI_IRQn  EXTI9_5_IRQHandler
```
Enumerations

```c
enum Led_TypeDef {
    LED3 = 0, LED4 = 1, LED5 = 2, LED6 = 3,
    LED7 = 4, LED8 = 5, LED9 = 6, LED10 = 7,
    LED_GREEN = LED6, LED_ORANGE = LED5, LED_RED = LED3, LED_BLUE = LED4,
    LED_GREEN_2 = LED7, LED_ORANGE_2 = LED8, LED_RED_2 = LED10, LED_BLUE_2 = LED9
}
LED Types Definition. More...
```

```c
enum Button_TypeDef { BUTTON_USER = 0 }
BUTTON Types Definition. More...
```

```c
enum ButtonMode_TypeDef { BUTTON_MODE_GPIO = 0, BUTTON_MODE_EXTI = 1 }
```
### Functions

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint32_t</td>
<td><code>BSP_GetVersion (void)</code></td>
<td>This method returns the STM32F3-DISCOVERY BSP Driver revision.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_LED_Init (Led_TypeDef Led)</code></td>
<td>Configures LED GPIO.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_LED_On (Led_TypeDef Led)</code></td>
<td>Turns selected LED On.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_LED_Off (Led_TypeDef Led)</code></td>
<td>Turns selected LED Off.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_LED_Toggle (Led_TypeDef Led)</code></td>
<td>Toggles the selected LED.</td>
</tr>
<tr>
<td>void</td>
<td><code>BSP_PB_Init (Button_TypeDef Button, ButtonMode_TypeDef ButtonMode)</code></td>
<td>Configures Push Button GPIO and EXTI Line.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><code>BSP_PB_GetState (Button_TypeDef Button)</code></td>
<td>Returns the selected Push Button state.</td>
</tr>
</tbody>
</table>
**Detailed Description**

This file contains definitions for STM32F3-Discovery's Leds, push-buttons hardware resources.

**Author:**
MCD Application Team

**Attention:**
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file stm32f3_discovery.h.
## stm32f3_discovery_accelerometer.c File Reference

This file provides a set of functions needed to manage the ACCELEROMETER MEMS available on STM32F3-Discovery Kit.

More...

```c
#include "stm32f3_discovery_accelerometer.h"
```

Go to the source code of this file.
## Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uint8_t</code></td>
<td><code>BSP_ACCELERO_Init (void)</code></td>
<td></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_ACCELERO_Reset (void)</code></td>
<td>Reboot memory content of ACCELEROMETER.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_ACCELERO_GetXYZ (int16_t *pDataXYZ)</code></td>
<td>Get XYZ acceleration.</td>
</tr>
</tbody>
</table>
Variables

static ACCELERO_DrvTypeDef * AccelerometerDrv
Detailed Description

This file provides a set of functions needed to manage the ACCELEROMETER MEMS available on STM32F3-Discovery Kit.

Author:
MCD Application Team

Attention:
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file stm32f3_discovery_accelerometer.c.
stm32f3_discovery_accelerometer.h File Reference

This file contains definitions for `stm32f3_discovery_accelerometer.c` firmware driver. More...

```c
#include "stm32f3_discovery.h"
#include ".../Components/lsm303dlhc/lsm303dlhc.h"
```

Go to the source code of this file.
Enumerations

```
enum ACCELERO_StatusTypeDef {
    ACCELERO_OK = 0,
    ACCELERO_ERROR = 1,
    ACCELERO_TIMEOUT = 2
}
```
## Functions

<table>
<thead>
<tr>
<th></th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td><strong>BSP_ACCELELO_Init</strong> (void)</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_ACCELELO_Reset</strong> (void)</td>
<td>Reboot memory content of ACCELEROMETER.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_ACCELELO_GetXYZ</strong> (int16_t *pDataXYZ)</td>
<td>Get XYZ acceleration.</td>
</tr>
</tbody>
</table>
Detailed Description

This file contains definitions for `stm32f3_discovery_accelerometer.c` firmware driver.

**Author:**

MCD Application Team

**Attention:**
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file stm32f3_discovery_accelerometer.h.
This file provides a set of functions needed to manage the L3gd20 MEMS accelerometer available on STM32F3-Discovery Kit. More...

#include "stm32f3_discovery_gyroscope.h"

Go to the source code of this file.
### Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uint8_t</code></td>
<td><code>BSP_GYRO_Init (void)</code></td>
<td>Set GYROSCOPE Initialization.</td>
</tr>
<tr>
<td><code>uint8_t</code></td>
<td><code>BSP_GYRO_ReadID (void)</code></td>
<td>Read ID of Gyroscope component.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_GYRO_Reset (void)</code></td>
<td>Reboot memory content of GYROSCOPE.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_GYRO_ITConfig (GYRO_InterruptConfigTypeDef *pIntConfig)</code></td>
<td>Configure INT1 interrupt.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_GYRO_EnableIT (uint8_t IntPin)</code></td>
<td>Enable INT1 or INT2 interrupt.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_GYRO_DisableIT (uint8_t IntPin)</code></td>
<td>Disable INT1 or INT2 interrupt.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_GYRO_GetXYZ (float *pfData)</code></td>
<td>Get XYZ angular acceleration.</td>
</tr>
</tbody>
</table>
## Variables

| static GYRO_DrvTypeDef * GyroscopeDrv |  |
Detailed Description

This file provides a set of functions needed to manage the I3gd20 MEMS accelerometer available on STM32F3-Discovery Kit.

Author:
   MCD Application Team

| Attention:
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file stm32f3_discovery_gyroscope.c.
This file contains definitions for `stm32f3_discovery_gyroscope.c` firmware driver. More...

```c
#include "stm32f3_discovery.h" #include ".../Components/l3gd20/l3gd20.h"
```

Go to the source code of this file.
Enumerations

```c
enum GYRO_StatusTypeDef { GYRO_OK = 0, GYRO_ERROR = 1, GYRO_TIMEOUT = 2 }
```
## Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td><strong>BSP_GYRO_Init</strong> (void)</td>
<td>Set GYROSCOPE Initialization.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_GYRO_Reset</strong> (void)</td>
<td>Reboot memory content of GYROSCOPE.</td>
</tr>
<tr>
<td>uint8_t</td>
<td><strong>BSP_GYRO_ReadID</strong> (void)</td>
<td>Read ID of Gyroscope component.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_GYRO_ITConfig</strong> (GYRO_InterruptConfigTypeDef *pIntConfig)</td>
<td>Configure INT1 interrupt.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_GYRO_EnableIT</strong> (uint8_t IntPin)</td>
<td>Enable INT1 or INT2 interrupt.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_GYRO_DisableIT</strong> (uint8_t IntPin)</td>
<td>Disable INT1 or INT2 interrupt.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_GYRO_GetXYZ</strong> (float *pfData)</td>
<td>Get XYZ angular acceleration.</td>
</tr>
</tbody>
</table>
Detailed Description

This file contains definitions for *stm32f3_discovery_gyroscope.c* firmware driver.

**Author:**

MCD Application Team

**Attention:**
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file `stm32f3_discovery_gyroscope.h`.
STM32F3-Discovery BSP User Manual

Modules

Here is a list of all modules:

- **BSP**
  - STM32F3-DISCOVERY
    - STM32F3-DISCOVERY Common
      - Bus Operation functions
      - Link Operation functions
      - Private Constants
      - Private Variables
      - Exported Types
      - Exported Constants
        - STM32F3-DISCOVERY LED
        - STM32F3-DISCOVERY BUTTON
        - STM32F3-DISCOVERY COM
        - STM32F3-DISCOVERY COMPONENT
    - Exported Functions
  - STM32F3-DISCOVERY ACCELEROMETER
    - Private Types
    - Private Constants
    - Private Macros
    - Private Variables
    - Exported Types
    - Exported Constants
    - Exported Macros
    - Exported Functions
    - Private Functions
  - STM32F3-DISCOVERY GYROSCOPE
    - Private Types
## STM32F3-Discovery BSP User Manual

### File List

Here is a list of all files with brief descriptions:

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>stm32f3_discovery.c</code></td>
<td>This file provides set of firmware functions to manage LEDs and push-button available on STM32F3-Discovery Kit from STMicroelectronics.</td>
</tr>
<tr>
<td><code>stm32f3_discovery.h</code></td>
<td>This file contains definitions for STM32F3-Discovery's LEDs, push-buttons hardware resources.</td>
</tr>
<tr>
<td><code>stm32f3_discovery_accelerometer.c</code></td>
<td>This file provides a set of functions needed to manage the ACCELEROMETER MEMS available on STM32F3-Discovery Kit.</td>
</tr>
<tr>
<td><code>stm32f3_discovery_accelerometer.h</code></td>
<td>This file contains definitions for <code>stm32f3_discovery_accelerometer.c</code> firmware driver.</td>
</tr>
<tr>
<td><code>stm32f3_discovery_gyroscope.c</code></td>
<td>This file provides a set of functions needed to manage the l3gd20 MEMS accelerometer available on STM32F3-Discovery Kit.</td>
</tr>
<tr>
<td><code>stm32f3_discovery_gyroscope.h</code></td>
<td>This file contains definitions for <code>stm32f3_discovery_gyroscope.c</code> firmware driver.</td>
</tr>
</tbody>
</table>

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP
Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

- Firmware
  - Drivers
    - BSP
    - STM32F3-Discovery
# STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F3-DISCOVERY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACCELEROMETER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STM32F3-DISCOVERY</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Modules

<table>
<thead>
<tr>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private Types</td>
</tr>
<tr>
<td>Private Constants</td>
</tr>
<tr>
<td>Private Macros</td>
</tr>
<tr>
<td>Private Variables</td>
</tr>
<tr>
<td>Exported Types</td>
</tr>
<tr>
<td>Exported Constants</td>
</tr>
<tr>
<td>Exported Macros</td>
</tr>
<tr>
<td>Exported Functions</td>
</tr>
<tr>
<td>Private Functions</td>
</tr>
</tbody>
</table>

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by **doxygen** 1.7.6.1
# STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modules</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**STM32F3-DISCOVERY**

**GYROSCOPE**

STM32F3-DISCOVERY
Modules

<table>
<thead>
<tr>
<th>Private Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private Constants</td>
</tr>
<tr>
<td>Private Macros</td>
</tr>
<tr>
<td>Private Variables</td>
</tr>
<tr>
<td>Private Functions</td>
</tr>
<tr>
<td>Exported Types</td>
</tr>
<tr>
<td>Exported Constants</td>
</tr>
<tr>
<td>Exported Macros</td>
</tr>
<tr>
<td>Exported Functions</td>
</tr>
</tbody>
</table>

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by doxygen 1.7.6.1
# STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
<th>Defines</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Private Constants</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

STM32F3-DISCOVERY Common
# Defines

<table>
<thead>
<tr>
<th>Define</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__STM32F3_DISCO_BSP_VERSION_MAIN</code></td>
<td>0x02</td>
</tr>
<tr>
<td>STM32F3 DISCOVERY BSP Driver version number V2.1.5.</td>
<td></td>
</tr>
<tr>
<td><code>__STM32F3_DISCO_BSP_VERSION_SUB1</code></td>
<td>0x01</td>
</tr>
<tr>
<td><code>__STM32F3_DISCO_BSP_VERSION_SUB2</code></td>
<td>0x05</td>
</tr>
<tr>
<td><code>__STM32F3_DISCO_BSP_VERSION_RC</code></td>
<td>0x00</td>
</tr>
<tr>
<td><code>__STM32F3_DISCO_BSP_VERSION</code></td>
<td></td>
</tr>
</tbody>
</table>
Define Documentation

#define __STM32F3_DISCO_BSP_VERSION

Value:

