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

- __STM32F072B_DISCO_BSP_VERSION :
  `stm32f072b_discovery.c`
- __STM32F072B_DISCO_BSP_VERSION_MAIN :
  `stm32f072b_discovery.c`
- __STM32F072B_DISCO_BSP_VERSION_RC :
  `stm32f072b_discovery.c`
- __STM32F072B_DISCO_BSP_VERSION_SUB1 :
  `stm32f072b_discovery.c`
- __STM32F072B_DISCO_BSP_VERSION_SUB2 :
  `stm32f072b_discovery.c`

- BSP_EEPROM_Init() :
  `stm32f072b_discovery_eeprom.c`
- BSP_EEPROM_ReadBuffer() :
  `stm32f072b_discovery_eeprom.c`
- BSP_EEPROM_TIMEOUT_UserCallback() :
  `stm32f072b_discovery_eeprom.c`
- BSP_EEPROM_WaitEepromStandbyState() :
  `stm32f072b_discovery_eeprom.c`
- BSP_EEPROM_WriteBuffer() :
  `stm32f072b_discovery_eeprom.c`
- BSP_EEPROM_WritePage() :
  `stm32f072b_discovery_eeprom.c`
- BSP_GetVersion() :
  `stm32f072b_discovery.c`
- d -

- DISCOVERY_EEPROM_DMA : stm32f072b_discovery.h
- DISCOVERY_EEPROM_DMA_CHANNEL_RX : stm32f072b_discovery.h
- DISCOVERY_EEPROM_DMA_CHANNEL_TX : stm32f072b_discovery.h
- DISCOVERY_EEPROM_DMA_CLK_DISABLE : stm32f072b_discovery.h
- DISCOVERY_EEPROM_DMA_CLK_ENABLE : stm32f072b_discovery.h
- DISCOVERY_EEPROM_DMA_PREPRIO :
stm32f072b_discovery.h

- DISCOVERY_EEPROM_DMA_RX_IRQn : I2C.Instance
- DISCOVERY_EEPROM_DMA_SUBPRIO : I2C.Instance
- DISCOVERY_EEPROM_DMA_TX_IRQn : I2C.Instance
- DISCOVERY_EEPROM_I2C_ADDRESS_A01 : I2C.Instance
- DISCOVERY_I2Cx : I2C.Instance
- DISCOVERY_I2Cx_AF : I2C.Instance
- DISCOVERY_I2Cx_CLK_DISABLE : I2C.Instance
- DISCOVERY_I2Cx_CLK_ENABLE : I2C.Instance
- DISCOVERY_I2Cx_FORCE_RESET : I2C.Instance
- DISCOVERY_I2Cx_GPIO_CLK_DISABLE : I2C.Instance
- DISCOVERY_I2Cx_GPIO_CLK_ENABLE : I2C.Instance
- DISCOVERY_I2Cx_GPIO_PORT : I2C.Instance
- DISCOVERY_I2Cx_SCL_PIN : I2C.Instance
- DISCOVERY_I2Cx_SDA_PIN : I2C.Instance
- DISCOVERY_I2Cx_TIMING : I2C.Instance
- DISCOVERY_SPIx : SPI.Instance
- DISCOVERY_SPIx_AF : SPI.Instance
- DISCOVERY_SPIx_CLOCK_ENABLE : SPI.Instance
- DISCOVERY_SPIx_GPIO_CLK_DISABLE : SPI.Instance
- DISCOVERY_SPIx_GPIO_CLK_ENABLE : SPI.Instance
- DISCOVERY_SPIx_GPIO_PORT : SPI.Instance
- DISCOVERY_SPIx_MISO_PIN : SPI.Instance
- DISCOVERY_SPIx_MOSI_PIN : SPI.Instance
- DISCOVERY_SPIx_SCK_PIN : SPI.Instance
- DUMMY_BYTE : I2C.Instance
- e -

- EEPROM_FAIL : stm32f072b_discovery_eeprom.h
- EEPROM_IO_Init() : stm32f072b_discovery.c ,
  stm32f072b_discovery_eeprom.h
- EEPROM_IO_IsDeviceReady() : stm32f072b_discovery.c ,
  stm32f072b_discovery_eeprom.h
- EEPROM_IO_ReadData() : stm32f072b_discovery.c ,
  stm32f072b_discovery_eeprom.h
- EEPROM_IO_WriteData() : stm32f072b_discovery.c ,
  stm32f072b_discovery_eeprom.h
- EEPROM_LONG_TIMEOUT : stm32f072b_discovery_eeprom.h
- EEPROM_MAX_SIZE : stm32f072b_discovery_eeprom.h
- EEPROM_MAX_TRIALS_NUMBER : 
  stm32f072b_discovery_eeprom.h
- EEPROM_OK : stm32f072b_discovery_eeprom.h
- EEPROM_PAGESIZE : stm32f072b_discovery_eeprom.h
- EEPROM_TIMEOUT : stm32f072b_discovery_eeprom.h
- EEPROMAddress : stm32f072b_discovery_eeprom.c
- EEPROMDataNum : stm32f072b_discovery_eeprom.c
- EEPROMDataRead : stm32f072b_discovery_eeprom.c
- EEPROMDataWritePointer : stm32f072b_discovery_eeprom.c
- EEPROMTimeout : stm32f072b_discovery_eeprom.c

- g -

- GYRO_CS_GPIO_CLK_DISABLE : stm32f072b_discovery.h
- GYRO_CS_GPIO_CLK_ENABLE : stm32f072b_discovery.h
- GYRO_CS_GPIO_PORT : stm32f072b_discovery.h
- GYRO_CS_HIGH : stm32f072b_discovery.h
- GYRO_CS_LOW : stm32f072b_discovery.h
- GYRO_CS_PIN : stm32f072b_discovery.h
- GYRO_ERROR : stm32f072b_discovery_gyroscope.h
- GYRO_INT1_EXTI_IRQHandler : stm32f072b_discovery.h
- GYRO_INT1_PIN : stm32f072b_discovery.h
- GYRO_INT2_EXTI_IRQHandler : stm32f072b_discovery.h
- GYRO_INT2_PIN : stm32f072b_discovery.h
- GYRO_INT_GPIO_CLK_DISABLE : stm32f072b_discovery.h
- h -

- i -

- l -
- m -
  
  • MULTIPLEBYTE_CMD : stm32f072b_discovery.h

- r -
  
  • READWRITE_CMD : stm32f072b_discovery.h

- s -
  
  • SpiHandle : stm32f072b_discovery.c
  • SPIx_Error() : stm32f072b_discovery.c
  • SPIx_Init() : stm32f072b_discovery.c
  • SPIx_MspInit() : stm32f072b_discovery.c
- SPIx_TIMEOUT_MAX : stm32f072b_discovery.h
- SPIx_WriteRead() : stm32f072b_discovery.c
- SpixTimeout : stm32f072b_discovery.c

- USER_BUTTON_EXTI_IRQn : stm32f072b_discovery.h
- USER_BUTTON_GPIO_CLK_DISABLE : stm32f072b_discovery.h
- USER_BUTTON_GPIO_CLK_ENABLE : stm32f072b_discovery.h
- USER_BUTTON_GPIO_PORT : stm32f072b_discovery.h
- USER_BUTTON_PIN : stm32f072b_discovery.h
STM32F072B-Discovery BSP User Manual

- b -

- BSP_EEPROM_Init() : stm32f072b_discovery_eeprom.c
- BSP_EEPROM_ReadBuffer() : stm32f072b_discovery_eeprom.c
- BSP_EEPROM_TIMEOUT_UserCallback() :
  stm32f072b_discovery_eeprom.c
- BSP_EEPROM_WaitEepromStandbyState() :
  stm32f072b_discovery_eeprom.c
- BSP_EEPROM_WriteBuffer() : stm32f072b_discovery_eeprom.c
- BSP_EEPROM_WritePage() : stm32f072b_discovery_eeprom.c
- BSP_GetVersion() : stm32f072b_discovery.c
- BSP_GYRO_DisableIT() : stm32f072b_discovery_gyroscope.c
- BSP_GYRO_EnableIT() : stm32f072b_discovery_gyroscope.c
- BSP_GYRO_GetXYZ() : stm32f072b_discovery_gyroscope.c
- BSP_GYRO_Init() : stm32f072b_discovery_gyroscope.c
- BSP_GYRO_ITConfig() : stm32f072b_discovery_gyroscope.c
- BSP_GYRO_ReadID() : stm32f072b_discovery_gyroscope.c
- BSP_GYRO_Reset() : stm32f072b_discovery_gyroscope.c
- BSP_LED_Init() : stm32f072b_discovery.c
- BSP_LED_Off() : stm32f072b_discovery.c
- BSP_LED_On() : stm32f072b_discovery.c
- BSP_LED_Toggle() : stm32f072b_discovery.c
- BSP_PB_GetState() : stm32f072b_discovery.c
- BSP_PB_Init() : stm32f072b_discovery.c
- e -

- EEPROM_IO_Init() : stm32f072b_discovery.c, stm32f072b_discovery_eeprom.h
- EEPROM_IO_IsDeviceReady() : stm32f072b_discovery_eeprom.h, stm32f072b_discovery.c
- EEPROM_IO_ReadData() : stm32f072b_discovery.c, stm32f072b_discovery_eeprom.h
- EEPROM_IO_WriteData() : stm32f072b_discovery_eeprom.h, stm32f072b_discovery.c

- g -

- GYRO_IO_Init() : stm32f072b_discovery.c
- GYRO_IO_Read() : stm32f072b_discovery.c
- GYRO_IO_Write() : stm32f072b_discovery.c

- h -

- HAL_I2C_MemRxCpltCallback() : stm32f072b_discovery_eeprom.c
- HAL_I2C_MemTxCpltCallback() : stm32f072b_discovery_eeprom.c

- i -

- I2Cx_Error() : stm32f072b_discovery.c
- I2Cx_Init() : stm32f072b_discovery.c
- I2Cx_MspInit() : stm32f072b_discovery.c
- I2Cx_ReadBuffer() : stm32f072b_discovery.c
- I2Cx_ReadData() : stm32f072b_discovery.c
- I2Cx_WriteBuffer() : stm32f072b_discovery.c
- I2Cx_WriteData() : stm32f072b_discovery.c

- s -

- SPIx_Error() : stm32f072b_discovery.c
- SPIx_Init() : stm32f072b_discovery.c
- SPIx_MspInit() : stm32f072b_discovery.c
• SPIx_WriteRead() : stm32f072b_discovery.c