```
((__STM32F3_DISCO_BSP_VERSION_MAIN << 24) | (__STM32F3_DISCO_BSP_VERSION_SUB1 << 16) | (__STM32F3_DISCO_BSP_VERSION_SUB2 << 8) | (__STM32F3_DISCO_BSP_VERSION_RC))
```

Definition at line 64 of file stm32f3_discovery.c.

Referenced by BSP_GetVersion().

#define __STM32F3_DISCO_BSP_VERSION_MAIN (0x02)

STM32F3 DISCOVERY BSP Driver version number V2.1.5.

[31:24] main version

Definition at line 60 of file stm32f3_discovery.c.

#define __STM32F3_DISCO_BSP_VERSION_RC (0x00)

[7:0] release candidate

Definition at line 63 of file stm32f3_discovery.c.

#define __STM32F3_DISCO_BSP_VERSION_SUB1 (0x01)
[23:16] sub1 version
Definition at line 61 of file stm32f3_discovery.c.

#define __STM32F3_DISCO_BSP_VERSION_SUB2 (0x05)

[15:8] sub2 version
Definition at line 62 of file stm32f3_discovery.c.

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP
User Manual by doxygen 1.7.6.1
# STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
<th>Defines</th>
</tr>
</thead>
</table>

## STM32F3-DISCOVERY COMPONENT

Exported Constants
**Defines**

<table>
<thead>
<tr>
<th>Preprocessor Directive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#define READWRITE_CMD</code></td>
<td><code>((uint8_t)0x80)</code></td>
</tr>
<tr>
<td><code>#define MULTIPLEBYTE_CMD</code></td>
<td><code>((uint8_t)0x40)</code></td>
</tr>
<tr>
<td><code>#define DUMMY_BYTE</code></td>
<td><code>((uint8_t)0x00)</code></td>
</tr>
<tr>
<td><code>#define GYRO_CS_LOW()</code></td>
<td><code>HAL_GPIO_WritePin(GYRO_CS_GPIO_PORT, GPIO_PIN_RESET)</code></td>
</tr>
<tr>
<td><code>#define GYRO_CS_HIGH()</code></td>
<td><code>HAL_GPIO_WritePin(GYRO_CS_GPIO_PORT, GPIO_PIN_SET)</code></td>
</tr>
<tr>
<td><code>#define GYRO_CS_GPIO_PORT</code></td>
<td><code>GPIOE /* GPIOE */</code></td>
</tr>
<tr>
<td><code>#define GYRO_CS_GPIO_CLK_ENABLE()</code></td>
<td><code>__HAL_RCC_GPIOE_CLK_ENABLE()</code></td>
</tr>
<tr>
<td><code>#define GYRO_CS_GPIO_CLK_DISABLE()</code></td>
<td><code>__HAL_RCC_GPIOE_CLK_DISABLE()</code></td>
</tr>
<tr>
<td><code>#define GYRO_CS_PIN</code></td>
<td><code>GPIO_PIN_3 /* PE.03 */</code></td>
</tr>
<tr>
<td><code>#define GYRO_INT_GPIO_PORT</code></td>
<td><code>GPIOE /* GPIOE */</code></td>
</tr>
<tr>
<td><code>#define GYRO_INT_GPIO_CLK_ENABLE()</code></td>
<td><code>__HAL_RCC_GPIOE_CLK_ENABLE()</code></td>
</tr>
<tr>
<td><code>#define GYRO_INT_GPIO_CLK_DISABLE()</code></td>
<td><code>__HAL_RCC_GPIOE_CLK_DISABLE()</code></td>
</tr>
<tr>
<td><code>#define GYRO_INT1_PIN</code></td>
<td><code>GPIO_PIN_0 /* PE.00 */</code></td>
</tr>
<tr>
<td><code>#define GYRO_INT1_EXTI_IRQn</code></td>
<td><code>EXTI0_IRQn</code></td>
</tr>
<tr>
<td><code>#define GYRO_INT2_PIN</code></td>
<td><code>GPIO_PIN_1 /* PE.01 */</code></td>
</tr>
<tr>
<td><code>#define GYRO_INT2_EXTI_IRQn</code></td>
<td><code>EXTI1_IRQn</code></td>
</tr>
<tr>
<td><code>#define ACCELEROMETER_I2C_ADDRESS</code></td>
<td><code>0x32</code></td>
</tr>
<tr>
<td><code>#define ACCELERO_DRDY_PIN</code></td>
<td><code>GPIO_PIN_2 /* PE.02 */</code></td>
</tr>
<tr>
<td><code>#define ACCELERO_DRDY_GPIO_PORT</code></td>
<td><code>GPIOE /* GPIOE */</code></td>
</tr>
<tr>
<td><code>#define ACCELERO_DRDY_GPIO_CLK_ENABLE()</code></td>
<td><code>__HAL_RCC_GPIOE_CLK_ENABLE()</code></td>
</tr>
<tr>
<td><code>#define ACCELERO_DRDY_GPIO_CLK_DISABLE()</code></td>
<td><code>__HAL_RCC_GPIOE_CLK_DISABLE()</code></td>
</tr>
<tr>
<td><code>#define ACCELERO_DRDY_EXTI_IRQn</code></td>
<td><code>EXTI2_TSC_IRQn /* TAMP_STAMP_IRQn */</code></td>
</tr>
<tr>
<td><code>#define ACCELEROMETER_INT_GPIO_PORT</code></td>
<td><code>GPIOE /* GPIOE */</code></td>
</tr>
<tr>
<td><code>#define ACCELEROMETER_INT_GPIO_CLK_ENABLE()</code></td>
<td><code>__HAL_RCC_GPIOE_CLK_ENABLE()</code></td>
</tr>
<tr>
<td><code>#define ACCELEROMETER_INT_GPIO_CLK_DISABLE()</code></td>
<td><code>__HAL_RCC_GPIOE_CLK_DISABLE()</code></td>
</tr>
<tr>
<td><code>#define ACCELEROMETER_INT1_PIN</code></td>
<td><code>GPIO_PIN_4 /* PE.04 */</code></td>
</tr>
<tr>
<td><code>#define ACCELEROMETER_INT1_EXTI_IRQn</code></td>
<td><code>EXTI4_IRQn</code></td>
</tr>
</tbody>
</table>
#define ACCELERO_INT2_PIN GPIO_PIN_5 /* PE.05 */
#define ACCELERO_INT2_EXTI_IRQn EXTI9_5_IRQHandler
Define Documentation

#define ACCELERO_DRDY_EXTI_IRQn EXIT2_TSC_IRQn /*TAMP_*

Definition at line 320 of file stm32f3_discovery.h.
Referenced by COMPASSACCELERO_IO_Init().

#define ACCELERO_DRDY_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOE_CLK_DISABLE()

Definition at line 319 of file stm32f3_discovery.h.

#define ACCELERO_DRDY_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOE_CLK_ENABLE()

Definition at line 318 of file stm32f3_discovery.h.
Referenced by COMPASSACCELERO_IO_Init().

#define ACCELERO_DRDY_GPIO_PORT GPIOE /* GPIOE */

Definition at line 317 of file stm32f3_discovery.h.
Referenced by COMPASSACCELERO_IO_Init().

#define ACCELERO_DRDY_PIN GPIO_PIN_2 /* PE.02 */

Definition at line 316 of file stm32f3_discovery.h.
Referenced by COMPASSACCELERO_IO_Init().

#define ACCELERO_I2C_ADDRESS 0x32
ACCELEROMETER I2C1 Interface pins.

Definition at line 314 of file stm32f3_discovery.h.

Referenced by I2Cx_Init().

#define ACCELERO_INT1_EXTIIRQn EXTI4_IRQn

Definition at line 326 of file stm32f3_discovery.h.

Referenced by COMPASSACCELERO_IO_ITConfig().

#define ACCELERO_INT1_PIN GPIO_PIN_4 /* PE.04 */

Definition at line 325 of file stm32f3_discovery.h.

Referenced by COMPASSACCELERO_IO_Init(), and COMPASSACCELERO_IO_ITConfig().

#define ACCELERO_INT2_EXTIIRQn EXTI9_5_IRQn

Definition at line 328 of file stm32f3_discovery.h.

#define ACCELERO_INT2_PIN GPIO_PIN_5 /* PE.05 */

Definition at line 327 of file stm32f3_discovery.h.

Referenced by COMPASSACCELERO_IO_Init(), and COMPASSACCELERO_IO_ITConfig().

#define ACCELERO_INT_GPIO_CLK_DISABLE () __HAL_RCC_GPIOE_CLK_DISABLE()
#define ACCELERO_INT_GPIO_CLK_ENABLE () __HAL_RCC_GPIOE_CLK_ENABLE()

Definition at line 324 of file stm32f3_discovery.h.

Referenced by COMPASSACCELERO_IO_Init(), and COMPASSACCELERO_IO_ITConfig().

#define ACCELERO_INT_GPIO_PORT GPIOE /* GPIOE */

Definition at line 323 of file stm32f3_discovery.h.

Referenced by COMPASSACCELERO_IO_Init(), and COMPASSACCELERO_IO_ITConfig().

#define DUMMY_BYTE ((uint8_t)0x00)

Definition at line 288 of file stm32f3_discovery.h.

Referenced by GYRO_IO_Read().

#define GYRO_CS_GPIO_CLK_DISABLE () __HAL_RCC_GPIOE

Definition at line 299 of file stm32f3_discovery.h.

#define GYRO_CS_GPIO_CLK_ENABLE () __HAL_RCC_GPIOE

Definition at line 298 of file stm32f3_discovery.h.

Referenced by GYRO_IO_Init().
#define GYRO_CS_GPIO_PORT GPIOE /* GPIOE */

GYRO SPI Interface pins.

Definition at line 297 of file stm32f3_discovery.h.

Referenced by GYRO_IO_Init().

#define GYRO_CS_HIGH () HAL_GPIO_WritePin(GYRO_CS_GPIO_PORT

Definition at line 292 of file stm32f3_discovery.h.

Referenced by GYRO_IO_Init(), GYRO_IO_Read(), and GYRO_IO_Write().

#define GYRO_CS_LOW () HAL_GPIO_WritePin(GYRO_CS_GPIO_PORT

Definition at line 291 of file stm32f3_discovery.h.

Referenced by GYRO_IO_Read(), and GYRO_IO_Write().

#define GYRO_CS_PIN GPIO_PIN_3 /* PE.03 */

Definition at line 300 of file stm32f3_discovery.h.

Referenced by GYRO_IO_Init().

#define GYRO_INT1_EXTI_IRQn EXTI0_IRQn

Definition at line 306 of file stm32f3_discovery.h.


```c
#define GYRO_INT1_PIN GPIO_PIN_0 /* PE.00 */

Definition at line 305 of file stm32f3_discovery.h.
Referenced by GYRO_IO_Init().
```

```c
#define GYRO_INT2_EXTI_IRQn EXTI1_IRQn

Definition at line 308 of file stm32f3_discovery.h.
```

```c
#define GYRO_INT2_PIN GPIO_PIN_1 /* PE.01 */

Definition at line 307 of file stm32f3_discovery.h.
Referenced by GYRO_IO_Init().
```

```c
#define GYRO_INT_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()

Definition at line 304 of file stm32f3_discovery.h.
```

```c
#define GYRO_INT_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()

Definition at line 303 of file stm32f3_discovery.h.
Referenced by GYRO_IO_Init().
```

```c
#define GYRO_INT_GPIO_PORT GPIOE /* GPIOE */

Definition at line 302 of file stm32f3_discovery.h.
Referenced by GYRO_IO_Init().
```
#define MULTIPLEBYTE_CMD ((uint8_t)0x40)

Definition at line 286 of file stm32f3_discovery.h.

Referenced by GYRO_IO_Read(), and GYRO_IO_Write().

#define READWRITE_CMD ((uint8_t)0x80)

Definition at line 284 of file stm32f3_discovery.h.

Referenced by GYRO_IO_Read().
## STM32F3-Discovery BSP User Manual

### Exported Constants

<table>
<thead>
<tr>
<th>STM32F3-DISCOVERY ACCELEROMETER</th>
<th></th>
</tr>
</thead>
</table>

---

**Enumerations**
## Enumerations

```c
enum ACCELERO_StatusTypeDef { ACCELERO_OK = 0,
                                  ACCELERO_ERROR = 1,
                                  ACCELERO_TIMEOUT = 2 }
```
Enumeration Type Documentation

```
enum ACCELERO_StatusTypeDef

    ACCELERO_OK
    ACCELERO_ERROR
    ACCELERO_TIMEOUT

Definition at line 74 of file stm32f3_discovery_accelerometer.h.
```

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by doxygen 1.7.6.1
## Private Variables

<table>
<thead>
<tr>
<th>STM32F3-DISCOVERY ACCELEROMETER</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Variables</th>
</tr>
</thead>
</table>

STM32F3-Discovery BSP User Manual
### Variables

| static ACCELERO_DrvTypeDef * AccelerometerDrv |
Variable Documentation

**ACCELERO_DrvTypeDef* AccelerometerDrv [static]**

Definition at line 80 of file `stm32f3_discovery_accelerometer.c`.

Referenced by `BSP_ACCELERO_GetXYZ()`, `BSP_ACCELERO_Init()`, and `BSP_ACCELERO_Reset()`.

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by **doxygen 1.7.6.1**
# STM32F3-Discovery BSP User Manual

## Exported Functions

<table>
<thead>
<tr>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F3-DISCOVERY ACCELEROMETER</td>
</tr>
</tbody>
</table>
## Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uint8_t</code></td>
<td><code>BSP_ACCELERO_Init</code> (void)</td>
<td></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_ACCELERO_Reset</code> (void)</td>
<td>Reboot memory content of ACCELEROMETER.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_ACCELERO_GetXYZ</code> (int16_t *pDataXYZ)</td>
<td>Get XYZ acceleration.</td>
</tr>
</tbody>
</table>
void BSP_ACCELERO_GetXYZ ( int16_t * pDataXYZ )

Get XYZ acceleration.

Parameters:


Return values:

None

Definition at line 174 of file stm32f3_discovery_accelerometer.c.

References AccelerometerDrv.

uint8_t BSP_ACCELERO_Init ( void )

Definition at line 99 of file stm32f3_discovery_accelerometer.c.

References ACCELERO_ERROR, ACCELERO_OK, and AccelerometerDrv.

void BSP_ACCELERO_Reset ( void )

Reboot memory content of ACCELEROMETER.

Return values:

None

Definition at line 160 of file stm32f3_discovery_accelerometer.c.
References AccelerometerDrv.

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by doxygen 1.7.6.1
# STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
<th>Functions</th>
</tr>
</thead>
</table>

## Exported Functions

**STM32F3-DISCOVERY Common**
## Functions

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Function Name</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uint32_t</code></td>
<td><code>BSP_GetVersion</code> (void)</td>
<td></td>
<td>This method returns the STM32F3-DISCOVERY BSP Driver revision.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_LED_Init</code> (Led_TypeDef Led)</td>
<td></td>
<td>Configures LED GPIO.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_LED_On</code> (Led_TypeDef Led)</td>
<td></td>
<td>Turns selected LED On.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_LED_Off</code> (Led_TypeDef Led)</td>
<td></td>
<td>Turns selected LED Off.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_LED_Toggle</code> (Led_TypeDef Led)</td>
<td></td>
<td>Toggles the selected LED.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_PB_Init</code> (Button_TypeDef Button, ButtonMode_TypeDef ButtonMode)</td>
<td></td>
<td>Configures Push Button GPIO and EXTI Line.</td>
</tr>
<tr>
<td><code>uint32_t</code></td>
<td><code>BSP_PB_GetState</code> (Button_TypeDef Button)</td>
<td></td>
<td>Returns the selected Push Button state.</td>
</tr>
</tbody>
</table>
Function Documentation

uint32_t BSP_GetVersion ( void )

This method returns the STM32F3-DISCOVERY BSP Driver revision.

Return values:
  version : 0xXYZR (8bits for each decimal, R for RC)

Definition at line 155 of file stm32f3_discovery.c.

References __STM32F3_DISCO_BSP_VERSION.

void BSP_LED_Init ( Led_TypeDef  Led )

Configures LED GPIO.

Parameters:
  Led Specifies the Led to be configured. This parameter can be one of following parameters:
    • LED_RED
    • LED_BLUE
    • LED_ORANGE
    • LED_GREEN
    • LED_GREEN2
    • LED_ORANGE2
    • LED_BLUE2
    • LED_RED2

Return values:
  None

Definition at line 174 of file stm32f3_discovery.c.

References LED_PIN, LED_PORT, and
void BSP_LED_Off (Led_TypeDef Led)

Turns selected LED Off.

Parameters:

   Led Specifies the Led to be set off. This parameter can be one of following parameters:
   
   - LED_RED
   - LED_BLUE
   - LED_ORANGE
   - LED_GREEN
   - LED_GREEN2
   - LED_ORANGE2
   - LED_BLUE2
   - LED_RED2

Return values:

   None

Definition at line 225 of file stm32f3_discovery.c.

References LED_PIN, and LED_PORT.

void BSP_LED_On (Led_TypeDef Led)

Turns selected LED On.

Parameters:

   Led Specifies the Led to be set on. This parameter can be one of following parameters:
   
   - LED_RED
   - LED4
   - LED5
- LED6
- LED7
- LED8
- LED9
- LED10

**Return values:**

None

Definition at line 206 of file `stm32f3_discovery.c`.

References `LED_PIN`, and `LED_PORT`.