Generated on Wed Jul 5 2017 09:43:20 for STM32F072B-Discovery
BSP User Manual by doxygen 1.7.6.1
STM32F072B-Discovery BSP User Manual

- BUTTON_IRQHandler: `stm32f072b_discovery.c`
- BUTTON_PIN: `stm32f072b_discovery.c`
- BUTTON_PORT: `stm32f072b_discovery.c`
- EEPROMAddress: `stm32f072b_discovery_eeprom.c`
- EEPROMDataNum: `stm32f072b_discovery_eeprom.c`
- EEPROMDataRead: `stm32f072b_discovery_eeprom.c`
- EEPROMDataWritePointer: `stm32f072b_discovery_eeprom.c`
- EEPROMTimeout: `stm32f072b_discovery_eeprom.c`
- GyroscopeDrv: `stm32f072b_discovery_gyroscope.c`
- I2cHandle: `stm32f072b_discovery.c`
- LED_PIN: `stm32f072b_discovery.c`
- LED_PORT: `stm32f072b_discovery.c`
- SpiHandle: `stm32f072b_discovery.c`
- SpixTimeout: `stm32f072b_discovery.c`

Generated on Wed Jul 5 2017 09:43:20 for STM32F072B-Discovery BSP User Manual by doxygen 1.7.6.1
### File List

<table>
<thead>
<tr>
<th>All</th>
<th>Functions</th>
<th>Variables</th>
<th>Enumerations</th>
<th>Enumerator</th>
<th>Defines</th>
</tr>
</thead>
</table>

#### Enumerations

- **Button_TypeDef** : `stm32f072b_discovery.h`
- **ButtonMode_TypeDef** : `stm32f072b_discovery.h`
- **GYRO_StatusTypeDef** : `stm32f072b_discovery_gyroscope.h`
- **Led_TypeDef** : `stm32f072b_discovery.h`

---

*Generated on Wed Jul 5 2017 09:43:20 for STM32F072B-Discovery BSP User Manual by [doxygen](http://www.doxygen.org) 1.7.6.1*
STM32F072B-Discovery BSP User Manual

- BUTTON_MODE_EXTI : stm32f072b_discovery.h
- BUTTON_MODE_GPIO : stm32f072b_discovery.h
- BUTTON_USER : stm32f072b_discovery.h
- GYRO_ERROR : stm32f072b_discovery_gyroscope.h
- GYRO_OK : stm32f072b_discovery_gyroscope.h
- GYRO_TIMEOUT : stm32f072b_discovery_gyroscope.h
- LED3 : stm32f072b_discovery.h
- LED4 : stm32f072b_discovery.h
- LED5 : stm32f072b_discovery.h
- LED6 : stm32f072b_discovery.h
- LED_BLUE : stm32f072b_discovery.h
- LED_GREEN : stm32f072b_discovery.h
- LED_ORANGE : stm32f072b_discovery.h
- LED_RED : stm32f072b_discovery.h

Generated on Wed Jul 5 2017 09:43:20 for STM32F072B-Discovery BSP User Manual by doxygen 1.7.6.1
STM32F072B-Discovery BSP User Manual

- _ _ -

  - __STM32F072B_DISCO_BSP_VERSION : 
    stm32f072b_discovery.c
  - __STM32F072B_DISCO_BSP_VERSION_MAIN : 
    stm32f072b_discovery.c
  - __STM32F072B_DISCO_BSP_VERSION_RC : 
    stm32f072b_discovery.c
  - __STM32F072B_DISCO_BSP_VERSION_SUB1 : 
    stm32f072b_discovery.c
  - __STM32F072B_DISCO_BSP_VERSION_SUB2 : 
    stm32f072b_discovery.c

- b -

  - BUTTONn : stm32f072b_discovery.h
  - BUTTONx_GPIO_CLK_DISABLE : stm32f072b_discovery.h
  - BUTTONx_GPIO_CLK_ENABLE : stm32f072b_discovery.h

- d -

  - DISCOVERY_EEPROM_DMA : stm32f072b_discovery.h
  - DISCOVERY_EEPROM_DMA_CHANNEL_RX : 
    stm32f072b_discovery.h
  - DISCOVERY_EEPROM_DMA_CHANNEL_TX : 

stm32f072b_discovery.h

- DISCOVERY_EEPROM_DMA_CLK_DISABLE : stm32f072b_discovery.h
- DISCOVERY_EEPROM_DMA_CLK_ENABLE : stm32f072b_discovery.h
- DISCOVERY_EEPROM_DMA_PREPRIO : stm32f072b_discovery.h
- DISCOVERY_EEPROM_DMA_RX_IRQn : stm32f072b_discovery.h
- DISCOVERY_EEPROM_DMA_SUBPrio : stm32f072b_discovery.h
- DISCOVERY_EEPROM_DMA_TX_IRQn : stm32f072b_discovery.h
- DISCOVERY_EEPROM_I2C_ADDRESS_A01 : stm32f072b_discovery.h
- DISCOVERY_I2Cx : stm32f072b_discovery.h
- DISCOVERY_I2Cx_AF : stm32f072b_discovery.h
- DISCOVERY_I2Cx_CLK_DISABLE : stm32f072b_discovery.h
- DISCOVERY_I2Cx_CLK_ENABLE : stm32f072b_discovery.h
- DISCOVERY_I2Cx_FORCE_RESET : stm32f072b_discovery.h
- DISCOVERY_I2Cx_GPIO_CLK_DISABLE : stm32f072b_discovery.h
- DISCOVERY_I2Cx_GPIO_CLK_ENABLE : stm32f072b_discovery.h
- DISCOVERY_I2Cx_GPIO_PORT : stm32f072b_discovery.h
- DISCOVERY_I2Cx_RELEASE_RESET : stm32f072b_discovery.h
- DISCOVERY_I2Cx_SCL_PIN : stm32f072b_discovery.h
- DISCOVERY_I2Cx_SDA_PIN : stm32f072b_discovery.h
- DISCOVERY_I2Cx_TIMING : stm32f072b_discovery.h
- DISCOVERY_SPIx : stm32f072b_discovery.h
- DISCOVERY_SPIx_AF : stm32f072b_discovery.h
- DISCOVERY_SPIx_CLOCK_ENABLE : stm32f072b_discovery.h
- DISCOVERY_SPIx_GPIO_CLK_DISABLE : stm32f072b_discovery.h
- DISCOVERY_SPIx_GPIO_CLK_ENABLE : stm32f072b_discovery.h
• DISCOVERY_SPIx_GPIO_PORT: stm32f072b_discovery.h
• DISCOVERY_SPIx_MISO_PIN: stm32f072b_discovery.h
• DISCOVERY_SPIx_MOSI_PIN: stm32f072b_discovery.h
• DISCOVERY_SPIx_SCK_PIN: stm32f072b_discovery.h
• DUMMY_BYTE: stm32f072b_discovery.h

- e -

• EEPROM_FAIL: stm32f072b_discovery_eeprom.h
• EEPROM_LONG_TIMEOUT: stm32f072b_discovery_eeprom.h
• EEPROM_MAX_SIZE: stm32f072b_discovery_eeprom.h
• EEPROM_MAX_TRIALS_NUMBER: stm32f072b_discovery_eeprom.h
• EEPROM_OK: stm32f072b_discovery_eeprom.h
• EEPROM_PAGESIZE: stm32f072b_discovery_eeprom.h
• EEPROM_TIMEOUT: stm32f072b_discovery_eeprom.h

- g -

• GYRO_CS_GPIO_CLK_DISABLE: stm32f072b_discovery.h
• GYRO_CS_GPIO_CLK_ENABLE: stm32f072b_discovery.h
• GYRO_CS_GPIO_PORT: stm32f072b_discovery.h
• GYRO_CS_HIGH: stm32f072b_discovery.h
• GYRO_CS_LOW: stm32f072b_discovery.h
• GYRO_CS_PIN: stm32f072b_discovery.h
• GYRO_INT1_EXTI_IRQn: stm32f072b_discovery.h
• GYRO_INT1_PIN: stm32f072b_discovery.h
• GYRO_INT2_EXTI_IRQn: stm32f072b_discovery.h
• GYRO_INT2_PIN: stm32f072b_discovery.h
• GYRO_INT_GPIO_CLK_DISABLE: stm32f072b_discovery.h
• GYRO_INT_GPIO_CLK_ENABLE: stm32f072b_discovery.h
• GYRO_INT_GPIO_PORT: stm32f072b_discovery.h

- i -

• I2Cx_TIMEOUT_MAX: stm32f072b_discovery.h

- l -
- m -
  • MULTIPLEBYTE_CMD : stm32f072b_discovery.h

- r -
  • READWRITE_CMD : stm32f072b_discovery.h

- s -
  • SPIx_TIMEOUT_MAX : stm32f072b_discovery.h

- u -
  • USER_BUTTON_EXTI_IRQn : stm32f072b_discovery.h
  • USER_BUTTON_GPIO_CLK_DISABLE : stm32f072b_discovery.h
  • USER_BUTTON_GPIO_CLK_ENABLE :
stm32f072b_discovery.h

- USER_BUTTON_GPIO_PORT : \texttt{stm32f072b\_discovery.h}
- USER_BUTTON_PIN : \texttt{stm32f072b\_discovery.h}

Generated on Wed Jul 5 2017 09:43:20 for STM32F072B-Discovery
BSP User Manual by \texttt{doxygen} 1.7.6.1
STM32F072B-Discovery BSP User Manual

stm32f072b_discovery.c File Reference

This file provides set of firmware functions to manage Leds, push-button available on STM32F072B-Discovery Kit from STMicroelectronics. More...