```c
void BSP_LED_Toggle ( Led_TypeDef Led )
```

Toggles the selected LED.

**Parameters:**

*`Led`* Specifies the Led to be toggled. This parameter can be one of following parameters:

- LED_RED
- LED_BLUE
- LED_ORANGE
- LED_GREEN
- LED_GREEN2
- LED_ORANGE2
- LED_BLUE2
- LED_RED2

**Return values:**

None

Definition at line 244 of file `stm32f3_discovery.c`.

References `LED_PIN`, and `LED_PORT`.  ```
uint32_t BSP_PB_GetState ( Button_TypeDef Button )

Returns the selected Push Button state.

Parameters:
  Button Specifies the Button to be checked. This parameter should be: BUTTON_USER

Return values:
  The Button GPIO pin value.

Definition at line 300 of file stm32f3_discovery.c.

References BUTTON_PIN, and BUTTON_PORT.

void BSP_PB_Init ( Button_TypeDef Button, 
                  ButtonMode_TypeDef ButtonMode )

Configures Push Button GPIO and EXTI Line.

Parameters:
  Button Specifies the Button to be configured. This parameter should be: BUTTON_USER
  ButtonMode Specifies Button mode. This parameter can be one of following parameters:
    • BUTTON_MODE_GPIO: Button will be used as simple IO
    • BUTTON_MODE_EXTI: Button will be connected to EXTI line with interrupt generation capability

Return values:
  None
Definition at line 261 of file stm32f3_discovery.c.

References BUTTON_IRQn, BUTTON_MODE_EXTI, BUTTON_MODE_GPIO, BUTTON_PIN, BUTTON_PORT, and BUTTONx_GPIO_CLK_ENABLE.
## STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
<th>Functions</th>
</tr>
</thead>
</table>

### Exported Functions

**STM32F3-DISCOVERY GYROSCOPE**
## Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t BSP_GYRO_Init (void)</td>
<td>Set GYROSCOPE Initialization.</td>
</tr>
<tr>
<td>uint8_t BSP_GYRO_ReadID (void)</td>
<td>Read ID of Gyroscope component.</td>
</tr>
<tr>
<td>void BSP_GYRO_Reset (void)</td>
<td>Reboot memory content of GYROSCOPE.</td>
</tr>
<tr>
<td>void BSP_GYRO_ITConfig (GYRO_InterruptConfigTypeDef *pIntConfig)</td>
<td>Configure INT1 interrupt.</td>
</tr>
<tr>
<td>void BSP_GYRO_EnableIT (uint8_t IntPin)</td>
<td>Enable INT1 or INT2 interrupt.</td>
</tr>
<tr>
<td>void BSP_GYRO_DisableIT (uint8_t IntPin)</td>
<td>Disable INT1 or INT2 interrupt.</td>
</tr>
<tr>
<td>void BSP_GYRO_GetXYZ (float *pfData)</td>
<td>Get XYZ angular acceleration.</td>
</tr>
</tbody>
</table>
Function Documentation

void BSP_GYRO_DisableIT ( uint8_t IntPin )

Disable INT1 or INT2 interrupt.

**Parameters:**

- **IntPin** Interrupt pin This parameter can be:
  - L3GD20_INT1
  - L3GD20_INT2

**Return values:**

- None

Definition at line 227 of file stm32f3_discovery_gyroscope.c.

References GyroscopeDrv.

void BSP_GYRO_EnableIT ( uint8_t IntPin )

Enable INT1 or INT2 interrupt.

**Parameters:**

- **IntPin** Interrupt pin This parameter can be:
  - L3GD20_INT1
  - L3GD20_INT2

**Return values:**

- None

Definition at line 211 of file stm32f3_discovery_gyroscope.c.

References GyroscopeDrv.
void BSP_GYRO_GetXYZ ( float * pfData )

Get XYZ angular acceleration.

Parameters:
  pfData pointer on floating array

Return values:
  None

Definition at line 240 of file stm32f3_discovery_gyroscope.c.
References GyroscopeDrv.

uint8_t BSP_GYRO_Init ( void )

Set GYROSCOPE Initialization.

Return values:
  GYRO_OK if no problem during initialization

Definition at line 102 of file stm32f3_discovery_gyroscope.c.
References GYRO_ERROR, GYRO_OK, and GyroscopeDrv.

void BSP_GYRO_ITConfig ( GYRO_InterruptConfigTypeDef * pIntC

Configure INT1 interrupt.

Parameters:
  pIntConfig pointer to a L3GD20_InterruptConfig_TypeDef structure that contains the configuration setting for the L3GD20 Interrupt.

Return values:
uint8_t BSP_GYRO_ReadID ( void )

Read ID of Gyroscope component.

**Return values:**

ID

References *GyroscopeDrv*.

---

void BSP_GYRO_Reset ( void )

Reboot memory content of GYROSCOPE.

**Return values:**

None

Definition at line 172 of file *stm32f3_discovery_gyroscope.c*.

References *GyroscopeDrv*.
# STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Private Variables</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>STM32F3-DISCOVERY Common</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GPIO_TypeDef *</code></td>
<td><code>LED_PORT [LEDn]</code></td>
</tr>
<tr>
<td><code>const uint16_t</code></td>
<td><code>LED_PIN [LEDn]</code></td>
</tr>
<tr>
<td><code>GPIO_TypeDef *</code></td>
<td><code>BUTTON_PORT [BUTTONn] = {USER_BUTTON_GPIO_PORT}</code></td>
</tr>
<tr>
<td><code>const uint16_t</code></td>
<td><code>BUTTON_PIN [BUTTONn] = {USER_BUTTON_PIN}</code></td>
</tr>
<tr>
<td><code>const uint8_t</code></td>
<td><code>BUTTON_IRQn [BUTTONn] = {USER_BUTTON_EXTI_IRQn}</code></td>
</tr>
<tr>
<td><code>uint32_t</code></td>
<td><code>SpixTimeout = SPIx_TIMEOUT_MAX</code></td>
</tr>
<tr>
<td><code>static SPI_HandleTypeDef</code></td>
<td><code>SpiHandle</code></td>
</tr>
<tr>
<td><code>static I2C_HandleTypeDef</code></td>
<td><code>I2cHandle</code></td>
</tr>
<tr>
<td><code>uint32_t</code></td>
<td><code>I2cxTimeout = I2Cx_TIMEOUT_MAX</code></td>
</tr>
</tbody>
</table>
Variable Documentation

const uint8_t BUTTON_IRQn[BUTTONn] = {USER_BUTTON EXTI_I

Definition at line 89 of file stm32f3_discovery.c.
Referenced by BSP_PB_Init().

const uint16_t BUTTON_PIN[BUTTONn] = {USER_BUTTON_PIN}

Definition at line 88 of file stm32f3_discovery.c.
Referenced by BSP_PB_GetState(), and BSP_PB_Init().

GPIO_TypeDef* BUTTON_PORT[BUTTONn] = {USER_BUTTON_GPIO_PORT

BUTTON variables.
Definition at line 87 of file stm32f3_discovery.c.
Referenced by BSP_PB_GetState(), and BSP_PB_Init().

I2C_HandleTypeDef I2cHandle [static]

Definition at line 100 of file stm32f3_discovery.c.
Referenced by I2Cx_Error(), I2Cx_Init(), I2Cx_ReadData(), and I2Cx_WriteData().

uint32_t I2cxTimeout = I2Cx_TIMEOUT_MAX

Definition at line 101 of file stm32f3_discovery.c.
Referenced by \texttt{I2Cx\_ReadData()}, and \texttt{I2Cx\_WriteData()}.

\begin{verbatim}
const uint16_t LED\_PIN[LEDn]

Initial value:
\{LED3\_PIN, LED4\_PIN, LED5\_PIN, LED6\_PIN, LED7\_PIN, LED8\_PIN, LED9\_PIN, LED10\_PIN\}
\end{verbatim}

Definition at line 81 of file \texttt{stm32f3\_discovery.c}.

Referenced by \texttt{BSP\_LED\_Init()}, \texttt{BSP\_LED\_Off()}, \texttt{BSP\_LED\_On()}, and \texttt{BSP\_LED\_Toggle()}.

\begin{verbatim}
GPIO\_TypeDef* LED\_PORT[LEDn]

Initial value:
\{LED3\_GPIO\_PORT, LED4\_GPIO\_PORT, LED5\_GPIO\_PORT, LED6\_GPIO\_PORT, LED7\_GPIO\_PORT, LED8\_GPIO\_PORT, LED9\_GPIO\_PORT, LED10\_GPIO\_PORT\}
\end{verbatim}

LED variables.

Definition at line 78 of file \texttt{stm32f3\_discovery.c}.

Referenced by \texttt{BSP\_LED\_Init()}, \texttt{BSP\_LED\_Off()}, \texttt{BSP\_LED\_On()}, and \texttt{BSP\_LED\_Toggle()}.

\begin{verbatim}
SPI\_HandleTypeDef SpiHandle [static]

Definition at line 96 of file \texttt{stm32f3\_discovery.c}.

Referenced by \texttt{SPIx\_Error()}, \texttt{SPIx\_Init()}, and \texttt{SPIx\_WriteRead()}.\end{verbatim}
```c
uint32_t SpixTimeout = SPIx_TIMEOUT_MAX
```

BUS variables.

Definition at line 95 of file `stm32f3_discovery.c`.

Referenced by `SPIx_WriteRead()`.
# STM32F3-Discovery BSP User Manual

## Exported Types

<table>
<thead>
<tr>
<th>STM32F3-DISCOVERY Common</th>
<th>Asynchronous</th>
<th>Buffering</th>
<th>Core</th>
<th>Clock</th>
<th>Section</th>
<th>Stream</th>
<th>System</th>
<th>Timer</th>
<th>UART</th>
<th>USB</th>
<th>Wire</th>
<th>Window</th>
</tr>
</thead>
</table>

**Enumerations**
## Enumerations

<table>
<thead>
<tr>
<th>Enum Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Led_TypeDef</td>
<td><code>enum Led_TypeDef { LED3 = 0, LED4 = 1, LED5 = 2, LED6 = 3, LED7 = 4, LED8 = 5, LED9 = 6, LED10 = 7, LED_GREEN = LED6, LED_ORANGE = LED5, LED_RED = LED3, LED_BLUE = LED4, LED_GREEN_2 = LED7, LED_ORANGE_2 = LED8, LED_RED_2 = LED10, LED_BLUE_2 = LED9 }</code>&lt;br&gt;LED Types Definition. More...</td>
</tr>
<tr>
<td>Button_TypeDef</td>
<td><code>enum Button_TypeDef { BUTTON_USER = 0 }</code>&lt;br&gt;BUTTON Types Definition. More...</td>
</tr>
<tr>
<td>ButtonMode_TypeDef</td>
<td><code>enum ButtonMode_TypeDef { BUTTON_MODE_GPIO = 0, BUTTON_MODE_EXTI = 1 }</code></td>
</tr>
</tbody>
</table>
Enumeration Type Documentation

enum Button_TypeDef

BUTTON Types Definition.

**Enumerator:**

`BUTTON_USER`

Definition at line 105 of file `stm32f3_discovery.h`.

enum ButtonMode_TypeDef

**Enumerator:**

`BUTTON_MODE_GPIO`
`BUTTON_MODE_EXTI`

Definition at line 111 of file `stm32f3_discovery.h`.

enum Led_TypeDef

LED Types Definition.

**Enumerator:**

`LED3`
`LED4`
`LED5`
`LED6`
`LED7`
`LED8`
`LED9`
`LED10`
LED_GREEN
LED_ORANGE
LED_RED
LED_BLUE
LED_GREEN_2
LED_ORANGE_2
LED_RED_2
LED_BLUE_2

Definition at line 81 of file stm32f3_discovery.h.
<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
<th>Defines</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F3-DISCOVERY</td>
<td>BUTTON</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Exported Constants
<table>
<thead>
<tr>
<th>Define</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#define BUTTONn 1</td>
<td></td>
</tr>
<tr>
<td>#define USER_BUTTON_PIN GPIO_PIN_0</td>
<td></td>
</tr>
<tr>
<td>push-button</td>
<td></td>
</tr>
<tr>
<td>#define USER_BUTTON_GPIO_PORT GPIOA</td>
<td></td>
</tr>
<tr>
<td>#define USER_BUTTON_GPIO_CLK_ENABLE( ) __HAL_RCC_GPIOA_CLK_ENABLE()</td>
<td></td>
</tr>
<tr>
<td>#define USER_BUTTON_GPIO_CLK_DISABLE( ) __HAL_RCC_GPIOA_CLK_DISABLE()</td>
<td></td>
</tr>
<tr>
<td>#define USER_BUTTON_EXTI_IRQn EXTI0_IRQn</td>
<td></td>
</tr>
<tr>
<td>#define BUTTONx_GPIO_CLK_ENABLE(<strong>BUTTON</strong>) do { if ((<strong>BUTTON</strong>) == BUTTON_USER) USER_BUTTON_GPIO_CLK_ENABLE();} while(0)</td>
<td></td>
</tr>
<tr>
<td>#define BUTTONx_GPIO_CLK_DISABLE(<strong>BUTTON</strong>) (((<strong>BUTTON</strong>) == BUTTON_USER) ? USER_BUTTON_GPIO_CLK_DISABLE() : 0)</td>
<td></td>
</tr>
</tbody>
</table>
Define Documentation

#define BUTTONn  1

Definition at line 206 of file stm32f3_discovery.h.

#define BUTTONx_GPIO_CLK_DISABLE (__BUTTON__) (((__BUTTON__) ==

Definition at line 219 of file stm32f3_discovery.h.

Referenced by BSP_PB_Init().

#define BUTTONx_GPIO_CLK_ENABLE (__BUTTON__) do { if (!

Definition at line 217 of file stm32f3_discovery.h.

#define USER_BUTTON_EXTI_IRQn  EXTI0_IRQn

Definition at line 215 of file stm32f3_discovery.h.

#define USER_BUTTON_GPIO_CLK_DISABLE () __HAL_RCC_GPIOA_CLK_DISABLE()

Definition at line 214 of file stm32f3_discovery.h.

#define USER_BUTTON_GPIO_CLK_ENABLE () __HAL_RCC_GPIOA_CLK_ENABLE()

Definition at line 213 of file stm32f3_discovery.h.

#define USER_BUTTON_GPIO_PORT GPIOA
Definition at line 212 of file `stm32f3_discovery.h`.

```c
#define USER_BUTTON_PIN GPIO_PIN_0
```

push-button

Definition at line 211 of file `stm32f3_discovery.h`.

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP
User Manual by doxygen 1.7.6.1
## Link Operation functions

STM32F3-DISCOVERY Common
## Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <strong>GYRO_IO_Init</strong> (void)</td>
<td>Configures the GYROSCOPE SPI interface.</td>
</tr>
<tr>
<td>void <strong>GYRO_IO_Write</strong> (uint8_t *pBuffer, uint8_t WriteAddr, uint16_t NumByteToWrite)</td>
<td>Writes one byte to the GYROSCOPE.</td>
</tr>
<tr>
<td>void <strong>GYRO_IO_Read</strong> (uint8_t *pBuffer, uint8_t ReadAddr, uint16_t NumByteToRead)</td>
<td>Reads a block of data from the GYROSCOPE.</td>
</tr>
<tr>
<td>void <strong>COMPASSACCELERO_IO_Init</strong> (void)</td>
<td>Configures COMPASS / ACCELEROMETER I2C interface.</td>
</tr>
<tr>
<td>void <strong>COMPASSACCELERO_IO_ITConfig</strong> (void)</td>
<td>Configures COMPASS / ACCELERO click IT.</td>
</tr>
<tr>
<td>void <strong>COMPASSACCELERO_IO_Write</strong> (uint16_t DeviceAddr, uint8_t RegisterAddr, uint8_t Value)</td>
<td>Writes one byte to the COMPASS / ACCELEROMETER.</td>
</tr>
<tr>
<td>uint8_t <strong>COMPASSACCELERO_IO_Read</strong> (uint16_t DeviceAddr, uint8_t RegisterAddr)</td>
<td>Reads a block of data from the COMPASS / ACCELEROMETER.</td>
</tr>
</tbody>
</table>
Function Documentation

```c
void COMPASSACCELERO_IO_Init ( void )
```

Configures COMPASS / ACCELEROMETER I2C interface.

**Return values:**

None

Definition at line 640 of file `stm32f3_discovery.c`.

References `ACCELERO_DRDY_EXTI_IRQn`, `ACCELERO_DRDY_GPIO_CLK_ENABLE`, `ACCELERO_DRDY_GPIO_PORT`, `ACCELERO_DRDY_PIN`, `ACCELERO_INT1_PIN`, `ACCELERO_INT2_PIN`, `ACCELERO_INT_GPIO_CLK_ENABLE`, `ACCELERO_INT_GPIO_PORT`, and `I2Cx_Init()`.

```c
void COMPASSACCELERO_IO_ITConfig ( void )
```

Configures COMPASS / ACCELERO click IT.

**Return values:**

None

Definition at line 675 of file `stm32f3_discovery.c`.

References `ACCELERO_INT1_EXTI_IRQn`, `ACCELERO_INT1_PIN`, `ACCELERO_INT2_PIN`, `ACCELERO_INT_GPIO_CLK_ENABLE`, and `ACCELERO_INT_GPIO_PORT`.

```c
uint8_t COMPASSACCELERO_IO_Read ( uint16_t DeviceAddr, uint8_t RegisterAddr )
```
Reads a block of data from the COMPASS / ACCELEROMETER.

**Parameters:**
- **DeviceAddr** specifies the slave address to be programmed (ACC_I2C_ADDRESS or MAG_I2C_ADDRESS).
- **RegisterAddr** specifies the COMPASS / ACCELEROMETER internal address register to read from.

**Return values:**
- **ACCELEROMETER** register value

Definition at line 714 of file `stm32f3_discovery.c`.

References `I2Cx_ReadData()`.

```c
void COMPASSACCELERO_IO_Write ( uint16_t DeviceAddr,
                                  uint8_t RegisterAddr,
                                  uint8_t Value )
```

Writes one byte to the COMPASS / ACCELEROMETER.

**Parameters:**
- **DeviceAddr** specifies the slave address to be programmed.
- **RegisterAddr** specifies the COMPASS / ACCELEROMETER register to be written.
- **Value** Data to be written

**Return values:**
- **None**

Definition at line 702 of file `stm32f3_discovery.c`. 
References I2Cx_WriteData().

```c
void GYRO_IO_Init ( void )
```

Configures the GYROSCOPE SPI interface.

**Return values:**

None

Definition at line 535 of file stm32f3_discovery.c.

References GYRO_CS_GPIO_CLK_ENABLE, GYRO_CS_GPIO_PORT, GYRO_CS_HIGH, GYRO_CS_PIN, GYRO_INT1_PIN, GYRO_INT2_PIN, GYRO_INT_GPIO_CLK_ENABLE, GYRO_INT_GPIO_PORT, and SPIx_Init().

```c
void GYRO_IO_Read ( uint8_t * pBuffer,
                    uint8_t ReadAddr,
                    uint16_t NumByteToRead
               )
```

Reads a block of data from the GYROSCOPE.

**Parameters:**

- **pBuffer**
  
  pointer to the buffer that receives the data read from the GYROSCOPE.

- **ReadAddr**
  
  GYROSCOPE's internal address to read from.

- **NumByteToRead**
  
  number of bytes to read from the GYROSCOPE.

**Return values:**

None
Definition at line 604 of file stm32f3_discovery.c.

References DUMMY_BYTE, GYRO_CS_HIGH, GYRO_CS_LOW, MULTIPLEBYTE_CMD, READWRITE_CMD, and SPIx_WriteRead().