#include "stm32f072b_discovery.h"

Go to the source code of this file.
Defines

```
#define __STM32F072B_DISCO_BSP_VERSION_MAIN (0x02)
STM32F072B DISCO BSP Driver version number V2.1.6.
#define __STM32F072B_DISCO_BSP_VERSION_SUB1 (0x01)
#define __STM32F072B_DISCO_BSP_VERSION_SUB2 (0x06)
#define __STM32F072B_DISCO_BSP_VERSION_RC (0x00)
#define __STM32F072B_DISCO_BSP_VERSION
```
## Functions

<table>
<thead>
<tr>
<th>Function (Type, Name)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td><strong>I2Cx_Init</strong> (void)</td>
</tr>
<tr>
<td></td>
<td>Discovery I2Cx Bus initialization.</td>
</tr>
<tr>
<td>static uint32_t</td>
<td><strong>I2Cx_WriteData</strong> (uint8_t Addr, uint16_t Reg, uint8_t Value)</td>
</tr>
<tr>
<td></td>
<td>Write a value in a register of the device through BUS.</td>
</tr>
<tr>
<td>static uint32_t</td>
<td><strong>I2Cx_WriteBuffer</strong> (uint8_t Addr, uint16_t Reg, uint8_t *pBuffer, uint16_t Length)</td>
</tr>
<tr>
<td></td>
<td>Write a value in a register of the device through BUS.</td>
</tr>
<tr>
<td>static uint32_t</td>
<td><strong>I2Cx_ReadData</strong> (uint8_t Addr, uint16_t Reg, uint8_t *Value)</td>
</tr>
<tr>
<td></td>
<td>Read a register of the device through BUS.</td>
</tr>
<tr>
<td>static uint32_t</td>
<td><strong>I2Cx_ReadBuffer</strong> (uint8_t Addr, uint16_t Reg, uint8_t *pBuffer, uint16_t Length)</td>
</tr>
<tr>
<td></td>
<td>Reads multiple data on the BUS.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>I2Cx_Error</strong> (uint8_t Addr)</td>
</tr>
<tr>
<td></td>
<td>Discovery I2Cx error treatment function.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>SPIx_Init</strong> (void)</td>
</tr>
<tr>
<td></td>
<td>SPI1 Bus initialization.</td>
</tr>
<tr>
<td>static uint8_t</td>
<td><strong>SPIx_WriteRead</strong> (uint8_t Byte)</td>
</tr>
<tr>
<td></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>
</tr>
<tr>
<td></td>
<td>SPI1 error treatment function.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>SPIx_MspInit</strong> (SPI_HandleTypeDef *hspi)</td>
</tr>
<tr>
<td></td>
<td>SPI MSP Init.</td>
</tr>
<tr>
<td>void</td>
<td><strong>GYRO_IO_Init</strong> (void)</td>
</tr>
<tr>
<td></td>
<td>Configures the GYRO SPI interface.</td>
</tr>
<tr>
<td>void</td>
<td><strong>GYRO_IO_Write</strong> (uint8_t *pBuffer, uint8_t WriteAddr, uint16_t NumByteToWrite)</td>
</tr>
</tbody>
</table>
voidGYRO_IO_Read (uint8_t *pBuffer, uint8_t ReadAddr, uint16_t NumByteToRead)
Reads a block of data from the GYROSCOPE.

voidEEPROM_IO_Init (void)
Initializes peripherals used by the I2C EEPROM driver.

uint32_tEEPROM_IO_WriteData (uint16_t DevAddress, uint16_t MemAddress, uint32_t pBuffer, uint32_t BufferSize)
Write data to I2C EEPROM driver in using DMA channel.

uint32_tEEPROM_IO_ReadData (uint16_t DevAddress, uint16_t MemAddress, uint32_t pBuffer, uint32_t BufferSize)
Read data from I2C EEPROM driver in using DMA channel.

HAL_StatusTypeDefEEPROM_IO_IsDeviceReady (uint16_t DevAddress, uint32_t Trials)
Checks if target device is ready for communication.

uint32_tBSP_GetVersion (void)
This method returns the STM32F072B DISCO BSP Driver revision.

voidBSP_LED_Init (Led_TypeDef Led)
Configures LED GPIO.

voidBSP_LED_On (Led_TypeDef Led)
Turns selected LED On.

voidBSP_LED_Off (Led_TypeDef Led)
Turns selected LED Off.

voidBSP_LED_Toggle (Led_TypeDef Led)
Toggles the selected LED.

voidBSP_PB_Init (Button_TypeDef Button, ButtonMode_TypeDef Mode)
uint32_t   **BSP_PB_GetState** *(Button_TypeDef Button)*
Returns the selected Push Button state.

static void **I2Cx_MspInit** *(I2C_HandleTypeDef *hi2c)*
Discovery I2Cx MSP Initialization.
## Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIO_TypeDef *</td>
<td>LED_PORT [LEDn]</td>
<td></td>
</tr>
<tr>
<td>const uint16_t</td>
<td>LED_PIN [LEDn]</td>
<td></td>
</tr>
<tr>
<td>GPIO_TypeDef *</td>
<td>BUTTON_PORT [BUTTONn] = (USER_BUTTON_GPIO_PORT)</td>
<td></td>
</tr>
<tr>
<td>const uint16_t</td>
<td>BUTTON_PIN [BUTTONn] = (USER_BUTTON_PIN)</td>
<td></td>
</tr>
<tr>
<td>const uint8_t</td>
<td>BUTTON_IRQn [BUTTONn] = (USER_BUTTON_EXTI_IRQn)</td>
<td></td>
</tr>
<tr>
<td>I2C_HandleTypeDef</td>
<td>I2cHandle</td>
<td>BUS variables.</td>
</tr>
<tr>
<td>uint32_t</td>
<td>SpixTimeout = SPIx_TIMEOUT_MAX</td>
<td></td>
</tr>
<tr>
<td>static SPI_HandleTypeDef</td>
<td>SpiHandle</td>
<td></td>
</tr>
</tbody>
</table>
Detailed Description

This file provides set of firmware functions to manage Leds, push-button available on STM32F072B-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 stm32f072b_discovery.c.
This file contains definitions for STM32F072B-Discovery's Leds, push-buttons hardware resources. More...

#include "stm32f0xx_hal.h"

Go to the source code of this file.
#define LEDn 4
#define LED3_PIN  GPIO_PIN_6
#define LED3_GPIO_PORT  GPIOC
#define LED3_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define LED3_GPIO_CLK_DISABLE() __HAL_RCC_GPIOC_CLK_DISABLE()
#define LED3_GPIO_CLK_ENABLE(__LED__) __HAL_RCC_GPIOC_CLK_ENABLE()
#define LED3_GPIO_CLK_DISABLE(__LED__) __HAL_RCC_GPIOC_CLK_DISABLE()
#define LED4_PIN  GPIO_PIN_8
#define LED4_GPIO_PORT  GPIOC
#define LED4_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define LED4_GPIO_CLK_DISABLE() __HAL_RCC_GPIOC_CLK_DISABLE()
#define LED4_GPIO_CLK_ENABLE(__BUTTON__) do {
    if((__BUTTON__) == BUTTON_USER) USER_BUTTON_GPIO_CLK_ENABLE();
} while(0)
#define LED4_GPIO_CLK_DISABLE(__BUTTON__) (((__BUTTON__) == BUTTON_USER) ? USER_BUTTON_GPIO_CLK_DISABLE() : 0)
#define LED5_PIN  GPIO_PIN_9
#define LED5_GPIO_PORT  GPIOC
#define LED5_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define LED5_GPIO_CLK_DISABLE() __HAL_RCC_GPIOC_CLK_DISABLE()
#define LED5_GPIO_CLK_ENABLE(__LED__) __HAL_RCC_GPIOC_CLK_ENABLE()
#define LED5_GPIO_CLK_DISABLE(__LED__) __HAL_RCC_GPIOC_CLK_DISABLE()
#define LED6_PIN  GPIO_PIN_7
#define LED6_GPIO_PORT  GPIOC
#define LED6_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define LED6_GPIO_CLK_DISABLE() __HAL_RCC_GPIOC_CLK_DISABLE()
#define LED6_GPIO_CLK_ENABLE(__LED__) __HAL_RCC_GPIOC_CLK_ENABLE()
#define LED6_GPIO_CLK_DISABLE(__LED__) __HAL_RCC_GPIOC_CLK_DISABLE()
#define USER_BUTTON_PIN  GPIO_PIN_0 /* PA0 */
#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_1_IRQHandler
#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 SPI2
#define DISCOVERY_SPIx_CLOCK_ENABLE() __HAL_RCC_SPI2_CLK_ENABLE()
#define DISCOVERY_SPIx_GPIO_PORT GPIOB /* GPIOB */
#define DISCOVERY_SPIx_AF GPIO_AF0_SPI2
#define DISCOVERY_SPIx_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_ENABLE()
#define DISCOVERY_SPIx_GPIO_CLK_DISABLE() __HAL_RCC_GPIOB_CLK_DISABLE()
#define DISCOVERY_SPIx_SCK_PIN GPIO_PIN_13 /* PB.13 */
#define DISCOVERY_SPIx_MISO_PIN GPIO_PIN_14 /* PB.14 */
#define DISCOVERY_SPIx_MOSI_PIN GPIO_PIN_15 /* PB.15 */
#define SPIx_TIMEOUT_MAX ((uint32_t)0x1000)
#define DISCOVERY_I2Cx I2C2
  Definition for I2C Interface pins (I2C2 used)
#define DISCOVERY_I2Cx_CLOCK_ENABLE() __HAL_RCC_I2C2_CLK_ENABLE()
#define DISCOVERY_I2Cx_CLOCK_DISABLE() __HAL_RCC_I2C2_CLK_DISABLE()
#define DISCOVERY_I2Cx_FORCE_RESET() __HAL_RCC_I2C2_FORCE_RESET()
#define DISCOVERY_I2Cx_RELEASE_RESET() __HAL_RCC_I2C2_RELEASE_RESET()
#define DISCOVERY_I2Cx_SCL_PIN GPIO_PIN_10 /* PB.10 */
#define DISCOVERY_I2Cx_SDA_PIN GPIO_PIN_11 /* PB.11 */
#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_AF1_I2C2
#define I2Cx_TIMEOUT_MAX ((uint32_t)0x10000)
#define DISCOVERY_I2Cx_TIMING 0x40B32537
#define DISCOVERY_EEPROM_I2C_ADDRESS_A01 0xA0
#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 GPIOC /* GPIOC */
  GYRO SPI Interface pins.
#define GYRO_CS_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define GYRO_CS_GPIO_CLK_DISABLE() __HAL_RCC_GPIOC_CLK_DISABLE()
#define GYRO_CS_PIN GPIO_PIN_0 /* PC.00 */
#define GYRO_INT_GPIO_PORT GPIOC /* GPIOC */
#define GYRO_INT_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define GYRO_INT_GPIO_CLK_DISABLE() __HAL_RCC_GPIOC_CLK_DISABLE()
#define GYRO_INT1_PIN GPIO_PIN_1 /* PC.01 */
#define GYRO_INT1_EXTI_IRQn EXTI0_1_IRQn
#define GYRO_INT2_PIN GPIO_PIN_2 /* PC.02 */
#define GYRO_INT2_EXTI_IRQn EXTI2_3_IRQn
#define DISCOVERY_EEPROM_DMA DMA1
I2C EEPROM Interface pins.
#define DISCOVERY_EEPROM_DMA_CHANNEL_TX DMA1_Channel4
#define DISCOVERY_EEPROM_DMA_CHANNEL_RX DMA1_Channel5
#define DISCOVERY_EEPROM_DMA_CLK_ENABLE() __HAL_RCC_DMA1_CLK_ENABLE()
#define DISCOVERY_EEPROM_DMA_CLK_DISABLE() __HAL_RCC_DMA1_CLK_DISABLE()
#define DISCOVERY_EEPROM_DMA_TX_IRQn DMA1_Channel4_5_6_7_IRQn
#define DISCOVERY_EEPROM_DMA_RX_IRQn DMA1_Channel4_5_6_7_IRQn
#define DISCOVERY_EEPROM_DMA_PREPRIO 0
#define DISCOVERY_EEPROM_DMA_SUBPRIO 0
## Enumerations