```c
void GYRO_IO_Write ( uint8_t * pBuffer,
                      uint8_t WriteAddr,
                      uint16_t NumByteToWrite
                   )
```

Writes one byte to the GYROSCOPE.

**Parameters:**

- **pBuffer** pointer to the buffer containing the data to be written to the GYROSCOPE.
- **WriteAddr** GYROSCOPE's internal address to write to.
- **NumByteToWrite** Number of bytes to write.

**Return values:**

- **None**

Definition at line 569 of file stm32f3_discovery.c.

References GYRO_CS_HIGH, GYRO_CS_LOW, MULTIPLEBYTE_CMD, and SPIx_WriteRead().
STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**STM32F3-DISCOVERY COM**

**Exported Constants**

**Defines**
#defines

**DISCOVERY_SPIx** SPI1
Definition for SPI Interface pins (SPI1 used)

#define DISCOVERY_SPIx_CLK_ENABLE() __HAL_RCC_SPI1_CLK_ENABLE()
define DISCOVERY_SPIx_GPIO_PORT GPIOA /* GPIOA */
define DISCOVERY_SPIx_AF GPIO_AF5_SPI1
#define DISCOVERY_SPIx_GPIO_CLK_ENABLE() __HAL_RCC_GPIOA_CLK_ENABLE()
define DISCOVERY_SPIx_GPIO_CLK_DISABLE() __HAL_RCC_GPIOA_CLK_DISABLE()
define DISCOVERY_SPIx_SCK_PIN GPIO_PIN_5 /* PA.05 */
define DISCOVERY_SPIx_MISO_PIN GPIO_PIN_6 /* PA.06 */
define DISCOVERY_SPIx_MOSI_PIN GPIO_PIN_7 /* PA.07 */
define SPIx_TIMEOUT_MAX ((uint32_t)0x1000)
define DISCOVERY_I2Cx I2C1
Definition for I2C Interface pins (I2C1 used)

define DISCOVERY_I2Cx_CLK_ENABLE() __HAL_RCC_I2C1_CLK_ENABLE()
define DISCOVERY_I2Cx_CLK_DISABLE() __HAL_RCC_I2C1_CLK_DISABLE()
define DISCOVERY_I2Cx_FORCE_RESET() __HAL_RCC_I2C1_FORCE_RESET()
define DISCOVERY_I2Cx_RELEASE_RESET() __HAL_RCC_I2C1_RELEASE_RESET()
define DISCOVERY_I2Cx_SCL_PIN GPIO_PIN_6 /* PB.06 */
define DISCOVERY_I2Cx_SDA_PIN GPIO_PIN_7 /* PB.07 */
define DISCOVERY_I2Cx_GPIO_PORT GPIOB /* GPIOB */
define DISCOVERY_I2Cx_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_ENABLE()
define DISCOVERY_I2Cx_GPIO_CLK_DISABLE() __HAL_RCC_GPIOB_CLK_DISABLE()
define DISCOVERY_I2Cx_AF GPIO_AF4_I2C1
#define I2Cx_TIMEOUT_MAX 0x10000
## Define Documentation

```c
#define DISCOVERY_I2Cx I2C1
```

Definition for I2C Interface pins (I2C1 used)
Definition at line 253 of file `stm32f3_discovery.h`.
Referenced by `I2Cx_Init()`.

```c
#define DISCOVERY_I2Cx_AF GPIO_AF4_I2C1
```

Definition at line 265 of file `stm32f3_discovery.h`.
Referenced by `I2Cx_MspInit()`.

```c
#define DISCOVERY_I2Cx_CLK_DISABLE () __HAL_RCC_I2C1_CLK_DISABLE()
```

Definition at line 255 of file `stm32f3_discovery.h`.

```c
#define DISCOVERY_I2Cx_CLK_ENABLE () __HAL_RCC_I2C1_CLK_ENABLE()
```

Definition at line 254 of file `stm32f3_discovery.h`.
Referenced by `I2Cx_MspInit()`.

```c
#define DISCOVERY_I2Cx_FORCE_RESET () __HAL_RCC_I2C1_FORCE_RESET()
```

Definition at line 256 of file `stm32f3_discovery.h`.

```c
#define DISCOVERY_I2Cx_GPIO_CLK_DISABLE () __HAL_RCC_GPIOB_CLK_DISABLE()
```
Definition at line 264 of file stm32f3_discovery.h.

```
#define DISCOVERY_I2Cx_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOB_CLK_ENABLE()
```
Definition at line 263 of file stm32f3_discovery.h.
Referenced by I2Cx_MspInit().

```
#define DISCOVERY_I2Cx_GPIO_PORT  GPIOB /* GPIOB */
```
Definition at line 262 of file stm32f3_discovery.h.
Referenced by I2Cx_MspInit().

```
#define DISCOVERY_I2Cx_RELEASE_RESET ( ) __HAL_RCC_I2C1_RELEASE_RESET()
```
Definition at line 257 of file stm32f3_discovery.h.

```
#define DISCOVERY_I2Cx_SCL_PIN  GPIO_PIN_6 /* PB.06 */
```
Definition at line 259 of file stm32f3_discovery.h.
Referenced by I2Cx_MspInit().

```
#define DISCOVERY_I2Cx_SDA_PIN  GPIO_PIN_7 /* PB.07 */
```
Definition at line 260 of file stm32f3_discovery.h.
Referenced by I2Cx_MspInit().

```
#define DISCOVERY_SPIx SPI1
```

Definition for SPI Interface pins (SPI1 used)

Definition at line 233 of file stm32f3_discovery.h.

Referenced by SPIx_Init().

#define DISCOVERY_SPIx_AF  GPIO_AF5_SPI1

Definition at line 236 of file stm32f3_discovery.h.

Referenced by SPIx_MspInit().

#define DISCOVERY_SPIx_CLK_ENABLE ( ) __HAL_RCC_SPI1_CLK_ENABLE()

Definition at line 234 of file stm32f3_discovery.h.

Referenced by SPIx_MspInit().

#define DISCOVERY_SPIx_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOA_CLK_DISABLE()

Definition at line 238 of file stm32f3_discovery.h.

Referenced by SPIx_MspInit().

#define DISCOVERY_SPIx_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOA_CLK_ENABLE()

Definition at line 237 of file stm32f3_discovery.h.

Referenced by SPIx_MspInit().

#define DISCOVERY_SPIx_GPIO_PORT GPIOA /* GPIOA */

Definition at line 235 of file stm32f3_discovery.h.

Referenced by SPIx_MspInit().
```c
#define DISCOVERY_SPIx_MISO_PIN GPIO_PIN_6 /* PA.06 */

Definition at line 240 of file stm32f3_discovery.h.
Referenced by SPIx_MspInit().

#define DISCOVERY_SPIx_MOSI_PIN GPIO_PIN_7 /* PA.07 */

Definition at line 241 of file stm32f3_discovery.h.
Referenced by SPIx_MspInit().

#define DISCOVERY_SPIx_SCK_PIN GPIO_PIN_5 /* PA.05 */

Definition at line 239 of file stm32f3_discovery.h.
Referenced by SPIx_MspInit().

#define I2Cx_TIMEOUT_MAX 0x10000

Definition at line 272 of file stm32f3_discovery.h.

#define SPIx_TIMEOUT_MAX ((uint32_t)0x1000)

Definition at line 247 of file stm32f3_discovery.h.
```

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by doxygen 1.7.6.1
## STM32F3-Discovery BSP User Manual

### Exported Types

<table>
<thead>
<tr>
<th>Enumeration Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F3-DISCOVERY GYROSCOPE</td>
</tr>
</tbody>
</table>
Enumerations

```c
enum GYRO_StatusTypeDef { GYRO_OK = 0, GYRO_ERROR = 1, GYRO_TIMEOUT = 2 }
```
Enumeration Type Documentation

```c
enum GYRO_StatusTypeDef
{
    GYRO_OK,
    GYRO_ERROR,
    GYRO_TIMEOUT
};
```

Definition at line 66 of file `stm32f3_discovery_gyroscope.h`.
### STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Private Variables</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

STM32F3-DISCOVERY GYROSCOPE
Variables

| static GYRO_DrvTypeDef * GyroscopeDrv |  |
Variable Documentation

GYRO_DrvTypeDef* GyroscopeDrv [static]

Definition at line 79 of file stm32f3_discovery_gyroscope.c.

Referenced by BSP_GYRO_DisableIT(), BSP_GYRO_EnableIT(), BSP_GYRO_GetXYZ(), BSP_GYRO_Init(), BSP_GYRO_ITConfig(), BSP_GYRO_ReadID(), and BSP_GYRO_Reset().

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by doxygen 1.7.6.1
STM32F3-Discovery BSP User Manual

Bus Operation functions

STM32F3-DISCOVERY Common
## Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td><strong>I2Cx_Init</strong> (void)</td>
<td>Discovery I2Cx Bus initialization.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>I2Cx_WriteData</strong> (uint16_t Addr, uint8_t Reg, uint8_t Value)</td>
<td>Write a value in a register of the device through BUS.</td>
</tr>
<tr>
<td>static uint8_t</td>
<td><strong>I2Cx_ReadData</strong> (uint16_t Addr, uint8_t Reg)</td>
<td>Read a value in a register of the device through BUS.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>I2Cx_Error</strong> (void)</td>
<td>I2C3 error treatment function.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>I2Cx_MspInit</strong> (I2C_HandleTypeDef *hi2c)</td>
<td>Discovery I2Cx MSP Initialization.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>SPIx_Init</strong> (void)</td>
<td>SPIx Bus initialization.</td>
</tr>
<tr>
<td>static uint8_t</td>
<td><strong>SPIx_WriteRead</strong> (uint8_t Byte)</td>
<td>Sends a Byte through the SPI interface and return the Byte received from the SPI bus.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>SPIx_Error</strong> (void)</td>
<td>SPIx error treatment function.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>SPIx_MspInit</strong> (SPI_HandleTypeDef *hspi)</td>
<td>SPI MSP Init.</td>
</tr>
</tbody>
</table>


Function Documentation

static void `I2Cx_Error` ( void ) [static]

I2C3 error treatment function.

**Return values:**

*None*

Definition at line **413** of file `stm32f3_discovery.c`.

References `I2cHandle`, and `I2Cx_Init()`.

Referenced by `I2Cx_ReadData()`, and `I2Cx_WriteData()`.

---

static void `I2Cx_Init` ( void ) [static]

Discovery I2Cx Bus initialization.

**Return values:**

*None*

Definition at line **347** of file `stm32f3_discovery.c`.

References `ACCELERO_I2C_ADDRESS`, `DISCOVERY_I2Cx`, `I2cHandle`, and `I2Cx_MspInit()`.

Referenced by `COMPASSACCELERO_IO_Init()`, and `I2Cx_Error()`.

---

static void `I2Cx_MspInit` ( I2C_HandleTypeDef * `hi2c` ) [static]

Discovery I2Cx MSP Initialization.

**Parameters:**
hi2c  I2C handle

Return values:
None

Definition at line 323 of file stm32f3_discovery.c.

References DISCOVERY_I2Cx_AF, DISCOVERY_I2Cx_CLK_ENABLE, DISCOVERY_I2Cx_GPIO_CLK_ENABLE, DISCOVERY_I2Cx_GPIO_PORT, DISCOVERY_I2Cx_SCL_PIN, and DISCOVERY_I2Cx_SDA_PIN.

Referenced by I2Cx_Init().

static uint8_t I2Cx_ReadData ( uint16_t Addr,
                              uint8_t Reg
) [static]

Read a value in a register of the device through BUS.

Parameters:
 Addr  Device address on BUS Bus.
 Reg   The target register address to write

Return values:
 Data  read at register @

Definition at line 392 of file stm32f3_discovery.c.

References I2cHandle, I2Cx_Error(), and I2cxTimeout.

Referenced by COMPASSACCELERO_IO_Read().

static void I2Cx_WriteData ( uint16_t Addr,
uint8_t Reg,
uint8_t Value
)
[static]

Write a value in a register of the device through BUS.

**Parameters:**

- **Addr** Device address on BUS Bus.
- **Reg** The target register address to write
- **Value** The target register value to be written

**Return values:**

- None

Definition at line 372 of file `stm32f3_discovery.c`.

References `I2cHandle`, `I2Cx_Error()`, and `I2cxTimeout`.

Referenced by `COMPASSACCELERO_IO_Write()`.

```c
static void SPIx_Error ( void ) [static]
```

SPIx error treatment function.

**Return values:**

- None

Definition at line 485 of file `stm32f3_discovery.c`.

References `SpiHandle`, and `SPIx_Init()`.

Referenced by `SPIx_WriteRead()`.

```c
static void SPIx_Init ( void ) [static]
```
SPIx Bus initialization.

**Return values:**

None

Definition at line 430 of file `stm32f3_discovery.c`.

References `DISCOVERY_SPIx`, `SpiHandle`, and `SPIx_MspInit()`.

Referenced by `GYRO_IO_Init()`, and `SPIx_Error()`.

```c
static void SPIx_MspInit ( SPI_HandleTypeDef * hspi )
```

SPI MSP Init.

**Parameters:**

- `hspi` SPI handle

**Return values:**

None

Definition at line 500 of file `stm32f3_discovery.c`.

References `DISCOVERY_SPIx_AF`, `DISCOVERY_SPIx_CLK_ENABLE`, `DISCOVERY_SPIx_GPIO_CLK_ENABLE`, `DISCOVERY_SPIx_GPIO_PORT`, `DISCOVERY_SPIx_MISO_PIN`, `DISCOVERY_SPIx_MOSI_PIN`, and `DISCOVERY_SPIx_SCK_PIN`.

Referenced by `SPIx_Init()`.

```c
static uint8_t SPIx_WriteRead ( uint8_t Byte )
```

Sends a Byte through the SPI interface and return the Byte received from the SPI bus.
**Parameters:**

*Byte* Byte send.

**Return values:**

*The* received byte value

Definition at line 465 of file *stm32f3_discovery.c*.

References *SpiHandle*, *SPIx_Error()*, and *SpixTimeout*.

Referenced by *GYRO_IO_Read()*, and *GYRO_IO_Write()*.
STM32F3-Discovery BSP User Manual

STM32F3-DISCOVERY LED

Exported Constants

Define for STM32F3_DISCOVERY board. More...
# Defines

```c
#define LEDn   8
#define LED6_PIN GPIO_PIN_15
#define LED6_GPIO_PORT GPIOE
#define LED6_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED6_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define LED8_PIN GPIO_PIN_14
#define LED8_GPIO_PORT GPIOE
#define LED8_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED8_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define LED10_PIN GPIO_PIN_13
#define LED10_GPIO_PORT GPIOE
#define LED10_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED10_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define LED9_PIN GPIO_PIN_12
#define LED9_GPIO_PORT GPIOE
#define LED9_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED9_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define LED7_PIN GPIO_PIN_11
#define LED7_GPIO_PORT GPIOE
#define LED7_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED7_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define LED5_PIN GPIO_PIN_10
#define LED5_GPIO_PORT GPIOE
#define LED5_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED5_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define LED3_PIN GPIO_PIN_9
#define LED3_GPIO_PORT GPIOE
#define LED3_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED3_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define LED4_PIN GPIO_PIN_8
#define LED4_GPIO_PORT GPIOE
```
#define LED4_GPIO_CLK_ENABLE (__HAL_RCC_GPIOE_CLK_ENABLE())
#define LED4_GPIO_CLK_DISABLE (__HAL_RCC_GPIOE_CLK_DISABLE())
#define LEDx_GPIO_CLK_ENABLE(__LED__)__LED__
#define LEDx_GPIO_CLK_DISABLE(__LED__)__LED__
Detailed Description

Define for STM32F3_DISCOVERY board.
Define Documentation

```c
#define LED10_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOE_CLK_DISABLE()
```
Definition at line 154 of file stm32f3_discovery.h.

```c
#define LED10_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOE_CLK_ENABLE()
```
Definition at line 153 of file stm32f3_discovery.h.

```c
#define LED10_GPIO_PORT GPIOE
```
Definition at line 152 of file stm32f3_discovery.h.

```c
#define LED10_PIN GPIO_PIN_13
```
Definition at line 151 of file stm32f3_discovery.h.

```c
#define LED3_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOE_CLK_DISABLE()
```
Definition at line 174 of file stm32f3_discovery.h.

```c
#define LED3_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOE_CLK_ENABLE()
```
Definition at line 173 of file stm32f3_discovery.h.

```c
#define LED3_GPIO_PORT GPIOE
```
Definition at line 172 of file stm32f3_discovery.h.
#define LED3_PIN GPIO_PIN_9

Definition at line 171 of file stm32f3_discovery.h.

#define LED4_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOE_CLK_DISABLE()

Definition at line 179 of file stm32f3_discovery.h.

#define LED4_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOE_CLK_ENABLE()

Definition at line 178 of file stm32f3_discovery.h.

#define LED4_GPIO_PORT GPIOE

Definition at line 177 of file stm32f3_discovery.h.

#define LED4_PIN GPIO_PIN_8

Definition at line 176 of file stm32f3_discovery.h.

#define LED5_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOE_CLK_DISABLE()

Definition at line 169 of file stm32f3_discovery.h.

#define LED5_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOE_CLK_ENABLE()

Definition at line 168 of file stm32f3_discovery.h.
#define LED5_GPIO_PORT GPIOE

Definition at line 167 of file stm32f3_discovery.h.

#define LED5_PIN GPIO_PIN_10

Definition at line 166 of file stm32f3_discovery.h.

#define LED6_GPIO_CLK_DISABLE () __HAL_RCC_GPIOE_CLK_DISABLE

Definition at line 144 of file stm32f3_discovery.h.

#define LED6_GPIO_CLK_ENABLE () __HAL_RCC_GPIOE_CLK_ENABLE

Definition at line 143 of file stm32f3_discovery.h.

#define LED6_GPIO_PORT GPIOE

Definition at line 142 of file stm32f3_discovery.h.

#define LED6_PIN GPIO_PIN_15

Definition at line 141 of file stm32f3_discovery.h.

#define LED7_GPIO_CLK_DISABLE () __HAL_RCC_GPIOE_CLK_DISABLE

Definition at line 164 of file stm32f3_discovery.h.

#define LED7_GPIO_CLK_ENABLE () __HAL_RCC_GPIOE_CLK_ENABLE
#define LED7_GPIO_PORT  GPIOE

Definition at line 162 of file stm32f3_discovery.h.

#define LED7_PIN      GPIO_PIN_11

Definition at line 161 of file stm32f3_discovery.h.

#define LED8_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOE_CLK_DISABLE()

Definition at line 149 of file stm32f3_discovery.h.

#define LED8_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOE_CLK_ENABLE()

Definition at line 148 of file stm32f3_discovery.h.

#define LED8_GPIO_PORT  GPIOE

Definition at line 147 of file stm32f3_discovery.h.

#define LED8_PIN      GPIO_PIN_14

Definition at line 146 of file stm32f3_discovery.h.

#define LED9_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOE_CLK_DISABLE()

Definition at line 159 of file stm32f3_discovery.h.
```c
#define LED9_GPIO_CLK_ENABLE () __HAL_RCC_GPIOE_CLK_ENABLE()

Definition at line 158 of file stm32f3_discovery.h.

#define LED9_GPIO_PORT GPIOE

Definition at line 157 of file stm32f3_discovery.h.

#define LED9_PIN GPIO_PIN_12

Definition at line 156 of file stm32f3_discovery.h.

#define LEDn 8

Definition at line 139 of file stm32f3_discovery.h.

#define LEDx_GPIO_CLK_DISABLE ( __LED__ )

Value:

```

```
```
#define LEDx_GPIO_CLK_ENABLE (__LED__) 

Value:

```c
#define LEDx_GPIO_CLK_ENABLE (__LED__) 

do { if ((__LED__) == LED3) LED3_GPIO_CLK_ENABLE();
    else if ((__LED__) == LED4) LED4_GPIO_CLK_ENABLE();
    else if ((__LED__) == LED5) LED5_GPIO_CLK_ENABLE();
    else if ((__LED__) == LED6) LED6_GPIO_CLK_ENABLE();
    else if ((__LED__) == LED7) LED7_GPIO_CLK_ENABLE();
    else if ((__LED__) == LED8) LED8_GPIO_CLK_ENABLE();
    else if ((__LED__) == LED9) LED9_GPIO_CLK_ENABLE();
    else if ((__LED__) == LED10) LED10_GPIO_CLK_ENABLE();} while(0)
```

Definition at line 181 of file stm32f3_discovery.h.

Referenced by BSP_LED_Init().

Definition at line 190 of file stm32f3_discovery.h.

Referenced by BSP_LED_Init().
STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Firmware</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Firmware Directory Reference**
Directories

directory Drivers

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP
User Manual by doxygen 1.7.6.1
# STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firmware</td>
<td>Drivers</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Drivers Directory Reference**
# STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firmware</td>
<td>Drivers</td>
<td>BSP</td>
<td></td>
</tr>
</tbody>
</table>

## BSP Directory Reference
### Directories

| directory | STM32F3-Discovery |

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by [doxygen](https://www.doxygen.nl) 1.7.6.1
# STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firmware</td>
<td>Drivers</td>
<td>BSP</td>
<td>STM32F3-Discovery</td>
</tr>
</tbody>
</table>

## STM32F3-Discovery Directory Reference
## Files

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stm32f3_discovery.c</td>
<td>This file provides set of firmware functions to manage Leds and push-button available on STM32F3-DISCOVERY Kit from STMicroelectronics.</td>
</tr>
<tr>
<td>stm32f3_discovery.h</td>
<td>This file contains definitions for STM32F3-Discovery’s Leds, push- buttons hardware resources.</td>
</tr>
<tr>
<td>stm32f3_discovery_accelerometer.c</td>
<td>This file provides a set of functions needed to manage the ACCELEROMETER MEMS available on STM32F3-Discovery Kit.</td>
</tr>
<tr>
<td>stm32f3_discovery_accelerometer.h</td>
<td>This file contains definitions for stm32f3_discovery_accelerometer.c firmware driver.</td>
</tr>
<tr>
<td>stm32f3_discovery_gyroscope.c</td>
<td>This file provides a set of functions needed to manage the I3gd20 MEMS accelerometer available on STM32F3-Discovery Kit.</td>
</tr>
<tr>
<td>stm32f3_discovery_gyroscope.h</td>
<td></td>
</tr>
</tbody>
</table>

[code]
This file contains definitions for `stm32f3_discovery_gyroscope.c` firmware driver.

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by doxygen 1.7.6.1
stm32f3_discovery.h

Go to the documentation of this file.

00001 /**
00002 ******************************************
00003 * @file stm32f3_discovery.h
00004 * @author MCD Application Team
00005 * @brief This file contains definitions for STM32F3-Discovery's Leds, push-
00006 * buttons hardware resources.
00007 ******************************************
00008 * @attention
00009 *
00010 * <h2><center>&copy; COPYRIGHT(c) 2016 STM microelectronics</center></h2>
00011 *
00012 * Redistribution and use in source and binary forms, with or without modification,
00013 * are permitted provided that the following conditions are met:
00014 * 1. Redistributions of source code must retain the above copyright notice,
00015 * this list of conditions and the following disclaimer.
00016 * 2. Redistributions in binary form must
reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

*/
/* Define to prevent recursive inclusion -----------------------------*/
#ifndef __STM32F3_DISCOVERY_H
#define __STM32F3_DISCOVERY_H

#ifdef __cplusplus
extern "C"
{
#endif

/** @addtogroup BSP *
@{ */

/** @defgroup STM32F3_DISCOVERY STM32F3-DISCOVERY *
@{ */

/* Includes -----------------------------------------------*/
#include "stm32f3xx_hal.h"

/** @defgroup STM32F3_DISCOVERY_Common STM32F3-DISCOVERY Common *
@{ */

/** @defgroup STM32F3_DISCOVERY_Private_Constants Private Constants *
@{ */

/** @defgroup STM32F3_DISCOVERY_Private_Variables Private Variables *
@{ */
ables Private Variables
00068 * @{
00069 */
00070 /**
00071 * @}
00072 */
00073
00074 /** @defgroup STM32F3_DISCOVERY_Exported_Types Exported Types
00075 * @{
00076 */
00077
00078 /**
00079 * @brief LED Types Definition
00080 */
00081 typedef enum
00082 {
00083 LED3 = 0,
00084 LED4 = 1,
00085 LED5 = 2,
00086 LED6 = 3,
00087 LED7 = 4,
00088 LED8 = 5,
00089 LED9 = 6,
00090 LED10 = 7,
00091 LED_GREEN = LED6,
00092 LED_ORANGE = LED5,
00093 LED_RED = LED3,
00094 LED_BLUE = LED4,
00095 LED_GREEN_2 = LED7,
00096 LED_ORANGE_2 = LED8,
00097 LED_RED_2 = LED10,
00098 LED_BLUE_2 = LED9
00099 }Led_TypeDef;
00100 */
00101
00102 /**
typedef enum
{
    BUTTON_USER = 0
} Button_TypeDef;

typedef enum
{
    BUTTON_MODE_GPIO = 0,
    BUTTON_MODE_EXTI = 1
} ButtonMode_TypeDef;

/**
 * @defgroup STM32F3_DISCOVERY_Exported_Constants Exported Constants
 *
 * @defgroup STM32F3_DISCOVERY_LED STM32F3_DISCOVERY_LED
 */

#if !defined (USE_STM32F3_DISCO)
#define USE_STM32F3_DISCO
#endif
#define LEDn 8
#define LED6_PIN GPIO_PIN_15
#define LED6_GPIO_PORT GPIO
#define LED6_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED6_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define LED8_PIN GPIO_PIN_14
#define LED8_GPIO_PORT GPIO
#define LED8_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED8_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define LED10_PIN GPIO_PIN_13
#define LED10_GPIO_PORT GPIO
#define LED10_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED10_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define LED9_PIN GPIO_PIN_12
#define LED9_GPIO_PORT GPIO
#define LED9_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED9_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()

#define LED7_PIN GPIO_PIN_11
#define LED7_GPIO_PORT GPIOE
#define LED7_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED7_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()

#define LED5_PIN GPIO_PIN_10
#define LED5_GPIO_PORT GPIOE
#define LED5_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED5_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()

#define LED3_PIN GPIO_PIN_9
#define LED3_GPIO_PORT GPIOE
#define LED3_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED3_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()

#define LED4_PIN GPIO_PIN_8
#define LED4_GPIO_PORT GPIOE
#define LED4_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define LED4_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
AL_RCC_GPIOE_CLK_DISABLE()

00180 #define LEDx_GPIO_CLK_ENABLE(__LED__) do {
00181   if ((__LED__) == LED3) LED3_GPIO_CLK_ENABLE(); else
00182   if ((__LED__) == LED4) LED4_GPIO_CLK_ENABLE(); else
00183   if ((__LED__) == LED5) LED5_GPIO_CLK_ENABLE(); else
00184   if ((__LED__) == LED6) LED6_GPIO_CLK_ENABLE(); else
00185   if ((__LED__) == LED7) LED7_GPIO_CLK_ENABLE(); else
00186   if ((__LED__) == LED8) LED8_GPIO_CLK_ENABLE(); else
00187   if ((__LED__) == LED9) LED9_GPIO_CLK_ENABLE(); else
00188   if ((__LED__) == LED10) LED10_GPIO_CLK_ENABLE();} while(0)

00189 #define LEDx_GPIO_CLK_DISABLE(__LED__) (((__
00190   __LED__) == LED3) ? LED3_GPIO_CLK_DISABLE() :
00191   ((__
00192   __LED__) == LED4) ? LED4_GPIO_CLK_DISABLE() :
00193   ((__
00194   __LED__) == LED5) ? LED5_GPIO_CLK_DISABLE() :
00195   ((__
00196   __LED__) == LED6) ? LED6_GPIO_CLK_DISABLE() :
00197   ((__
00198   __LED__) == LED7) ? LED7_GPIO_CLK_DISABLE() :\
### STM32F3_DISCOVERY_BUTTON

@defgroup STM32F3_DISCOVERY_BUTTON STM32F3-DISCOVERY BUTTON

#define BUTTONn 1

#define USER_BUTTON_PIN GPIO_PIN_0
#define USER_BUTTON_GPIO_PORT GPIOA
#define USER_BUTTON_GPIO_CLK_ENABLE() __HAL_RCC_GPIOA_CLK_ENABLE()
#define USER_BUTTON_GPIO_CLK_DISABLE() __HAL_RCC_GPIOA_CLK_DISABLE()
#define USER_BUTTON_EXTI_IRQn EXTI0_IRQn
#define BUTTONx_GPIO_CLK_ENABLE(__BUTTON__) do {
    if ((__BUTTON__) == BUTTON_USER) USER_BUTTON_GPIO_CLK_ENABLE();
} while(0)

#define BUTTONx_GPIO_CLK_DISABLE(__BUTTON__) (((__BUTTON__) == BUTTON_USER) ? USER_BUTTON_GPIO_CLK_DISABLE() : 0)
### STM32F3_DISCOVERY_COM

#### SPI Interface Pin Definitions (SPI1 used)

```c
#define DISCOVERY_SPIx_SPI1 __HAL_RCC_SPI1_CLK_ENABLE()
#define DISCOVERY_SPIx_GPIO_PORT GPIOA /* GPIOA */
#define DISCOVERY_SPIx_AF GPIO_AF5_SPI1
#define DISCOVERY_SPIx_GPIO_CLK_ENABLE() __HAL_RCC_GPIOA_CLK_ENABLE()
#define DISCOVERY_SPIx_GPIO_CLK_DISABLE() __HAL_RCC_GPIOA_CLK_DISABLE()
#define DISCOVERY_SPIx_SCK_PIN GPIO_PIN_5 /* PA.05 */
#define DISCOVERY_SPIx_MISO_PIN GPIO_PIN_6 /* PA.06 */
#define DISCOVERY_SPIx_MOSI_PIN GPIO_PIN_7 /* PA.07 */
```

Maximum Timeout values for flags waiting loops. These timeouts are not based...
on accurate values, they just guarantee that the application will not remain stuck if the SPI communication is corrupted.

You may modify these timeout values depending on CPU frequency and application conditions (interrupts routines ...). */

#define SPIx_TIMEOUT_MAX ((uint32_t)0x1000)

/*#####################	I2Cx	###############
#####################*/

/**
@brief Definition for I2C Interface pins (I2C1 used)
*/
#define DISCOVERY_I2Cx

#define DISCOVERY_I2Cx_CLK_ENABLE() __HAL_RCC_I2C1_CLK_ENABLE()
#define DISCOVERY_I2Cx_CLK_DISABLE() __HAL_RCC_I2C1_CLK_DISABLE()
#define DISCOVERY_I2Cx_FORCE_RESET() __HAL_RCC_I2C1_FORCE_RESET()
#define DISCOVERY_I2Cx_RELEASE_RESET() __HAL_RCC_I2C1_RELEASE_RESET()

#define DISCOVERY_I2Cx_SCL_PIN GPIO_PIN_6 /* PB.06 */
#define DISCOVERY_I2Cx_SDA_PIN GPIO_PIN_7 /* PB.07 */

#define DISCOVERY_I2Cx_GPIO_PORT GPIOB /* GPIOB */
#define DISCOVERY_I2Cx_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_ENABLE()
#define DISCOVERY_I2Cx_GPIO_CLK_DISABLE()
__HAL_RCC_GPIOB_CLK_DISABLE()
#define DISCOVERY_I2Cx_AF
GPIO_AF4_I2C1

/*
Maximum Timeout values for flags waiting loops. These timeouts are not based
on accurate values, they just guarantee that the application will not remain
stuck if the I2C communication is corrupted.
You may modify these timeout values depending on CPU frequency and application
conditions (interrupts routines ...). */

#define I2Cx_TIMEOUT_MAX
0x10000

/**
@}
*/

/**
@defgroup STM32F3_DISCOVERY_COMPONENT STM32F3-DISCOVERY COMPONENT
@

/*#####################
GYRO ###############*/
/*
Read/Write command */
#define READWRITE_CMD ((uint8_t)0x80)
/*
Multiple byte read/write command */
#define MULTIPLEBYTE_CMD ((uint8_t)0x40)
/* Dummy Byte Send by the SPI Master device in order to generate the Clock to the Slave device
/*
 * Chip Select macro definition */
#define GYRO_CS_LOW() HAL_GPIO_WritePin(GYRO_CS_GPIO_PORT, GYRO_CS_PIN, GPIO_PIN_RESET)
#define GYRO_CS_HIGH() HAL_GPIO_WritePin(GYRO_CS_GPIO_PORT, GYRO_CS_PIN, GPIO_PIN_SET)

/**
 * @brief GYRO SPI Interface pins
 */
#define GYRO_CS_GPIO_PORT GPIOE
#define GYRO_CS_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define GYRO_CS_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define GYRO_CS_PIN GPIO_PIN_3 /* PE.03 */
#define GYRO_INT_GPIO_PORT GPIOE
#define GYRO_INT_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define GYRO_INT_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define GYRO_INT1_PIN GPIO_PIN_0 /* PE.00 */
#define GYRO_INT1_EXTI_IRQHandler EXTI0_IRQn
#define GYRO_INT2_PIN GPIO_PIN_1 /* PE.01 */
#define GYRO_INT2_EXTI_IRQHandler EXTI1_IRQn
/* # ACCELEROMETER # # # # # # # */

/**
 * @brief ACCELEROMETER I2C1 Interface pins
 */

#define ACCELERO_I2C_ADDRESS 0x32

#define ACCELERO_DRDY_PIN GPIO_PIN_2 /* PE.02 */
#define ACCELERO_DRDY_GPIO_PORT GPIOE /* GPIOE */
#define ACCELERO_DRDY_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define ACCELERO_DRDY_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define ACCELERO_DRDY_EXTI_IRQn EXTI2_TSC_IRQHandler /* TAMP_STAMP_IRQHandler */

#define ACCELERO_INT_GPIO_PORT GPIOE /* GPIOE */
#define ACCELERO_INT_GPIO_CLK_ENABLE() __HAL_RCC_GPIOE_CLK_ENABLE()
#define ACCELERO_INT_GPIO_CLK_DISABLE() __HAL_RCC_GPIOE_CLK_DISABLE()
#define ACCELERO_INT1_PIN GPIO_PIN_4 /* PE.04 */
#define ACCELERO_INT1_EXTI_IRQHandler EXTI4_IRQHandler
#define ACCELERO_INT2_PIN GPIO_PIN_5 /* PE.05 */
#define ACCELERO_INT2_EXTI_IRQHandler EXTI9_5_IRQHandler

/** *
 * @}
/**
 * @defgroup STM32F3_DISCOVERY_Exported_Functions Exported Functions
 * @{
*/

uint32_t BSP_GetVersion(void);
void BSP_LED_Init(Led_TypeDef Led);
void BSP_LED_On(Led_TypeDef Led);
void BSP_LED_Off(Led_TypeDef Led);
void BSP_LED_Toggle(Led_TypeDef Led);
void BSP_PB_Init(Button_TypeDef Button, ButtonMode_TypeDef ButtonMode);
uint32_t BSP_PB_GetState(Button_TypeDef Button);

/**
 * @}*/

#endif __cplusplus
} #endif
#endif /* __STM32F3_DISCOVERY_H */

/**************************** (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
**stm32f3_discovery.c**

Go to the documentation of this file.