<table>
<thead>
<tr>
<th>Enum</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Led_TypeDef</strong></td>
<td><code>enum</code></td>
</tr>
<tr>
<td></td>
<td><code>Led_TypeDef {</code></td>
</tr>
<tr>
<td></td>
<td><code>   LED3 = 0, LED4 = 1, LED5 = 2, LED6 = 3,</code></td>
</tr>
<tr>
<td></td>
<td><code>   LED_RED = LED3, LED_ORANGE = LED4, LED_GREEN = LED5,</code></td>
</tr>
<tr>
<td></td>
<td><code>   LED_BLUE = LED6</code></td>
</tr>
<tr>
<td></td>
<td><code>}</code></td>
</tr>
<tr>
<td></td>
<td>LED Types Definition. <strong>More...</strong></td>
</tr>
<tr>
<td><strong>Button_TypeDef</strong></td>
<td><code>enum</code></td>
</tr>
<tr>
<td></td>
<td><code>Button_TypeDef {</code></td>
</tr>
<tr>
<td></td>
<td><code>   BUTTON_USER = 0</code></td>
</tr>
<tr>
<td></td>
<td><code>}</code></td>
</tr>
<tr>
<td></td>
<td>BUTTON Types Definition. <strong>More...</strong></td>
</tr>
<tr>
<td><strong>ButtonMode_TypeDef</strong></td>
<td><code>enum</code></td>
</tr>
<tr>
<td></td>
<td><code>ButtonMode_TypeDef {</code></td>
</tr>
<tr>
<td></td>
<td><code>   BUTTON_MODE_GPIO = 0,</code></td>
</tr>
<tr>
<td></td>
<td><code>   BUTTON_MODE_EXTI = 1</code></td>
</tr>
<tr>
<td></td>
<td><code>}</code></td>
</tr>
</tbody>
</table>

**More...**

**LED_TYPES**

- **LED3**
- **LED4**
- **LED5**
- **LED6**
- **LED_RED**
- **LED_ORANGE**
- **LED_GREEN**
- **LED_BLUE**

**BUTTON_TYPES**

- **BUTTON_USER**

**BUTTON_MODE_TYPES**

- **BUTTON_MODE_GPIO**
- **BUTTON_MODE_EXTI**
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BSP_GetVersion</code> (void)</td>
<td>This method returns the STM32F072B DISCO BSP Driver revision.</td>
</tr>
<tr>
<td><code>BSP_LED_Init</code> (Led_TypeDef Led)</td>
<td>Configures LED GPIO.</td>
</tr>
<tr>
<td><code>BSP_LED_On</code> (Led_TypeDef Led)</td>
<td>Turns selected LED On.</td>
</tr>
<tr>
<td><code>BSP_LED_Off</code> (Led_TypeDef Led)</td>
<td>Turns selected LED Off.</td>
</tr>
<tr>
<td><code>BSP_LED_Toggle</code> (Led_TypeDef Led)</td>
<td>Toggles the selected LED.</td>
</tr>
<tr>
<td><code>BSP_PB_Init</code> (Button_TypeDef Button, ButtonMode_TypeDef Mode)</td>
<td>Configures Button GPIO and EXTI Line.</td>
</tr>
<tr>
<td><code>BSP_PB_GetState</code> (Button_TypeDef Button)</td>
<td>Returns the selected Push Button state.</td>
</tr>
</tbody>
</table>
Detailed Description

This file contains definitions for STM32F072B-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 stm32f072b_discovery.h.
stm32f072b_discovery_eeprom.c File Reference

This file provides a set of functions needed to manage an I2C M24LR64 EEPROM memory. More...

#include "stm32f072b_discovery_eeprom.h"

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

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint32_t <strong>BSP_EEPROM_Init</strong> (void)</td>
<td>Initializes peripherals used by the I2C EEPROM driver.</td>
</tr>
<tr>
<td>uint32_t <strong>BSP_EEPROM_ReadBuffer</strong> (uint8_t *pBuffer, uint16_t ReadAddr, uint16_t *NumByteToRead)</td>
<td>Reads a block of data from the EEPROM.</td>
</tr>
<tr>
<td>uint32_t <strong>BSP_EEPROM_WritePage</strong> (uint8_t *pBuffer, uint16_t WriteAddr, uint8_t *NumByteToWrite)</td>
<td>Writes more than one byte to the EEPROM with a single WRITE cycle.</td>
</tr>
<tr>
<td>uint32_t <strong>BSP_EEPROM_WriteBuffer</strong> (uint8_t *pBuffer, uint16_t WriteAddr, uint16_t NumByteToWrite)</td>
<td>Writes buffer of data to the I2C EEPROM.</td>
</tr>
<tr>
<td>uint32_t <strong>BSP_EEPROM_WaitEepromStandbyState</strong> (void)</td>
<td>Wait for EEPROM Standby state.</td>
</tr>
<tr>
<td>void <strong>HAL_I2C_MemTxCpltCallback</strong> (I2C_HandleTypeDef *hi2c)</td>
<td>Memory Tx Transfer completed callbacks.</td>
</tr>
<tr>
<td>void <strong>HAL_I2C_MemRxCpltCallback</strong> (I2C_HandleTypeDef *hi2c)</td>
<td>Memory Rx Transfer completed callbacks.</td>
</tr>
<tr>
<td>__weak uint32_t <strong>BSP_EEPROM_TIMEOUT_UserCallback</strong> (void)</td>
<td>Basic management of the timeout situation.</td>
</tr>
</tbody>
</table>
### Variables

<table>
<thead>
<tr>
<th>Access Modifier</th>
<th>Type</th>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>__IO uint16_t</td>
<td></td>
<td>EEPROMAddress</td>
<td>0</td>
</tr>
<tr>
<td>__IO uint32_t</td>
<td></td>
<td>EEPROMTimeout</td>
<td>EEPROM_LONG_TIMEOUT</td>
</tr>
<tr>
<td>__IO uint16_t</td>
<td></td>
<td>EEPROMDataRead</td>
<td></td>
</tr>
<tr>
<td>__IO uint8_t*</td>
<td></td>
<td>EEPROMDataWritePointer</td>
<td></td>
</tr>
<tr>
<td>__IO uint8_t</td>
<td></td>
<td>EEPROMDataNum</td>
<td></td>
</tr>
</tbody>
</table>
**Detailed Description**

This file provides a set of functions needed to manage an I2C M24LR64 EEPROM memory.

**Author:**

MCD Application Team

===================================================================

**Notes:**

- This driver is intended for stm32F0xx families devices only.
- The I2C EEPROM memory (M24LR64) is available on RF EEPROM daughter board (ANT7-M24LR-A) provided with the STM32F072B-Discovery board. To use this driver you have to connect the ANT7-M24LR-A to CN3 connector.

===================================================================

It implements a high level communication layer for read and write from/to this memory. The needed STM32F0xx hardware resources (I2C and GPIO) are defined in `stm32f072b_discovery.h` file, and the initialization is performed in `EEPROM_IO_Init()` function declared in `stm32f072b_discovery.c` file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and `EEPROM_IO_Init()` function.

**Note:**

In this driver, basic read and write functions (`EEPROM_ReadBuffer()` and `EEPROM_WritePage()`) use Polling mode to perform the data transfer to/from EEPROM memory.

+-----------------------------------------------------------------+
| Pin assignment for M24LR64 EEPROM |
+-----------------------------------+--------------------+
| +---------------------------------+--------------------+
| | STM32F0xx I2C Pins | EEPROM | Pin |
| +--------------------------+--------------------+
| | E0(GND) | 1 (0V) | AC0 | 2 |
| | AC1 | 3 | VSS | 4 (0V) | EEPROM_I2Cx_SDA_PIN/ SDA | SDA | 5 |
| | | | | EEPROM_I2Cx_SCL_PIN/ SCL | SCL | 6 |
| | | | | E1(GND) | 7 (0V) | | |
| | | | | VDD | 8 (3.3V) | +---------------------------------+----------------------+
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 stm32f072b_discovery_eeprom.c.
stm32f072b_discovery_eeprom.h File Reference

This file contains all the functions prototypes for the stm32072b_discovery_eeprom.c firmware driver. More...

#include "stm32f072b_discovery.h"

Go to the source code of this file.
# Define defines for EEPROM operations

```c
#define EEPROM_PAGESIZE 4
#define EEPROM_MAX_SIZE 0x2000 /* 64Kbit*/
#define EEPROM_LONG_TIMEOUT ((uint32_t)(1000))
#define EEPROM_MAX_TRIALS_NUMBER 300
#define EEPROM_OK 0
#define EEPROM_FAIL 1
#define EEPROM_TIMEOUT 2
```
# Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint32_t</td>
<td><strong>BSP_EEPROM_Init</strong> (void)</td>
<td>Initializes peripherals used by the I2C EEPROM driver.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><strong>BSP_EEPROM_ReadBuffer</strong> (uint8_t *pBuffer, uint16_t ReadAddr, uint16_t *NumByteToRead)</td>
<td>Reads a block of data from the EEPROM.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><strong>BSP_EEPROM_WritePage</strong> (uint8_t *pBuffer, uint16_t WriteAddr, uint8_t *NumByteToWrite)</td>
<td>Writes more than one byte to the EEPROM with a single WRITE cycle.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><strong>BSP_EEPROM_WriteBuffer</strong> (uint8_t *pBuffer, uint16_t WriteAddr, uint16_t NumByteToWrite)</td>
<td>Writes buffer of data to the I2C EEPROM.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><strong>BSP_EEPROM_WaitEepromStandbyState</strong> (void)</td>
<td>Wait for EEPROM Standby state.</td>
</tr>
<tr>
<td>__weak uint32_t</td>
<td><strong>BSP_EEPROM_TIMEOUT_UserCallback</strong> (void)</td>
<td>Basic management of the timeout situation.</td>
</tr>
<tr>
<td>void</td>
<td><strong>EEPROM_IO_Init</strong> (void)</td>
<td>Initializes peripherals used by the I2C EEPROM driver.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><strong>EEPROM_IO_WriteData</strong> (uint16_t DevAddress, uint16_t MemAddress, uint32_t pBuffer, uint32_t BufferSize)</td>
<td>Write data to I2C EEPROM driver in using DMA channel.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><strong>EEPROM_IO_ReadData</strong> (uint16_t DevAddress, uint16_t MemAddress, uint32_t pBuffer, uint32_t BufferSize)</td>
<td>Read data from I2C EEPROM driver in using DMA channel.</td>
</tr>
<tr>
<td>HAL_StatusTypeDef</td>
<td>EEPROM_IO_IsDeviceReady (uint16_t DevAddress, uint32_t Trials)</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Checks if target device is ready for communication.</td>
<td></td>
</tr>
</tbody>
</table>
Detailed Description

This file contains all the functions prototypes for the stm32072b_discovery_eeprom.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 stm32f072b_discovery_eeprom.h.
STM32F072B-Discovery BSP User Manual

stm32f072b_discovery_gyroscope.c File Reference

This file provides a set of functions needed to manage the l3gd20 MEMS accelerometer available on STM32F072B-Discovery Kit. More...

#include "stm32f072b_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 GYRO 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 GYRO.</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 L3gd20 MEMS accelerometer available on STM32F072B-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 stm32f072b_discovery_gyroscope.c.
stm32f072b_discovery_gyroscope.h File Reference

This file contains definitions for stm32f072b_discovery_gyroscope.c firmware driver. More...

#include "stm32f072b_discovery.h" #include ".../Components/l3gd20/l3gd20.h"

Go to the source code of this file.
Enumerations

def GYRO_StatusTypeDef { GYRO_OK = 0, GYRO_ERROR = 1, GYRO_TIMEOUT = 2 }
### Functions

<table>
<thead>
<tr>
<th>Data 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 GYRO Initialization.</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>BSP_GYRO_Reset (void)</code></td>
<td>Reboot memory content of GYRO.</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_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>
Detailed Description

This file contains definitions for `stm32f072b_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 stm32f072b_discovery_gyroscope.h.
STM32F072B-Discovery BSP User Manual

Here is a list of all modules:

- **BSP**
  - **STM32F072B_DISCOVERY**
    - **STM32F072B_DISCOVERY Common**
      - Private Constants
      - Private Variables
      - Bus Operations functions
      - Link Operations functions
      - Exported Types
      - Exported Constants
        - **STM32F072B_DISCOVERY LED**
        - **STM32F072B_DISCOVERY BUTTON**
        - **STM32F072B_DISCOVERY BUS**
        - **STM32F072B_DISCOVERY COMPONENT**
    - **Exported Functions**
  - **STM32F072B_DISCOVERY EEPROM**
    - Private Variables
    - Exported Constants
    - Exported Functions
  - **STM32F072B_DISCOVERY GYROSCOPE**
    - Private Variables
    - Exported Types
    - Exported Functions
# STM32F072B-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>stm32f072b_discovery.c</code> [code]</td>
<td>This file provides set of firmware functions to manage LEDs and push-button available on STM32F072B-Discovery Kit from STMicroelectronics.</td>
</tr>
<tr>
<td><code>stm32f072b_discovery.h</code> [code]</td>
<td>This file contains definitions for STM32F072B-Discovery's LEDs and push-buttons hardware resources.</td>
</tr>
<tr>
<td><code>stm32f072b_discovery_eeprom.c</code> [code]</td>
<td>This file provides a set of functions needed to manage an I2C M24LR64 EEPROM memory.</td>
</tr>
<tr>
<td><code>stm32f072b_discovery_eeprom.h</code> [code]</td>
<td>This file contains all the prototypes for the <code>stm32f072b_discovery_eeprom.c</code> firmware driver.</td>
</tr>
<tr>
<td><code>stm32f072b_discovery_gyroscope.c</code> [code]</td>
<td>This file provides a set of functions needed to manage the l3gd20 MEMS accelerometer available on STM32F072B-Discovery Kit.</td>
</tr>
<tr>
<td><code>stm32f072b_discovery_gyroscope.h</code> [code]</td>
<td>This file contains definitions for <code>stm32f072b_discovery_gyroscope.c</code> firmware driver.</td>
</tr>
</tbody>
</table>

Generated on Wed Jul 5 2017 09:43:20 for STM32F072B-Discovery
Directories

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

- **Firmware**
  - **Drivers**
    - **BSP**
      - **STM32F072B-Discovery**
<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>
<tr>
<td><strong>STM32F072B_DISCOVERY Common</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Defines

<table>
<thead>
<tr>
<th>Define</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__STM32F072B_DISCO_BSP_VERSION_MAIN</code></td>
<td>0x02</td>
</tr>
<tr>
<td>STM32F072B DISCO BSP Driver version number</td>
<td>V2.1.6.</td>
</tr>
<tr>
<td><code>__STM32F072B_DISCO_BSP_VERSION_SUB1</code></td>
<td>0x01</td>
</tr>
<tr>
<td><code>__STM32F072B_DISCO_BSP_VERSION_SUB2</code></td>
<td>0x06</td>
</tr>
<tr>
<td><code>__STM32F072B_DISCO_BSP_VERSION_RC</code></td>
<td>0x00</td>
</tr>
<tr>
<td><code>__STM32F072B_DISCO_BSP_VERSION</code></td>
<td></td>
</tr>
</tbody>
</table>
Define Documentation

```c
#define __STM32F072B_DISCO_BSP_VERSION

Value:

```
(((__STM32F072B_DISCO_BSP_VERSION_MAIN << 24) | (   
__STM32F072B_DISCO_BSP_VERSION_SUB1  << 16) | (   
__STM32F072B_DISCO_BSP_VERSION_SUB2  << 8 ) | (   
__STM32F072B_DISCO_BSP_VERSION_RC))
```

Definition at line 66 of file stm32f072b_discovery.c.

Referenced by BSP_GetVersion().

```c
#define __STM32F072B_DISCO_BSP_VERSION_MAIN (0x02)

STM32F072B DISCO BSP Driver version number V2.1.6.

[31:24] main version

Definition at line 62 of file stm32f072b_discovery.c.

```c
#define __STM32F072B_DISCO_BSP_VERSION_RC (0x00)

[7:0] release candidate

Definition at line 65 of file stm32f072b_discovery.c.

```c
#define __STM32F072B_DISCO_BSP_VERSION_SUB1 (0x01)
```
[23:16] sub1 version

Definition at line 63 of file stm32f072b_discovery.c.

#define __STM32F072B_DISCO_BSP_VERSION_SUB2 (0x06)

[15:8] sub2 version

Definition at line 64 of file stm32f072b_discovery.c.

Generated on Wed Jul 5 2017 09:43:19 for STM32F072B-Discovery
BSP User Manual by doxygen 1.7.6.1
STM32F072B-Discovery BSP User Manual

Exported Functions

STM32F072B_DISCOVERY EEPROM
### Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uint32_t BSP_EEPROM_Init (void)</code></td>
<td>Initializes peripherals used by the I2C EEPROM driver.</td>
</tr>
<tr>
<td><code>uint32_t BSP_EEPROM_ReadBuffer (uint8_t *pBuffer, uint16_t ReadAddr, uint16_t *NumByteToRead)</code></td>
<td>Reads a block of data from the EEPROM.</td>
</tr>
<tr>
<td><code>uint32_t BSP_EEPROM_WritePage (uint8_t *pBuffer, uint16_t WriteAddr, uint8_t *NumByteToWrite)</code></td>
<td>Writes more than one byte to the EEPROM with a single WRITE cycle.</td>
</tr>
<tr>
<td><code>uint32_t BSP_EEPROM_WriteBuffer (uint8_t *pBuffer, uint16_t WriteAddr, uint16_t NumByteToWrite)</code></td>
<td>Writes buffer of data to the I2C EEPROM.</td>
</tr>
<tr>
<td><code>uint32_t BSP_EEPROM_WaitEepromStandbyState (void)</code></td>
<td>Wait for EEPROM Standby state.</td>
</tr>
<tr>
<td><code>void HAL_I2C_MemTxCpltCallback (I2C_HandleTypeDef *hi2c)</code></td>
<td>Memory Tx Transfer completed callbacks.</td>
</tr>
<tr>
<td><code>void HAL_I2C_MemRxCpltCallback (I2C_HandleTypeDef *hi2c)</code></td>
<td>Memory Rx Transfer completed callbacks.</td>
</tr>
<tr>
<td><code>__weak uint32_t BSP_EEPROM_TIMEOUT_UserCallback (void)</code></td>
<td>Basic management of the timeout situation.</td>
</tr>
<tr>
<td><code>void EEPROM_IO_Init (void)</code></td>
<td>Initializes peripherals used by the I2C EEPROM driver.</td>
</tr>
<tr>
<td><code>uint32_t EEPROM_IO_WriteData (uint16_t DevAddress, uint16_t MemAddress, uint32_t pBuffer, uint32_t BufferSize)</code></td>
<td>Write data to I2C EEPROM driver in using DMA.</td>
</tr>
</tbody>
</table>
channel.

**uint32_t EEPROM_IO_ReadData (uint16_t DevAddress, uint16_t MemAddress, uint32_t pBuffer, uint32_tBufferSize)**
Read data from I2C EEPROM driver in using DMA channel.

**HAL_StatusTypeDef EEPROM_IO_IsDeviceReady (uint16_t DevAddress, uint32_t Trials)**
Checks if target device is ready for communication.
Function Documentation

uint32_t BSP_EEPROM_Init ( void )

Initializes peripherals used by the I2C EEPROM driver.

Note:
There are 2 different versions of M24LR64 (A01 & A02). Then try to connect on 1st one (EEPROM_I2C_ADDRESS_A01) and if problem, check the 2nd one (EEPROM_I2C_ADDRESS_A02)

Return values:
EEPROM_OK (0) if operation is correctly performed, else return value different from EEPROM_OK (0)

Definition at line 114 of file stm32f072b_discovery_eeprom.c.

References DISCOVERY_EEPROM_I2C_ADDRESS_A01, EEPROM_FAIL, EEPROM_IO_Init(), EEPROM_IO_IsDeviceReady(), EEPROM_MAX_TRIALS_NUMBER, EEPROM_OK, and EEPROMAddress.

uint32_t BSP_EEPROM_ReadBuffer ( uint8_t * pBuffer, uint16_t ReadAddr, uint16_t * NumByteToRead )

Reads a block of data from the EEPROM.

Parameters:
pBuffer pointer to the buffer that receives the data read from the EEPROM.
ReadAddr EEPROM's internal address to start reading
from.

**NumByteToRead** pointer to the variable holding number of bytes to be read from the EEPROM.

**Note:**
The variable pointed by NumByteToRead is reset to 0 when all the data are read from the EEPROM. Application should monitor this variable in order know when the transfer is complete.

**Return values:**

**EEPROM_OK** (0) if operation is correctly performed, else return value different from EEPROM_OK (0) or the timeout user callback.

Definition at line 143 of file *stm32f072b_discovery_eeprom.c*.

References **BSP_EEPROM_TIMEOUT_UserCallback**, **EEPROM_FAIL**, **EEPROM_IO_ReadData**, **EEPROM_LONG_TIMEOUT**, **EEPROM_OK**, **EEPROM_TIMEOUT**, **EEPROMAddress**, **EEPROMDataRead**, and **EEPROMTimeout**.