```
00001 /*
00002  ******************************************
00003  ************************************
00004  * @file   stm32f3_discovery.c
00005  * @author MCD Application Team
00006  * @brief   This file provides set of firmware functions to manage Leds and
00007  *          push-button available on STM32F3-DISCOVERY Kit from STMicroelectronics.
00008  ******************************************
00009  ************************************
00100  * @attention
00101  *
00102  * <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>
00103  *
00104  * Redistribution and use in source and binary forms, with or without modification,
00105  * are permitted provided that the following conditions are met:
00106  * 1. Redistributions of source code must retain the above copyright notice,
00107  * 2. Redistributions in binary form must reproduce the above copyright notice,
00108  *    this list of conditions and the following disclaimer.
```
2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.
3. Neither the name of STMicroelectronics nor the names of its contributors
   may be used to endorse or promote products derived from this software
   without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
/* Includes ------------------------------------------*/
#include "stm32f3_discovery.h"

/**	@addtogroup BSP
* @{
*/

/**	@addtogroup STM32F3_DISCOVERY
* @brief This file provides set of firmware functions to manage Leds and push-button
* available on STM32F3-Discovery Kit from STMicroelectronics.
* @{
*/

/**	@addtogroup STM32F3_DISCOVERY_Common
* @{
*/

/**	@addtogroup STM32F3_DISCOVERY_Private_Constants
* @{
*/

/**	@brief STM32F3 DISCOVERY BSP Driver version number V2.1.5
* */

#define __STM32F3_DISCO_BSP_VERSION_MAIN ( 0x02) /*!< [31:24] main version */
#define __STM32F3_DISCO_BSP_VERSION_SUB1 ( 0x01) /*!< [23:16] sub1 version */
#define __STM32F3_DISCO_BSP_VERSION_SUB2 ( 0x05) /*!< [15:8] sub2 version */
#define __STM32F3_DISCO_BSP_VERSION_RC ( 0x00) /*!< [7:0] release candidate */
#define __STM32F3_DISCO_BSP_VERSION ((__STM32F3_DISCO_BSP_VERSION_MAIN << 24)\ |(__STM32F3_DISCO_BSP_VERSION_SUB1 << 16)\ |(__STM32F3_DISCO_BSP_VERSION_SUB2 << 8 )\ |(__STM32F3_DISCO_BSP_VERSION_RC))

/**
* @}
*/

/**
* @addtogroup STM32F3_DISCOVERY_Private_Variables
* @{
*/

/**
* @brief LED variables
*/
GPIO_TypeDef* LED_PORT[LEDn] = {LED3_GPIO_PORT, LED4_GPIO_PORT, LED5_GPIO_PORT, LED6_GPIO_PORT, LED7_GPIO_PORT, LED8_GPIO_PORT, LED9_GPIO_PORT, LED10_GPIO_PORT};

const uint16_t LED_PIN[LEDn] = {LED3_PIN, LED4_PIN, LED5_PIN, LED6_PIN, LED7_PIN, LED8_PIN, LED9_PIN, LED10_PIN};

/**
* @brief BUTTON variables
*/
GPIO_TypeDef* BUTTON_PORT[BUTTONn] = {USER_BUTTON_GPIO_PORT};

const uint16_t BUTTON_PIN[BUTTONn] = {USER_B
UTTON_PIN};
00089 const uint8_t BUTTON_IRQn[BUTTONn] = {USER_B
00090 UTTON_EXTI_IRQn};
00091 /**<
00092 	* @brief BUS variables
00093 */
00094 #ifdef HAL_SPI_MODULE_ENABLED
00095 uint32_t SpixTimeout = SPIx_TIMEOUT_MAX;
00096 /*<! Value of Timeout when SPI communication fails
00097 */
00098 static SPI_HandleTypeDef SpiHandle;
00099 #endif
00100 #ifdef HAL_I2C_MODULE_ENABLED
00101 static I2C_HandleTypeDef I2cHandle;
00102 uint32_t I2cxTimeout = I2Cx_TIMEOUT_MAX;
00103 /*<! Value of Timeout when I2C communication fails
00104 */
00105 #endif
00106 #ifdef HAL_I2C_MODULE_ENABLED
00107 /* I2Cx bus function */
00108 /** @defgroup STM32F3_DISCOVERY_BUS Bus Operation functions
00109 */
00110 #endif
00111 #ifdef HAL_I2C_MODULE_ENABLED
00112 /* I2Cx bus function */
00113 static void I2Cx_Init(void);
00114 static void I2Cx_WriteData(uint16_t Addr,
00115 static void I2Cx_ReadData(uint16_t Addr,
00116 static void I2Cx_Error (void);
static void I2Cx_MspInit(I2C_HandleTypeDef *hi2c);
#endif

#ifdef HAL_SPI_MODULE_ENABLED
/* SPIx bus function */
static void SPIx_Init(void);
static uint8_t SPIx_WriteRead(uint8_t byte);
static void SPIx_Error (void);
static void SPIx_MspInit(SPI_HandleTypeDef *hspi);
#endif

#ifdef HAL_SPI_MODULE_ENABLED
/* Link function for GYRO peripheral */
void GYRO_IO_Init(void);
void GYRO_IO_Write(uint8_t* pBuffer, uint8_t WriteAddr, uint16_t NumByteToWrite);
void GYRO_IO_Read(uint8_t* pBuffer, uint8_t ReadAddr, uint16_t NumByteToRead);
#endif

#ifdef HAL_I2C_MODULE_ENABLED
/* Link function for COMPASS / ACCELEROMETER peripheral */
void COMPASSACCELERO_IO_Init(void);
void COMPASSACCELERO_IO_ITConfig(void);
void COMPASSACCELERO_IO_Write(uint16_t DeviceAddr, uint8_t RegisterAddr, uint8_t Value);
uint8_t COMPASSACCELERO_IO_Read(uint16_t DeviceAddr, uint8_t RegisterAddr);
#endif

/**
 * @}
 */
/**
 * @addtogroup STM32F3_DISCOVERY_Exported_Functions
 */

/**
 * @brief This method returns the STM32F3-DISCOVERY BSP Driver revision
 * @retval version : 0xXYZR (8 bits for each decimal, R for RC)
 */
uint32_t BSP_GetVersion(void) {
    return __STM32F3_DISCO_BSP_VERSION;
}

/**
 * @brief Configures LED GPIO.
 * @param Led Specifies the Led to be configured.
 * This parameter can be one of following parameters:
 * @arg LED_RED
 * @arg LED_BLUE
 * @arg LED_ORANGE
 * @arg LED_GREEN
 * @arg LED_GREEN2
 * @arg LED_ORANGE2
 * @arg LED_BLUE2
 * @arg LED_RED2
 * @retval None
 */
void BSP_LED_Init(Led_TypeDef Led) {
    GPIO_InitTypeDef GPIO_InitStruct;
    /*
    void BSP_LED_Init(Led_TypeDef Led)
    {*/
    GPIO_InitTypeDef GPIO_InitStruct;
/* Enable the GPIO_LED Clock */
LEDx_GPIO_CLK_ENABLE(Led);

/* Configure the GPIO_LED pin */
GPIO_InitStruct.Pin = LED_PIN[Led];
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_PULLUP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
HAL_GPIO_Init(LED_PORT[Led], &GPIO_InitStruct);

HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[Led], GPIO_PIN_RESET);
}

/**
 * @brief Turns selected LED On.
 * @param Led Specifies the Led to be set on.
 * This parameter can be one of following parameters:
 * @arg LED_RED
 * @arg LED4
 * @arg LED5
 * @arg LED6
 * @arg LED7
 * @arg LED8
 * @arg LED9
 * @arg LED10
 * @retval None
 */
void BSP_LED_On(Led_TypeDef Led) {
HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[Led], GPIO_PIN_RESET);
ed], GPIO_PIN_SET);
00209 }
00210
00211 /**<
00212  * @brief Turns selected LED Off.
00213  * @param Led Specifies the Led to be set off.
00214  * This parameter can be one of following parameters:
00215  *  @arg LED_RED
00216  *  @arg LED_BLUE
00217  *  @arg LED_ORANGE
00218  *  @arg LED_GREEN
00219  *  @arg LED_GREEN2
00220  *  @arg LED_ORANGE2
00221  *  @arg LED_BLUE2
00222  *  @arg LED_RED2
00223  *  @retval None
00224 */
00225 void BSP_LED_Off(Led_TypeDef Led)
00226 {
00227  HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[Led], GPIO_PIN_RESET);
00228 }
00229
00230 /**<
00231  * @brief Toggles the selected LED.
00232  * @param Led Specifies the Led to be toggled.
00233  * This parameter can be one of following parameters:
00234  *  @arg LED_RED
00235  *  @arg LED_BLUE
00236  *  @arg LED_ORANGE
00237  *  @arg LED_GREEN
00238  *  @arg LED_GREEN2
00239  *  @arg LED_ORANGE2
void BSP_LED_Toggle(Led_TypeDef Led) {
    HAL_GPIO_TogglePin(LED_PORT[Led], LED_PIN[Led]);
}

void BSP_PB_Init(Button_TypeDef Button, ButtonMode_TypeDef ButtonMode)
{
    GPIO_InitTypeDef GPIO_InitStruct;
    /* Enable the BUTTON Clock */
    BUTTONx_GPIO_CLK_ENABLE(Button);
    __HAL_RCC_SYSCFG_CLK_ENABLE();
}
if (ButtonMode == BUTTON_MODE_GPIO)
{
    /* Configure Button pin as input */
    GPIO_InitStruct.Pin = BUTTON_PIN[Button];
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
    GPIO_InitStruct.Pull = GPIO_PULLDOWN;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
    HAL_GPIO_Init(BUTTON_PORT[Button], &GPIO_InitStruct);
}

if (ButtonMode == BUTTON_MODE_EXTI)
{
    /* Configure Button pin as input with External interrupt */
    GPIO_InitStruct.Pin = BUTTON_PIN[Button];
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
    GPIO_InitStruct.Mode = GPIO_MODE_IT RISING;
    HAL_GPIO_Init(BUTTON_PORT[Button], &GPIO_InitStruct);

    /* Enable and set Button EXTI Interrupt to the lowest priority */
    HAL_NVIC_SetPriority((IRQn_Type)(BUTTON_IRQn[Button]), 0x0F, 0x00);
    HAL_NVIC_EnableIRQ((IRQn_Type)(BUTTON_IRQn[Button]));
}

}
/**
 * @brief Returns the selected Push Button state.
 * @param Button Specifies the Button to be checked.
 * This parameter should be: BUTTON_USER
 * @retval The Button GPIO pin value.
 */

uint32_t BSP_PB_GetState(Button_TypeDef Button)
{
    return HAL_GPIO_ReadPin(BUTTON_PORT[Button], BUTTON_PIN[Button]);
}

/**
 * @defgroup STM32F3_DISCOVERY_BUS
 * @{
 */

/*******************************************
***********************************
BUS OPERATIONS
********************************************
***********************************/

#ifdef HAL_I2C_MODULE_ENABLED
/******************************* I2C Routine s***********************************/

/**
 * @brief Discovery I2Cx MSP Initialization
 * @param hi2c I2C handle
 * @retval None
 */
static void I2Cx_MspInit(I2C_HandleTypeDef *hi2c) {
    GPIO_InitTypeDef GPIO_InitStructure;
    /* Enable SCK and SDA GPIO clocks */
    DISCOVERY_I2Cx_GPIO_CLK_ENABLE();
    /* I2Cx SD1 & SCK pin configuration */
    GPIO_InitStructure.Pin = (DISCOVERY_I2Cx_SDA_PIN | DISCOVERY_I2Cx_SCL_PIN);
    GPIO_InitStructure.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStructure.Pull = GPIO_PULLDOWN;
    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_HIGH;
    GPIO_InitStructure.Alternate = DISCOVERY_I2Cx_AF;
    HAL_GPIO_Init(DISCOVERY_I2Cx_GPIO_PORT, &GPIO_InitStructure);
    /* Enable the I2C clock */
    DISCOVERY_I2Cx_CLK_ENABLE();
}

/**
 * @brief Discovery I2Cx Bus initialization
 * @retval None
 */
static void I2Cx_Init(void) {
    if(HAL_I2C_GetState(&I2cHandle) == HAL_I2C_STATE_RESET) {
        I2cHandle.Instance = DISCOVERY_I2Cx;
        I2cHandle.Init.OwnAddress1 = ACCELERO_I2C_ADDRESS;
    }
00353    I2cHandle.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
00354    I2cHandle.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
00355    I2cHandle.Init.OwnAddress2 = 0;
00356    I2cHandle.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
00357    I2cHandle.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
00358
00359    /* Init the I2C */
00360    I2Cx_MspInit(&I2cHandle);
00361    HAL_I2C_Init(&I2cHandle);
00362 }
00363 }
00364
00365 /**
00366    * @brief Write a value in a register of the device through BUS.
00367    * @param Addr Device address on BUS Bus.
00368    * @param Reg The target register address to write
00369    * @param Value The target register value to be written
00370    * @retval None
00371 */
00372 static void I2Cx_WriteData(uint16_t Addr, uint8_t Reg, uint8_t Value)
00373 {
00374    HAL_StatusTypeDef status = HAL_OK;
00375
00376    status = HAL_I2C_Mem_Write(&I2cHandle, Addr, (uint16_t)Reg, I2C_MEMADD_SIZE_8BIT, &Value, 1, I2cxTimeout);
00377
00378    /* Check the communication status */
if(status != HAL_OK) {
    /* Execute user timeout callback */
    I2Cx_Error();
}

/**
 * @brief Read a value in a register of the device through BUS.
 * @param Addr Device address on BUS Bus.
 * @param Reg The target register address to write
 * @retval Data read at register @
 */
static uint8_t I2Cx_ReadData(uint16_t Addr, uint8_t Reg) {
    HAL_StatusTypeDef status = HAL_OK;
    uint8_t value = 0;
    status = HAL_I2C_Mem_Read(&I2cHandle, Addr, Reg, I2C_MEMADD_SIZE_8BIT, &value, 1, I2cxTimeout);
    /* Check the communication status */
    if(status != HAL_OK) {
        /* Execute user timeout callback */
        I2Cx_Error();
    }
    return value;
}

/**
static void I2Cx_Error (void)
{
    /* De-initialize the I2C comunication BUS */
    HAL_I2C_DeInit(&I2cHandle);
    /* Re- Initialize the I2C comunication BUS */