```c
__weak uint32_t BSP_EEPROM_TIMEOUT_UserCallback ( void )
```

Basic management of the timeout situation.

**Return values:**

**None**

Definition at line 511 of file *stm32f072b_discovery_eeprom.c*.

Referenced by **BSP_EEPROM_ReadBuffer**, **BSP_EEPROM_WaitEepromStandbyState**, and **BSP_EEPROM_WriteBuffer**.

```c
uint32_t BSP_EEPROM_WaitEepromStandbyState ( void )
```
Wait for EEPROM Standby state.

Note: This function allows to wait and check that EEPROM has finished the last operation. It is mostly used after Write operation: after receiving the buffer to be written, the EEPROM may need additional time to actually perform the write operation. During this time, it doesn’t answer to I2C packets addressed to it. Once the write operation is complete the EEPROM responds to its address.

Return values:

EEPROM_OK (0) if operation is correctly performed, else return value different from EEPROM_OK (0) or the timeout user callback.

Definition at line 460 of file stm32f072b_discovery_eeprom.c.

References BSP_EEPROM_TIMEOUT_UserCallback(), EEPROM_FAIL, EEPROM_IO_IsDeviceReady(), EEPROM_LONG_TIMEOUT, EEPROM_MAX_TRIALS_NUMBER, EEPROM_OK, EEPROM_TIMEOUT, EEPROMAddress, and EEPROMTimeout.

Referenced by BSP_EEPROM_WriteBuffer().

```c
uint32_t BSP_EEPROM_WriteBuffer ( uint8_t * pBuffer,
        uint16_t WriteAddr,
        uint16_t NumByteToWrite
    )

Writes buffer of data to the I2C EEPROM.

Parameters:

pBuffer pointer to the buffer containing the data to be written to the EEPROM.
**WriteAddr**  
EEPROM's internal address to write to.

**NumByteToWrite**  
number of bytes to write to the EEPROM.

**Return values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEPROM_OK (0)</td>
<td>if operation is correctly performed, else return value different from EEPROM_OK (0) or the timeout user callback.</td>
</tr>
</tbody>
</table>

- If WriteAddr is EEPROM_PAGESIZE aligned
- If NumByteToWrite < EEPROM_PAGESIZE
- If NumByteToWrite < EEPROM_PAGESIZE
- If the number of data to be written is more than the remaining space in the current page:
  - Write the data contained in same page
  - Write the remaining data in the following page

Definition at line 225 of file `stm32f072b_discovery_eeprom.c`.

References `BSP_EEPROM_TIMEOUT_UserCallback()`, `BSP_EEPROM_WaitEepromStandbyState()`, `BSP_EEPROM_WritePage()`, `EEPROM_FAIL`, `EEPROM_LONG_TIMEOUT`, `EEPROM_OK`, `EEPROM_PAGESIZE`, `EEPROM_TIMEOUT`, `EEPROMDataNum`, and `EEPROMTimeout`.

```c
uint32_t BSP_EEPROM_WritePage ( uint8_t * pBuffer,  
                               uint16_t WriteAddr,  
                               uint8_t * NumByteToWrite  
)
```

Writes more than one byte to the EEPROM with a single WRITE cycle.
Note:
The number of bytes (combined to write start address) must not cross the EEPROM page boundary. This function can only write into the boundaries of an EEPROM page. This function doesn’t check on boundaries condition (in this driver the function BSP_EEPROM_WriteBuffer() which calls BSP_EEPROM_WritePage() is responsible of checking on Page boundaries).

Parameters:
- **pBuffer** pointer to the buffer containing the data to be written to the EEPROM.
- **WriteAddr** EEPROM's internal address to write to.
- **NumByteToWrite** pointer to the variable holding number of bytes to be written into the EEPROM.

Note:
The variable pointed by NumByteToWrite is reset to 0 when all the data are written to the EEPROM. Application should monitor this variable in order know when the transfer is complete. This function just configure the communication and enable the DMA channel to transfer data. Meanwhile, the user application may perform other tasks in parallel.

Return values:
- **EEPROM_OK** (0) if operation is correctly performed, else return value different from EEPROM_OK (0) or the timeout user callback.

Definition at line 200 of file stm32f072b_discovery_eeprom.c.

References EEPROM_IO_WriteData(), EEPROM_OK, EEPROMAddress, and EEPROMDataWritePointer.

Referenced by BSP_EEPROM_WriteBuffer().
void EEPROM_IO_Init ( void )

Initializes peripherals used by the I2C EEPROM driver.

**Return values:**

*None*

Definition at line 729 of file `stm32f072b_discovery.c`.

References `I2Cx_Init()`.

Referenced by `BSP_EEPROM_Init()`.

---

HAL_StatusTypeDef EEPROM_IO_IsDeviceReady ( uint16_t DevAddress,
                                          uint32_t Trials )

Checks if target device is ready for communication.

**Note:**

This function is used with Memory devices

**Parameters:**

- **DevAddress** Target device address
- **Trials** Number of trials

**Return values:**

- **HAL** status

Definition at line 781 of file `stm32f072b_discovery.c`.

References `I2cHandle`, and `I2Cx_TIMEOUT_MAX`.

Referenced by `BSP_EEPROM_Init()`, and
`BSP_EEPROM_WaitEepromStandbyState()`.
uint32_t EEPROM_IO_ReadData ( uint16_t DevAddress, 
    uint16_t MemAddress, 
    uint32_t pBuffer, 
    uint32_t BufferSize 
)

Read data from I2C EEPROM driver in using DMA channel.

Parameters:
- **DevAddress**  Target device address
- **MemAddress** Internal memory address
- **pBuffer** Pointer to data buffer
- **BufferSize** Amount of data to be read

Return values:
- **HAL** status

Definition at line 762 of file stm32f072b_discovery.c.

References **I2Cx_ReadBuffer()**, and **I2Cx_ReadData()**.

Referenced by **BSP_EEPROM_ReadBuffer()**.

uint32_t EEPROM_IO_WriteData ( uint16_t DevAddress, 
    uint16_t MemAddress, 
    uint32_t pBuffer, 
    uint32_t BufferSize 
)

Write data to I2C EEPROM driver in using DMA channel.

Parameters:
- **DevAddress** Target device address
- **MemAddress** Internal memory address
**pBuffer**          Pointer to data buffer  
**BufferSize**      Amount of data to be sent  

**Return values:**  
**HAL** status  

Definition at line 742 of file `stm32f072b_discovery.c`.  
References **I2Cx_WriteBuffer()**, and **I2Cx_WriteData()**.  
Referenced by **BSP_EEPROM_WritePage()**.

```c
void HAL_I2C_MemRxCpltCallback ( I2C_HandleTypeDef * hi2c )
```

Memory Rx Transfer completed callbacks.  

**Parameters:**  
  - **hi2c** I2C handle  

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

Definition at line 502 of file `stm32f072b_discovery_eeprom.c`.  
References **EEPROMDataRead**.

```c
void HAL_I2C_MemTxCpltCallback ( I2C_HandleTypeDef * hi2c )
```

Memory Tx Transfer completed callbacks.  

**Parameters:**  
  - **hi2c** I2C handle  

**Return values:**  
  - **None**
Definition at line 492 of file stm32f072b_discovery_eeprom.c.

References EEPROMDataWritePointer.
# STM32F072B-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

`STM32F072B_DISCOVERY Common`
## Functions

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint32_t</td>
<td><strong>BSP_GetVersion</strong> (void)</td>
<td>This method returns the STM32F072B DISCO BSP Driver revision.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_LED_Init</strong> (Led_TypeDef Led)</td>
<td>Configures LED GPIO.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_LED_On</strong> (Led_TypeDef Led)</td>
<td>Turns selected LED On.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_LED_Off</strong> (Led_TypeDef Led)</td>
<td>Turns selected LED Off.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_LED_Toggle</strong> (Led_TypeDef Led)</td>
<td>Toggles the selected LED.</td>
</tr>
<tr>
<td>void</td>
<td><strong>BSP_PB_Init</strong> (Button_TypeDef Button, ButtonMode_TypeDef Mode)</td>
<td>Configures Button GPIO and EXTI Line.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><strong>BSP_PB_GetState</strong> (Button_TypeDef Button)</td>
<td>Returns the selected Push Button state.</td>
</tr>
</tbody>
</table>
**Function Documentation**

```c
uint32_t BSP_GetVersion ( void )
```

This method returns the STM32F072B DISCO BSP Driver revision.

**Return values:**

- **version**: 0xXYZR (8 bits for each decimal, R for RC)

Definition at line 163 of file `stm32f072b_discovery.c`.

References `__STM32F072B_DISCO_BSP_VERSION`.

```c
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:
  - LED3
  - LED4
  - LED5
  - LED6

**Return values:**

- **None**

Definition at line 178 of file `stm32f072b_discovery.c`.

References `LED_PIN`, `LED_PORT`, and `LEDx_GPIO_CLK_ENABLE`.

```c
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:
  - LED3
  - LED4
  - LED5
  - LED6

**Return values:**

- **None**

Definition at line 221 of file `stm32f072b_discovery.c`.

References **LED_PIN**, and **LED_PORT**.

---

```c
void BSP_LED_Off (Led_TypeDef Led)
```

Turns selected LED On.

**Parameters:**

- **Led** Specifies the Led to be set on. This parameter can be one of following parameters:
  - LED3
  - LED4
  - LED5
  - LED6

**Return values:**

- **None**

Definition at line 206 of file `stm32f072b_discovery.c`.