    I2Cx_Init();
}

#define HAL_SPI_MODULE_ENABLED

/**************************** SPI Routine ****************************/

/**
@brief SPIx Bus initialization
@retval None
*/
static void SPIx_Init(void)
{
    if(HAL_SPI_GetState(&SpiHandle) == HAL_SPI_STATE_RESET)
    {
        /* SPI Config */
        SpiHandle.Instance = DISCOVERY_SPIx;
        /* SPI baudrate is set to 5.6 MHz (PCLK2/SPI_BaudRatePrescaler = 90/16 = 5.625 MHz)
        to verify these constraints:
        ILI9341 LCD SPI interface max baudrate is 10MHz for write and 6.66MHz for read
        l3gd20 SPI interface max baudrate is 10MHz for write/read
    }
PCLK2 frequency is set to 90 MHz

SpiHandle.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_16;
SpiHandle.Init.Direction = SPI_DIRECTION_2LINES;
SpiHandle.Init.CLKPhase = SPI_PHASE_1EDGE;
SpiHandle.Init.CLKPolarity = SPI_POLARITY_LOW;
SpiHandle.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
SpiHandle.Init.CRCPolynomial = 7;
SpiHandle.Init.DataSize = SPI_DATASIZE_8BIT;
SpiHandle.Init.FirstBit = SPI_FIRSTBIT_MSB;
SpiHandle.Init.NSS = SPI_NSS_SOFT;
SpiHandle.Init.TIMode = SPI_TIMODE_DISABLE;
SpiHandle.Init.Mode = SPI_MODE_MASTER;

SPIx_MspInit(&SpiHandle);
HAL_SPI_Init(&SpiHandle);

/**
 @brief Sends a Byte through the SPI interface and return the Byte received from the SPI bus.
 @param Byte Byte send.
 @retval The received byte value
 */
static uint8_t SPIx_WriteRead(uint8_t Byte)
{


uint8_t receivedbyte = 0;

/* Send a Byte through the SPI peripheral */
/* Read byte from the SPI bus */
if (HAL_SPI_TransmitReceive(&SpiHandle, (uint8_t*)&Byte, (uint8_t*)&receivedbyte, 1, SpiTimeout) != HAL_OK)
{
    SPIx_Error();
}
return receivedbyte;

/**
 * @brief SPIx error treatment function
 * @retval None
 */
static void SPIx_Error (void)
{
    /* De-initialize the SPI comunication BUS */
    HAL_SPI_DeInit(&SpiHandle);
    /* Re- Initialize the SPI comunication BUS */
    SPIx_Init();
}

/**
 * @brief SPI MSP Init
 * @param hsipi SPI handle
 * @retval None
 */
00500  static void SPIx_MspInit(SPI_HandleTypeDef * hspi) {
00501  {
00502      GPIO_InitTypeDef GPIO_InitStructure;
00503
00504  /* Enable SPI1 clock */
00505      DISCOVERY_SPIx_CLK_ENABLE();
00506
00507  /* enable SPI1 gpio clock */
00508      DISCOVERY_SPIx_GPIO_CLK_ENABLE();
00509
00510  /* configure SPI1 SCK, MOSI and MISO */
00511      GPIO_InitStructure.Pin = (DISCOVERY_SPIx_SCK_PIN | DISCOVERY_SPIx_MOSI_PIN | DISCOVERY_SPIx_MISO_PIN);
00512      GPIO_InitStructure.Mode = GPIO_MODE_AF_PP;
00513      GPIO_InitStructure.Pull = GPIO_NOPULL; /* or GPIO_PULLDOWN */
00514      GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_HIGH;
00515      GPIO_InitStructure.Alternate = DISCOVERY_SPIx_AF;
00516      HAL_GPIO_Init(DISCOVERY_SPIx_GPIO_PORT, &GPIO_InitStructure);
00517  }
00518
00519  /**
00520      * @}
00521
00522  /** @defgroup STM32F3_DISCOVERY_LINK_OPERATIONS Link Operation functions
00523      * @{*
00524    */
00525
00526  /******************************************************************************
00527  ***********************************************************************
00528  LINK OPERATIONS
void GYRO_IO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;

    /* Configure the Gyroscope Control pins */
    /* Enable CS GPIO clock and Configure GPIO PIN for Gyroscope Chip select */
    GYRO_CS_GPIO_CLK_ENABLE();
    GPIO_InitStructure.Pin = GYRO_CS_PIN;
    GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStructure.Pull = GPIO_NOPULL;
    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_HIGH;
    HAL_GPIO_Init(GYRO_CS_GPIO_PORT, &GPIO_InitStructure);

    /* Deselect : Chip Select high */
    GYRO_CS_HIGH();

    /* Enable INT1, INT2 GPIO clock and Configure GPIO PINs to detect Interrupts */
    GYRO_INT_GPIO_CLK_ENABLE();
    GPIO_InitStructure.Pin = GYRO_INT1_PIN | GYRO_INT2_PIN;
    GPIO_InitStructure.Mode = GPIO_MODE_INPUT;
00555  GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_HIGH;
00556  GPIO_InitStructure.Pull = GPIO_NOPULL;
00557  HAL_GPIO_Init(GYRO_INT_GPIO_PORT, &GPIO_InitStructure);
00558
00559  SPIx_Init();
00560 }
00561
00562 /**
00563  * @brief  Writes one byte to the GYROSCOPE.
00564  * @param  pBuffer pointer to the buffer containing the data to be written to the GYROSCOPE.
00565  * @param  WriteAddr GYROSCOPE's internal address to write to.
00566  * @param  NumByteToWrite Number of bytes to write.
00567  * @retval None
00568 */
00569 void GYRO_IO_Write(uint8_t* pBuffer, uint8_t WriteAddr, uint16_t NumByteToWrite)
00570 {
00571   /* Configure the MS bit:
00572       - When 0, the address will remain unchanged in multiple read/write commands.
00573       - When 1, the address will be auto incremented in multiple read/write commands.
00574 */
00575   if(NumByteToWrite > 0x01)
00576     {
00577     WriteAddr |= (uint8_t)MULTIPLEBYTE_CMD;
00578   }
00579   /* Set chip select Low at the start of the transmission */
00580   GYRO_CS_LOW();
SPIx_WriteRead(WriteAddr);
/* Send the data that will be written into the device (MSB First) */
while(NumByteToWrite >= 0x01) {
   SPIx_WriteRead(*pBuffer);
   NumByteToWrite--;
   pBuffer++;
}
/* Set chip select High at the end of the transmission */
GYRO_CS_HIGH();
/**
@brief Reads a block of data from the GYROSCOPE.
@param pBuffer pointer to the buffer that receives the data read from the GYROSCOPE.
@param ReadAddr GYROSCOPE's internal address to read from.
@param NumByteToRead number of bytes to read from the GYROSCOPE.
@return None */
void GYRO_IO_Read(uint8_t* pBuffer, uint8_t ReadAddr, uint16_t NumByteToRead)
{
if(NumByteToRead > 0x01) {
   ReadAddr |= (uint8_t)(READWRITE_CMD | MULTIPLEBYTE_CMD);
}
else
{
    ReadAddr |= (uint8_t)READWRITE_CMD;
}

/* Set chip select Low at the start of the transmission */
GYRO_CS_LOW();
/* Send the Address of the indexed register */
SPIx_WriteRead(ReadAddr);
/* Receive the data that will be read from the device (MSB First) */
while(NumByteToRead > 0x00)
{
    /* Send dummy byte (0x00) to generate the SPI clock to GYROSCOPE (Slave device) */
    *pBuffer = SPIx_WriteRead(DUMMY_BYTE);
    NumByteToRead--;
    pBuffer++;
}
/* Set chip select High at the end of the transmission */
GYRO_CS_HIGH();

#endif /* HAL_SPI_MODULE_ENABLED */

#ifdef HAL_I2C_MODULE_ENABLED
/******************************** LINK ACCELEROMETER *************************************/
/**
 * @brief Configures COMPASS / ACCELEROMETER I2C interface.
 * @retval None
 */
#endif /* HAL_I2C_MODULE_ENABLED */
00640 void COMPASSACCELERO_IO_Init(void) 
00641 {
00642    GPIO_InitTypeDef GPIO_InitStructure;
00643
00644    /* Enable DRDY clock */
00645    ACCELERO_DRDY_GPIO_CLK_ENABLE();
00646
00647    /* Enable INT1 & INT2 GPIO clock */
00648    ACCELERO_INT_GPIO_CLK_ENABLE();
00649
00650    /* Mem's DRDY pin configuration */
00651    GPIO_InitStructure.Pin = ACCELERO_DRDY_PIN;
00652    GPIO_InitStructure.Mode = GPIO_MODE_INPUT;
00653    GPIO_InitStructure.Pull = GPIO_NOPULL;
00654    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_HIGH;
00655    HAL_GPIO_Init(ACCELERO_DRDY_GPIO_PORT, &GPIO_InitStructure);
00656
00657    /* Enable and set Button EXTI Interrupt to the lowest priority */
00658    HAL_NVIC_SetPriority(ACCELERO_DRDY_EXTI_IRQn, 0x0F, 0x00);
00659    HAL_NVIC_EnableIRQ(ACCELERO_DRDY_EXTI_IRQn);
00660
00661    /* Configure GPIO PINs to detect Interrupts */
00662    GPIO_InitStructure.Pin = ACCELERO_INT1_PIN | ACCELERO_INT2_PIN;
00663    GPIO_InitStructure.Mode = GPIO_MODE_INPUT;
00664    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_HIGH;
00665    GPIO_InitStructure.Pull = GPIO_NOPULL;
00666    HAL_GPIO_Init(ACCELERO_INT_GPIO_PORT, &GPIO_InitStructure);
void COMPASSACCELERO_IO_ITConfig(void) {
    GPIO_InitTypeDef GPIO_InitStructure;

    /* Enable INT1 & INT2 GPIO clock */
    ACCELERO_INT_GPIO_CLK_ENABLE();

    /* Configure GPIO PINs to detect Interrupts */
    GPIO_InitStructure.Pin = ACCELERO_INT1_PIN | ACCELERO_INT2_PIN;
    GPIO_InitStructure.Mode = GPIO_MODE_IT_RISING;
    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_HIGH;
    GPIO_InitStructure.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(ACCELERO_INT_GPIO_PORT, &GPIO_InitStructure);

    /* Enable and set Button EXTI Interrupt to the lowest priority */
    HAL_NVIC_SetPriority(ACCELERO_INT1_EXTI_IRQn, 0x0F, 0x00);
    HAL_NVIC_EnableIRQ(ACCELERO_INT1_EXTI_IRQn);
}

I2Cx_Init();

*/
/**
 * @brief Writes one byte to the COMPASS / ACCELEROMETER.
 * @param DeviceAddr specifies the slave address to be programmed.
 * @param RegisterAddr specifies the COMPASS / ACCELEROMETER register to be written.
 * @param Value Data to be written
 * @retval None
 */

void COMPASSACCELERO_IO_Write(uint16_t DeviceAddr, uint8_t RegisterAddr, uint8_t Value)
{
    /* call I2Cx Read data bus function */
    I2Cx_WriteData(DeviceAddr, RegisterAddr, Value);
}

/**
 * @brief Reads a block of data from the COMPASS / ACCELEROMETER.
 * @param DeviceAddr specifies the slave address to be programmed(ACC_I2C_ADDRESS or MAG_I2C_ADDRESS).
 * @param RegisterAddr specifies the COMPASS / ACCELEROMETER internal address register to read from
 * @retval ACCELEROMETER register value
 */

uint8_t COMPASSACCELERO_IO_Read(uint16_t DeviceAddr, uint8_t RegisterAddr)
{
    /* call I2Cx Read data bus function */
    return I2Cx_ReadData(DeviceAddr, RegisterAddr);
}

#endif /* HAL_I2C_MODULE_ENABLED */
/********************
(C)
COPYRIGHT STMicroelectronics
*****END OF FILE****/
stm32f3_discovery_accelerometer.h

Go to the documentation of this file.

00001  /**
00002  ******************************************
00003  * @file  stm32f3_discovery_accelerometer.h
00004  * @author MCD Application Team
00005  * @brief This file contains definitions for stm32f3_discovery_accelerometer.c
00006  * firmware driver.
00007  ******************************************
00008  * @attention
00009  *
00010  * <h2><center>&copy; COPYRIGHT(c) 2016 STM microelectronics</center></h2>
00011  *
00012  * Redistribution and use in source and binary forms, with or without modification,
00013  * are permitted provided that the following conditions are met:
00014  * 1. Redistributions of source code must retain the above copyright notice,
00015  * this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.

3. Neither the name of STMicroelectronics nor the names of its contributors
may be used to endorse or promote products derived from this software
without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

*/
/* Define to prevent recursive inclusion -------------------*/

#ifndef __STM32F3_DISCOVERY_ACCELERO_H
#define __STM32F3_DISCOVERY_ACCELERO_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes ------------------------------------------*/

#include "stm32f3_discovery.h"

/* Include Gyroscope component driver */
#include "../Components/lsm303dlhc/lsm303dlhc"c.h"

/** @addtogroup BSP */
* @{
  */

/** @addtogroup STM32F3_DISCOVERY */
* @{
  */

/** @addtogroup STM32F3_DISCOVERY_ACCELEROMETER */
* @{
  */

/** @addtogroup STM32F3_DISCOVERY_ACCELEROMETER_Exported_Types */
* @{
  */

/** @defgroup STM32F3_DISCOVERY_ACCELEROMETER_Exported_Types */
* @{
  */

/** */
typedef enum {
    ACCELERO_OK = 0,
    ACCELERO_ERROR = 1,
    ACCELERO_TIMEOUT = 2
} ACCELERO_StatusTypeDef;

uint8_t BSP_ACCELERO_Init(void);
void BSP_ACCELERO_Reset(void);
void BSP_ACCELERO_GetXYZ(int16_t *pData XYZ);
/**
 * @}
 */
/**
 * @}
 */
/**
 * @}
 */
/**
 * @}
 */
#ifdef __cplusplus
}
#endif
#else
#endif /*__STM32F3_DISCOVERY_ACCELER0_H */
/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
stm32f3_discovery_accelerometer.c

Go to the documentation of this file.