References **LED_PIN**, and **LED_PORT**.
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:
  - LED3
  - LED4
  - LED5
  - LED6

**Return values:**

- None

Definition at line 236 of file stm32f072b_discovery.c.

References LED_PIN, and LED_PORT.

```c
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 290 of file stm32f072b_discovery.c.

References BUTTON_PIN, and BUTTON_PORT.

```c
void BSP_PB_Init (Button_TypeDef Button,
```
Configures Button GPIO and EXTI Line.

**Parameters:**

- **Button** Specifies the Button to be configured. This parameter should be: BUTTON_USER
- **Mode** 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 252 of file stm32f072b_discovery.c.

References BUTTON_IRQn, BUTTON_MODE_EXTI, BUTTON_MODE_GPIO, BUTTON_PIN, BUTTON_PORT, and BUTTONx_GPIO_CLK_ENABLE.
## Exported Functions

<table>
<thead>
<tr>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F072B_DISCOVERY GYROSCOPE</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_GYRO_Init (void)</code></td>
<td>Set GYRO 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 GYRO.</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>
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 192 of file *stm32f072b\_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 176 of file *stm32f072b\_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 205 of file stm32f072b_discovery_gyroscope.c.

References GyroscopeDrv.

uint8_t BSP_GYRO_Init ( void )

Set GYRO Initialization.

**Return values:**
- **GYRO_OK** if no problem during initialization

Definition at line 68 of file stm32f072b_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:**
Definition at line 152 of file stm32f072b_discovery_gyroscope.c.

References GyroscopeDrv.

```
uint8_t BSP_GYRO_ReadID ( void )
```

Read ID of Gyroscope component.

**Return values:**

ID

Definition at line 123 of file stm32f072b_discovery_gyroscope.c.

References GyroscopeDrv.

```
void BSP_GYRO_Reset ( void )
```

Reboot memory content of GYRO.

**Return values:**

None

Definition at line 138 of file stm32f072b_discovery_gyroscope.c.

References GyroscopeDrv.
**STM32F072B-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>
</table>

**Private Variables**

*STM32F072B_DISCOVERY Common*
Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIO_TypeDef * LED_PORT</td>
<td>ledn</td>
</tr>
<tr>
<td>const uint16_t LED_PIN</td>
<td></td>
</tr>
<tr>
<td>GPIO_TypeDef * BUTTON_PORT</td>
<td>USER_BUTTON_GPIO_PORT</td>
</tr>
<tr>
<td>const uint16_t BUTTON_PIN</td>
<td>USER_BUTTON_PIN</td>
</tr>
<tr>
<td>const uint8_t BUTTON_IRQn</td>
<td>USER_BUTTON_EXTIIRQn</td>
</tr>
<tr>
<td>I2C_HandleTypeDef I2cHandle</td>
<td></td>
</tr>
<tr>
<td>uint32_t SpixTimeout</td>
<td>SPIx_TIMEOUT_MAX</td>
</tr>
<tr>
<td>static SPI_HandleTypeDef SpiHandle</td>
<td></td>
</tr>
</tbody>
</table>

I2C variables.
### Variable Documentation

**const uint8_t** `BUTTON_IRQn[BUTTONn] = {USER_BUTTON_EXTI_IRQn}`

Definition at line 90 of file `stm32f072b_discovery.c`. Referenced by `BSP_PB_Init()`.

**const uint16_t** `BUTTON_PIN[BUTTONn] = {USER_BUTTON_PIN}`

Definition at line 89 of file `stm32f072b_discovery.c`. Referenced by `BSP_PB_GetState()`, and `BSP_PB_Init()`.

**GPIO_TypeDef** `BUTTON_PORT[BUTTONn] = {USER_BUTTON_GPIO_PORT}`

Definition at line 88 of file `stm32f072b_discovery.c`. Referenced by `BSP_PB_GetState()`, and `BSP_PB_Init()`.

**I2C_HandleTypeDef** `I2cHandle`

BUS variables.

Definition at line 96 of file `stm32f072b_discovery.c`. Referenced by `EEPROM_IO_IsDeviceReady()`, `I2Cx_Error()`, `I2Cx_Init()`, `I2Cx_ReadBuffer()`, `I2Cx_ReadData()`, `I2Cx_WriteBuffer()`, and `I2Cx_WriteData()`.

**const uint16_t** `LED_PIN[LEDn]`

Initial value:
{LED3_PIN,
  LED4_PIN,
  LED5_PIN,
  LED6_PIN}

Definition at line 83 of file stm32f072b_discovery.c.

Referenced by BSP_LED_Init(), BSP_LED_Off(), BSP_LED_On(), and BSP_LED_Toggle().

GPIO_TypeDef* LED_PORT[LEDn]

Initial value:

{LED3_GPIO_PORT,
  LED4_GPIO_PORT,
  LED5_GPIO_PORT,
  LED6_GPIO_PORT}

Definition at line 78 of file stm32f072b_discovery.c.

Referenced by BSP_LED_Init(), BSP_LED_Off(), BSP_LED_On(), and BSP_LED_Toggle().

SPI_HandleTypeDef SpiHandle [static]

Definition at line 101 of file stm32f072b_discovery.c.

Referenced by SPIx_Error(), SPIx_Init(), and SPIx_WriteRead().

uint32_t SpixTimeout = SPIx_TIMEOUT_MAX

Definition at line 100 of file stm32f072b_discovery.c.

Referenced by SPIx_WriteRead().
### Exported Types

<table>
<thead>
<tr>
<th>STM32F072B_DISCOVERY Common</th>
<th></th>
</tr>
</thead>
</table>
## Enumerations

<table>
<thead>
<tr>
<th>Enum Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Led_TypeDef</td>
<td><code>enum</code></td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>LED3 = 0, LED4 = 1, LED5 = 2, LED6 = 3,</td>
</tr>
<tr>
<td></td>
<td>LED_RED = LED3, LED_ORANGE = LED4, LED_GREEN = LED5, LED_BLUE = LED6</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>LED Types Definition. More...</td>
</tr>
<tr>
<td>Button_TypeDef</td>
<td><code>enum</code></td>
</tr>
<tr>
<td></td>
<td>{ BUTTON_USER = 0 }</td>
</tr>
<tr>
<td></td>
<td>BUTTON Types Definition. More...</td>
</tr>
<tr>
<td>ButtonMode_TypeDef</td>
<td><code>enum</code></td>
</tr>
<tr>
<td></td>
<td>{ BUTTON_MODE_GPIO = 0, BUTTON_MODE_EXTI = 1 }</td>
</tr>
</tbody>
</table>
Enumeration Type Documentation

enum **Button_TypeDef**

BUTTON Types Definition.

**Enumerator:**

`BUTTON_USER`

Definition at line 83 of file `stm32f072b_discovery.h`.

enum **ButtonMode_TypeDef**

**Enumerator:**

`BUTTON_MODE_GPIO`

`BUTTON_MODE_EXTI`

Definition at line 88 of file `stm32f072b_discovery.h`.

enum **Led_TypeDef**

LED Types Definition.

**Enumerator:**

`LED3`

`LED4`

`LED5`

`LED6`

`LED_RED`

`LED_ORANGE`

`LED_GREEN`

`LED_BLUE`
Definition at line 67 of file stm32f072b_discovery.h.
STM32F072B-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>STM32F072B_DISCOVERY BUTTON</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exported Constants</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Defines

```c
#define BUTTONn 1
#define USER_BUTTON_PIN GPIO_PIN_0 /* PA0 */
USER 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_EXTI_IRQn EXTI0_1_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 Documentation

```c
#define BUTTONn  1
```

Definition at line 148 of file stm32f072b_discovery.h.

```c
#define BUTTONx_GPIO_CLK_DISABLE ( __BUTTON__ ) ( ((__BUTTON__) ==
```

Definition at line 161 of file stm32f072b_discovery.h.

```c
#define BUTTONx_GPIO_CLK_ENABLE ( __BUTTON__ ) do { if((
```

Definition at line 159 of file stm32f072b_discovery.h.

Referenced by BSP_PB_Init().

```c
#define USER_BUTTON_EXTI_IRQHandler   EXTI0_1_IRQHandler 
```

Definition at line 157 of file stm32f072b_discovery.h.

```c
#define USER_BUTTON_GPIO_CLK_DISABLE ( )   __HAL_RCC_GPIOA_CLK_DISABLE() 
```

Definition at line 156 of file stm32f072b_discovery.h.

```c
#define USER_BUTTON_GPIO_CLK_ENABLE ( )   __HAL_RCC_GPIOA_CLK_ENABLE() 
```

Definition at line 155 of file stm32f072b_discovery.h.

```c
#define USER_BUTTON_GPIO_PORT   GPIOA 
```

Definition at line 154 of file stm32f072b_discovery.h.
Definition at line 154 of file stm32f072b_discovery.h.

```c
#define USER_BUTTON_PIN GPIO_PIN_0 /* PA0 */
```

USER push-button.

Definition at line 153 of file stm32f072b_discovery.h.

---

Generated on Wed Jul 5 2017 09:43:19 for STM32F072B-Discovery
BSP User Manual by doxygen 1.7.6.1
## STM32F072B-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Defines</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STM32F072B_DISCOVERY COMPONENT</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Exported Constants</strong></td>
<td></td>
</tr>
</tbody>
</table>
### Defines

```c
#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  GPIOC /* GPIOC */

#define GYRO_CS_GPIO_CLK_ENABLE()  __HAL_RCC_GPIOC_CLK_ENABLE()
#define GYRO_CS_GPIO_CLK_DISABLE()  __HAL_RCC_GPIOC_CLK_DISABLE()
#define GYRO_CS_PIN  GPIO_PIN_0 /* PC.00 */
#define GYRO_INT_GPIO_PORT  GPIOC /* GPIOC */
#define GYRO_INT_GPIO_CLK_ENABLE()  __HAL_RCC_GPIOC_CLK_ENABLE()
#define GYRO_INT_GPIO_CLK_DISABLE()  __HAL_RCC_GPIOC_CLK_DISABLE()
#define GYRO_INT1_PIN  GPIO_PIN_1 /* PC.01 */
#define GYRO_INT1_EXTI_IRQn  EXTI0_1_IRQn
#define GYRO_INT2_PIN  GPIO_PIN_2 /* PC.02 */
#define GYRO_INT2_EXTIIRQn  EXTI2_3_IRQHandler
#define DISCOVERY_EEPROM_DMA  DMA1
I2C EEPROM Interface pins.

#define DISCOVERY_EEPROM_DMA_CHANNEL_TX  DMA1_Channel4
#define DISCOVERY_EEPROM_DMA_CHANNEL_RX  DMA1_Channel5
#define DISCOVERY_EEPROM_DMA_CLK_ENABLE()  __HAL_RCC_DMA1_CLK_ENABLE()
#define DISCOVERY_EEPROM_DMA_CLK_DISABLE()  __HAL_RCC_DMA1_CLK_DISABLE()
#define DISCOVERY_EEPROM_DMA_TX_IRQn  DMA1_Channel4_IRQHandler
#define DISCOVERY_EEPROM_DMA_RX_IRQn  DMA1_Channel5_IRQHandler
#define DISCOVERY_EEPROM_DMA_PREPRIO  0
#define DISCOVERY_EEPROM_DMA_SUBPRIO  0
```
### Define Documentation