```c
/**
 * @file    stm32f3_discovery_accelerometer.c
 * @author  MCD Application Team
 * @brief   This file provides a set of functions needed to manage the ACCELEROMETER MEMS available on STM32F3-Discovery Kit.
 */

/**
 * @attention
 * <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>

 * Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
 * 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the fol
```
2. Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.

3. Neither the name of STMicroelectronics nor the names of its contributors
may be used to endorse or promote products derived from this software
without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
/* Includes ----------------------------------*/
#include "stm32f3_discovery_accelerometer.h"

/** @addtogroup BSP *
  @{
*/

/** @addtogroup STM32F3_DISCOVERY *
  @{
*/

/** @defgroup STM32F3_DISCOVERY_ACCELEROMETER STM32F3-DISCOVERY ACCELEROMETER *
  @{
*/

/* Private typedef ----------------------------------*/
/** @defgroup STM32F3_DISCOVERY_ACCELEROMETER_Private_Types Private Types *
  @{
*/

/** @defgroup STM32F3_DISCOVERY_ACCELEROMETER_R_Private_Types Private Types *
  @{
*/

/* Private defines ----------------------------------*/
/** @defgroup STM32F3_DISCOVERY_ACCELEROMETER_R_Private_Constants Private Constants *
  @{
*/

/** @defgroup STM32F3_DISCOVERY_ACCELEROMETER_R_Private_Defines Private Defines *
  @{
*/
* Private macros -------------------------------
----------------------------------*
/** @defgroup STM32F3_DISCOVERY_ACCELEROMETER_Private_Macros Private Macros *
* @{
* /
/**
* @} */
* /
* Private variables -----------------------
----------------------------------*
/** @defgroup STM32F3_DISCOVERY_ACCELEROMETER_Private_Variables Private Variables *
* @{
* /
static ACCELERO_DrvTypeDef *AccelerometerDrv = 0;
* /
/**
* @} */
* /
* Private function prototypes -------------
----------------------------------*
/** @addtogroup STM32F3_DISCOVERY_ACCELEROMETER_Private_FunctionPrototypes Private Functions *
* @{
* /
* @} */
* /
* Exported functions ----------------------
uint8_t BSP_ACCELERO_Init(void) {
  uint8_t ret = ACCELERO_ERROR;
  uint16_t ctrl = 0x0000;
  ACCELERO_InitTypeDef LSM303DLHC_InitStructure;
  ACCELERO_FilterConfigTypeDef LSM303DLHC_FilterStructure;
  if(Lsm303dlhcDrv.ReadID() == I_AM_LMS303DLHC) {
    /* Initialize the gyroscope driver structure */
    AccelerometerDrv = &Lsm303dlhcDrv;
  }
  /* MEMS configuration ------------------------------*/
  /* Fill the accelerometer structure */
  LSM303DLHC_InitStructure.Power_Mode = LSM303DLHC_NORMAL_MODE;
  LSM303DLHC_InitStructure.AccOutput_DataRate = LSM303DLHC_ODR_50_HZ;
  LSM303DLHC_InitStructure.Axes_Enable = LSM303DLHC_AXES_ENABLE;
  LSM303DLHC_InitStructure.AccFull_Scale = LSM303DLHC_FULLSCALE_2G;
  LSM303DLHC_InitStructure.BlockData_Update = LSM303DLHC_BlockUpdate_Continous;
  LSM303DLHC_InitStructure.Endianness=LSM303DLHC_BLE_LSB;
  LSM303DLHC_InitStructure.High_Resolution
/* Configure MEMS: data rate, power mode, full scale and axes */
ctrl |= (LSM303DLHC_InitStructure.Power_Mode | LSM303DLHC_InitStructure.AccOutput_DataRate |
LSM303DLHC_InitStructure.Axes_Enable);

/* Configure the accelerometer main parameters */
AccelerometerDrv->Init(ctrl);

/* Fill the accelerometer LPF structure */
LSM303DLHC_FilterStructure.HighPassFilter_Mode_Selection =LSM303DLHC_HPM_NORMAL_MODE;
LSM303DLHC_FilterStructure.HighPassFilter_CutOff_Frequency = LSM303DLHC_HPFCF_16;
LSM303DLHC_FilterStructure.HighPassFilter_AOI1 = LSM303DLHC_HPF_AOI1_DISABLE;
LSM303DLHC_FilterStructure.HighPassFilter_AOI2 = LSM303DLHC_HPF_AOI2_DISABLE;

/* Configure MEMS: mode, cutoff frequency, Filter status, Click, AOI1 and AOI2 */
ctrl = (uint8_t) (LSM303DLHC_FilterStructure.HighPassFilter_Mode_Selection |\
LSM303DLHC_FilterStructure.HighPassFilter_CutOff_Frequency|\
/* Configure the accelerometer LPF main parameters */
AccelerometerDrv->FilterConfig(ctrl);
ret = ACCELERO_OK;
else {
    ret = ACCELERO_ERROR;
}
return ret;
}

/**
 * @brief Reboot memory content of ACCELEROMETER
 * @retval None
 */
void BSP_ACCELERO_Reset(void)
{
    if(AccelerometerDrv->Reset != NULL)
    {
        AccelerometerDrv->Reset();
    }
}

/**
 * @brief Get XYZ acceleration
 * @param pDataXYZ Pointeur on 3 angular accelerations
 * pDataXYZ[0] = X axis, pD
00172 * @retval None
00173 */
00174 void BSP_ACCELERO_GetXYZ(int16_t *pDataXYZ)
00175 {
00176 if (AccelerometerDrv->GetXYZ != NULL)
00177 {
00178 AccelerometerDrv->GetXYZ(pDataXYZ);
00179 }
00180 }
00181
00182
00183 /**<
00184 * @@}
00185 */
00186
00187 /**<
00188 * @@}
00189 */
00190
00191 /**<
00192 * @@}
00193 */
00194
00195 /**<
00196 * @@}
00197 */
00198
00199
00200 /************************************** (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
stm32f3_discovery_gyroscope.h

Go to the documentation of this file.

```c
/**
   ******************************************
   ******************************************
   * @file     stm32f3_discovery_gyroscope.h
   * @author   MCD Application Team
   * @brief    This file contains definitions for stm32f3_discovery_gyroscope.c
   *           firmware driver.
   ******************************************
   ******************************************
   *
   * <h2><center>&copy; Copyright(c) 2016 STM Microelectronics</center></h2>
   *
   * Redistribution and use in source and binary forms, with or without modification,
   * are permitted provided that the following conditions are met:
   * 1. Redistributions of source code must retain the above copyright notice,
   * 2. Redistributions in binary form must
```
reproduce the above copyright notice,
00017  * this list of conditions and the following disclaimer in the documentation
00018  * and/or other materials provided with the distribution.
00019  * 3. Neither the name of STMicroelectronics nor the names of its contributors
00020  * may be used to endorse or promote products derived from this software
00021  * without specific prior written permission.
00022  *
00023  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
00024  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
00025  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00026  * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
00027  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
00028  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
00029  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
00030  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
00031  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00032  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00033  *
00034  ******************************************************
00035  ******************************************************
00036  */
00037
/* Define to prevent recursive inclusion */

#ifndef __STM32F3_DISCOVERY_GYRO_H
#define __STM32F3_DISCOVERY_GYRO_H

#ifdef __cplusplus
extern "C"
{

/* Includes */
#include "stm32f3_discovery.h"

/* Include Gyroscope component driver */
#include "../Components/l3gd20/l3gd20.h"

/** @addtogroup BSP *
@{ *

/** @addtogroup STM32F3_DISCOVERY *
@{ *

/** @addtogroup STM32F3_DISCOVERY_GYROSCOPE *
@{ *

/** @defgroup STM32F3_DISCOVERY_GYROSCOPE_Exported_Types Exported Types *
@{ *

typedef enum {
    GYRO_OK = 0,
    GYRO_ERROR = 1,
    GYRO_TIMEOUT = 2
}
GYRO_StatusTypeDef;

/**
 * @}
 */

/**
 * @defgroup STM32F3_DISCOVERY_GYROSCOPE_Exported_Constants Exported Constants
 * @{
 */

/**
 * @defgroup STM32F3_DISCOVERY_GYROSCOPE_Exported_Macros Exported Macros
 * @{
 */

/**
 * @defgroup STM32F3_DISCOVERY_GYROSCOPE_Exported_Functions Exported Functions
 * @{
 */

/* Sensor Configuration Functions */
uint8_t BSP_GYRO_Init(void);
void BSP_GYRO_Reset(void);
uint8_t BSP_GYRO_ReadID(void);
void BSP_GYRO_ITConfig(GYRO_INTERRUPTConfigTypeDef *pIntConfigStruct);
void BSP_GYRO_EnableIT(uint8_t IntPin);
void BSP_GYRO_DisableIT(uint8_t IntPin);
void BSP_GYRO_GetXYZ(float* pfData);

/**
 * @} 
 */

/**
 * @} 
 */

/**
 * @} 
 */

/**
 * @} 
 */

#ifdef __cplusplus
}
#endif

#endif /*__STM32F3_DISCOVERY_GYRO_H__*/

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
### stm32f3_discovery_gyroscope.c

Go to the documentation of this file.

```c
/**
  ******************************************
  ************************************
  *@file		stm32f3_discovery_gyroscope.c
  *@author	MCD Application Team
  *@brief	This file provides a set of functions needed to manage the l3gd20
          MEMS accelerometer available on STM32F3-Discovery Kit.
  ******************************************
  ************************************
  *@attention

  * <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</center></h2>

  * Redistribution and use in source and binary forms, with or without modification,
  * are permitted provided that the following conditions are met:
  * 1. Redistributions of source code must retain the above copyright notice,
  *    this list of conditions and the following disclaimer.
```
2. Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.
3. Neither the name of STMicroelectronics nor the names of its contributors
may be used to endorse or promote products derived from this software
without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
Including "stm32f3_discovery_gyroscope.h"

/**
 * @addtogroup BSP
 * @{
 *
 */

/**
 * @addtogroup STM32F3_DISCOVERY
 * @{
 *
 */

/**
 * @defgroup STM32F3_DISCOVERY_GYROSCOPE STM32F3-DISCOVERY GYROSCOPE
 *
 */

/* Private typedef ----------------------------------*/

/**
 * @defgroup STM32F3_DISCOVERY_GYROSCOPE_Private_Types Private Types
 *
 */

/**
 * @defgroup STM32F3_DISCOVERY_GYROSCOPE_Private_Constants Private Constants
 *
 */

/* Private defines ----------------------------------*/
/**
 * @defgroup STM32F3_DISCOVERY_GYROSCOPE_Private_Macros Private Macros
 */

/**
 * @defgroup STM32F3_DISCOVERY_GYROSCOPE_Private_Variables Private Variables
 */

static GYRO_DrvTypeDef *GyroscopeDrv;

/**
 * @defgroup STM32F3_DISCOVERY_GYROSCOPE_Private_FunctionPrototypes Private Functions
 */

/**
 * @defgroup STM32F3_DISCOVERY_GYROSCOPE_Exported_Functions Exported Functions
 */

*/
/**
 * @brief  Set GYROSCOPE Initialization.
 * @retval GYRO_OK if no problem during initialization
 */

uint8_t BSP_GYRO_Init(void)
{
    uint8_t ret = GYRO_ERROR;
    uint16_t ctrl = 0x0000;
    GYRO_InitTypeDef L3GD20_InitStructure;
    GYRO_FilterConfigTypeDef L3GD20_FilterStructure;

    if((L3gd20Drv.ReadID() == I_AM_L3GD20) ||
        (L3gd20Drv.ReadID() == I_AM_L3GD20_TR))
    {
        /* Initialize the gyroscope driver structure */
        GyroscopeDrv = &L3gd20Drv;

        /* Configure Mems : data rate, power mode, full scale and axes */
        L3GD20_InitStructure.Power_Mode = L3GD20_MODE_ACTIVE;
        L3GD20_InitStructure.Output_DataRate = L3GD20_OUTPUT_DATARATE_1;
        L3GD20_InitStructure.Axes_Enable = L3GD20_AXES_ENABLE;
        L3GD20_InitStructure.Band_Width = L3GD20_BANDWIDTH_4;
        L3GD20_InitStructure.BlockData_Update = L3GD20_BlockDataUpdate_Continous;
        L3GD20_InitStructure.Endianness = L3GD20_BLE_LSB;
        L3GD20_InitStructure.Full_Scale = L3GD20...
/* Configure MEMS: data rate, power mode, full scale and axes */
ctrl = (uint16_t) (L3GD20_InitStructure.Power_Mode | L3GD20_InitStructure.Output_DataRate |
L3GD20_InitStructure.Axes_Enable | L3GD20_InitStructure.Band_Width);
ctrl |= (uint16_t) ((L3GD20_InitStructure.BlockData_Update | L3GD20_InitStructure.Endianness |
L3GD20_InitStructure.Full_Scale) << 8);
GyroscopeDrv->Init(ctrl);
L3GD20_Filt{}
else {
    ret = GYRO_ERROR;
}

return ret;

/**
 * @brief Read ID of Gyroscope component
 * @retval ID
 */

uint8_t BSP_GYRO_ReadID(void)
{
    uint8_t id = 0x00;

    if(GyroscopeDrv->ReadID != NULL)
    {
        id = GyroscopeDrv->ReadID();
    }

    return id;
}

/**
 * @brief Reboot memory content of GYROSCOPE
 * @retval None
 */

void BSP_GYRO_Reset(void)
{
    if(GyroscopeDrv->Reset != NULL)
    {
        GyroscopeDrv->Reset();
    }
}
/**
 * @brief Configure INT1 interrupt
 * @param pIntConfig pointer to a L3GD20_INTERRUPTConfig_TypeDef structure that contains the configuration setting for the L3GD20 Interrupt.
 * @retval None
 */

void BSP_GYRO_ITConfig(GYRO_INTERRUPTConfigTypeDef *pIntConfig)
{
    uint16_t interruptconfig = 0x0000;

    if (GyroscopeDrv->ConfigIT != NULL)
    {
        /* Configure latch Interrupt request and axe interrupts */
        interruptconfig |= ((uint8_t)(pIntConfig->Latch_Request) |
            pIntConfig->Interrupt_Axes) << 8);
        interruptconfig |= (uint8_t)(pIntConfig->Interrupt_ActiveEdge);

        GyroscopeDrv->ConfigIT(interruptconfig);
    }
}

/**
 * @brief Enable INT1 or INT2 interrupt
 * @param IntPin Interrupt pin
 * @retval None
 */

void BSP_GYRO_ITEnable(GYRO_Pin_TypeDef *IntPin)
{
    switch (IntPin)
    {
        case L3GD20_INT1:
            /* Configure Interrupt INT1 */
            break;
        case L3GD20_INT2:
            /* Configure Interrupt INT2 */
            break;
    }
}

void BSP_GYRO_EnableIT(uint8_t IntPin) {
    if(GyroscopeDrv->EnableIT != NULL) {
        GyroscopeDrv->EnableIT(IntPin);
    }
}

/**
 * @brief Disable INT1 or INT2 interrupt
 * @param IntPin Interrupt pin
 *       This parameter can be:
 *       @arg L3GD20_INT1
 *       @arg L3GD20_INT2
 * @retval None
 */
void BSP_GYRO_DisableIT(uint8_t IntPin) {
    if(GyroscopeDrv->DisableIT != NULL) {
        GyroscopeDrv->DisableIT(IntPin);
    }
}

/**
 * @brief Get XYZ angular acceleration
 * @param pfData pointer on floating array
 * @retval None
 */
void BSP_GYRO_GetXYZ(float* pfData) {
    if(GyroscopeDrv->GetXYZ != NULL) {
        GyroscopeDrv->GetXYZ(pfData);
    }
}
} */

/***/ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
# STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BSP</strong></td>
<td>Modules</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Modules

STM32F3-DISCOVERY

This file provides set of firmware functions to manage Leds and push-button available on STM32F3-Discovery Kit from STMicroelectronics.
STM32F3-Discovery BSP User Manual

STM32F3-DISCOVERY

STM32F3-DISCOVERY BSP

This file provides set of firmware functions to manage Leds and push-button available on STM32F3-Discovery Kit from STMicroelectronics. More...
# Modules

<table>
<thead>
<tr>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F3-DISCOVERY Common</td>
</tr>
<tr>
<td>STM32F3-DISCOVERY ACCELEROMETER</td>
</tr>
<tr>
<td>STM32F3-DISCOVERY GYROSCOPE</td>
</tr>
</tbody>
</table>
Detailed Description

This file provides set of firmware functions to manage Leds and push-button available on STM32F3-Discovery Kit from STMicroelectronics.
<table>
<thead>
<tr>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F3-DISCOVERY</td>
<td>Common</td>
<td></td>
</tr>
</tbody>
</table>

STM32F3-DISCOVERY
**Modules**

<table>
<thead>
<tr>
<th>Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus Operation functions</td>
</tr>
<tr>
<td>Link Operation functions</td>
</tr>
<tr>
<td>Private Constants</td>
</tr>
<tr>
<td>Private Variables</td>
</tr>
<tr>
<td>Exported Types</td>
</tr>
<tr>
<td>Exported Constants</td>
</tr>
<tr>
<td>Exported Functions</td>
</tr>
</tbody>
</table>

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP
User Manual by doxygen 1.7.6.1
### STM32F3-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Modules</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Exported Constants

STM32F3-DISCOVERY Common
### Modules

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F3-DISCOVERY LED</td>
<td>Define for STM32F3_DISCOVERY board.</td>
</tr>
<tr>
<td>STM32F3-DISCOVERY BUTTON</td>
<td></td>
</tr>
<tr>
<td>STM32F3-DISCOVERY COM</td>
<td></td>
</tr>
<tr>
<td>STM32F3-DISCOVERY COMPONENT</td>
<td></td>
</tr>
</tbody>
</table>

Generated on Wed May 31 2017 10:47:41 for STM32F3-Discovery BSP User Manual by doxygen 1.7.6.1