<table>
<thead>
<tr>
<th>Definition</th>
<th>Description</th>
<th>File</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>#define DISCOVERY_EEPROM_DMA DMA1</td>
<td>I2C EEPROM Interface pins.</td>
<td>stm32f072b_discovery.h</td>
<td>260</td>
</tr>
<tr>
<td>#define DISCOVERY_EEPROM_DMA_CHANNEL_RX DMA1_Channel5</td>
<td>Definition at line 262 of file stm32f072b_discovery.h.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#define DISCOVERY_EEPROM_DMA_CHANNEL_TX DMA1_Channel4</td>
<td>Definition at line 261 of file stm32f072b_discovery.h.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#define DISCOVERY_EEPROM_DMA_CLK_DISABLE() __HAL_RCC_DMA1_CLK_DISABLE()</td>
<td>Definition at line 264 of file stm32f072b_discovery.h.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#define DISCOVERY_EEPROM_DMA_CLK_ENABLE() __HAL_RCC_DMA1_CLK_ENABLE()</td>
<td>Definition at line 263 of file stm32f072b_discovery.h.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#define DISCOVERY_EEPROM_DMA_PREPRIO 0</td>
<td>Referenced by I2Cx_MspInit().</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
#define DISCOVERY_EEPROM_DMA_RX_IRQn DMA1_Channel4

Definition at line 268 of file stm32f072b_discovery.h.
Referenced by I2Cx_MspInit().

#define DISCOVERY_EEPROM_DMA_SUBPRIO 0

Definition at line 269 of file stm32f072b_discovery.h.
Referenced by I2Cx_MspInit().

#define DISCOVERY_EEPROM_DMA_TX_IRQn DMA1_Channel4

Definition at line 266 of file stm32f072b_discovery.h.
Referenced by I2Cx_MspInit().

#define DUMMY_BYTE ((uint8_t)0x00)

Definition at line 234 of file stm32f072b_discovery.h.
Referenced by GYRO_IO_Read().

#define GYRO_CS_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOC

Definition at line 245 of file stm32f072b_discovery.h.
#define GYRO_CS_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOC_CLK_ENABLE()
Definition at line 244 of file stm32f072b_discovery.h.
Referenced by GYRO_IO_Init().

#define GYRO_CS_GPIO_PORT GPIOC /* GPIOC */
GYRO SPI Interface pins.
Definition at line 243 of file stm32f072b_discovery.h.
Referenced by GYRO_IO_Init().

#define GYRO_CS_HIGH ( ) HAL_GPIO_WritePin(GYRO_CS_GPIO_PORT, GPIO_PIN_0, GPIO_PIN_VALUE_HIGH)
Definition at line 238 of file stm32f072b_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, GPIO_PIN_0, GPIO_PIN_VALUE_LOW)
Definition at line 237 of file stm32f072b_discovery.h.
Referenced by GYRO_IO_Read(), and GYRO_IO_Write().

#define GYRO_CS_PIN GPIO_PIN_0 /* PC.00 */
Definition at line 246 of file stm32f072b_discovery.h.
Referenced by GYRO_IO_Init().
```c
#define GYRO_INT1_EXTI_IRQn EXTI0_1_IRQn
Definition at line 252 of file stm32f072b_discovery.h.

#define GYRO_INT1_PIN GPIO_PIN_1 /* PC.01 */
Definition at line 251 of file stm32f072b_discovery.h.
Referenced by GYRO_IO_Init().

#define GYRO_INT2_EXTI_IRQn EXTI2_3_IRQn
Definition at line 254 of file stm32f072b_discovery.h.

#define GYRO_INT2_PIN GPIO_PIN_2 /* PC.02 */
Definition at line 253 of file stm32f072b_discovery.h.
Referenced by GYRO_IO_Init().

#define GYRO_INT_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOC_CLK_DISABLE()
Definition at line 250 of file stm32f072b_discovery.h.

#define GYRO_INT_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOC_CLK_ENABLE()
Definition at line 249 of file stm32f072b_discovery.h.
Referenced by GYRO_IO_Init().

#define GYRO_INT_GPIO_PORT GPIOC /* GPIOC */
```
#define MULTIPLEBYTE_CMD ((uint8_t)0x40)

Definition at line 248 of file stm32f072b_discovery.h.

Referenced by GYRO_IO_Init().


#define READWRITE_CMD ((uint8_t)0x80)

Definition at line 232 of file stm32f072b_discovery.h.

Referenced by GYRO_IO_Read(), and GYRO_IO_Write().

Generated on Wed Jul 5 2017 09:43:20 for STM32F072B-Discovery BSP User Manual by doxygen 1.7.6.1
STM32F072B-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>STM32F072B_DISCOVERY BUS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exported Constants</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Defines

```c
#define DISCOVERY_SPIx SPI2
#define DISCOVERY_SPIx_CLOCK_ENABLE() __HAL_RCC_SPI2_CLK_ENABLE()
#define DISCOVERY_SPIx_GPIO_PORT GPIOB /* GPIOB */
#define DISCOVERY_SPIx_AF GPIO_AF0_SPI2
#define DISCOVERY_SPIx_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_ENABLE()
#define DISCOVERY_SPIx_GPIO_CLK_DISABLE() __HAL_RCC_GPIOB_CLK_DISABLE()
#define DISCOVERY_SPIx_SCK_PIN GPIO_PIN_13 /* PB.13 */
#define DISCOVERY_SPIx_MISO_PIN GPIO_PIN_14 /* PB.14 */
#define DISCOVERY_SPIx_MOSI_PIN GPIO_PIN_15 /* PB.15 */
#define SPIx_TIMEOUT_MAX ((uint32_t)0x1000)
#define DISCOVERY_I2Cx I2C2
  Definition for I2C Interface pins (I2C2 used)
#define DISCOVERY_I2Cx_CLK_ENABLE() __HAL_RCC_I2C2_CLK_ENABLE()
#define DISCOVERY_I2Cx_CLK_DISABLE() __HAL_RCC_I2C2_CLK_DISABLE()
#define DISCOVERY_I2Cx_FORCE_RESET() __HAL_RCC_I2C2_FORCE_RESET()
#define DISCOVERY_I2Cx_RELEASE_RESET() __HAL_RCC_I2C2_RELEASE_RESET()
#define DISCOVERY_I2Cx_SCL_PIN GPIO_PIN_10 /* PB.10 */
#define DISCOVERY_I2Cx_SDA_PIN GPIO_PIN_11 /* PB.11 */
#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_AF1_I2C2
#define I2Cx_TIMEOUT_MAX ((uint32_t)0x10000)
#define DISCOVERY_I2Cx_TIMING 0x40B32537
#define DISCOVERY_EEPROM_I2C_ADDRESS_A01 0xA0
```
### Define Documentation

<table>
<thead>
<tr>
<th>Define</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#define DISCOVERY_EEPROM_I2C_ADDRESS_A01</code></td>
<td>0xA0</td>
</tr>
<tr>
<td>Definition at line 217 of file <code>stm32f072b_discovery.h</code>.</td>
<td></td>
</tr>
<tr>
<td>Referenced by <code>BSP_EEPROM_Init()</code>.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Define</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#define DISCOVERY_I2Cx</code></td>
<td>I2C2</td>
</tr>
<tr>
<td>Definition for I2C Interface pins (I2C2 used)</td>
<td></td>
</tr>
<tr>
<td>Definition at line 193 of file <code>stm32f072b_discovery.h</code>.</td>
<td></td>
</tr>
<tr>
<td>Referenced by <code>I2Cx_Init()</code>.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Define</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#define DISCOVERY_I2Cx_AF</code></td>
<td>GPIO_AF1_I2C2</td>
</tr>
<tr>
<td>Definition at line 205 of file <code>stm32f072b_discovery.h</code>.</td>
<td></td>
</tr>
<tr>
<td>Referenced by <code>I2Cx_MspInit()</code>.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Define</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#define DISCOVERY_I2Cx_CLK_DISABLE</code> ( )</td>
<td>__HAL_RCC_I2C2_CLK_DISABLE()</td>
</tr>
<tr>
<td>Definition at line 195 of file <code>stm32f072b_discovery.h</code>.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Define</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#define DISCOVERY_I2Cx_CLK_ENABLE</code> ( )</td>
<td>__HAL_RCC_I2C2_CLK_ENABLE()</td>
</tr>
<tr>
<td>Definition at line 194 of file <code>stm32f072b_discovery.h</code>.</td>
<td></td>
</tr>
<tr>
<td>Referenced by <code>I2Cx_MspInit()</code>.</td>
<td></td>
</tr>
</tbody>
</table>
#define DISCOVERY_I2Cx_FORCE_RESET ( ) __HAL_RCC_I2C2

Definition at line 196 of file stm32f072b_discovery.h.

#define DISCOVERY_I2Cx_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOB_CLK_DISABLE()

Definition at line 204 of file stm32f072b_discovery.h.

#define DISCOVERY_I2Cx_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOB_CLK_ENABLE()

Definition at line 203 of file stm32f072b_discovery.h.

Referenced by I2Cx_MspInit().

#define DISCOVERY_I2Cx_GPIO_PORT GPIOB /* GPIOB */

Definition at line 202 of file stm32f072b_discovery.h.

Referenced by I2Cx_MspInit().

#define DISCOVERY_I2Cx_RELEASE_RESET ( ) __HAL_RCC_I2C

Definition at line 197 of file stm32f072b_discovery.h.

#define DISCOVERY_I2Cx_SCL_PIN GPIO_PIN_10 /* PB.10 */

Definition at line 199 of file stm32f072b_discovery.h.

Referenced by I2Cx_MspInit().

#define DISCOVERY_I2Cx_SDA_PIN GPIO_PIN_11 /* PB.11 */
Definition at line 200 of file stm32f072b_discovery.h.
Referenced by I2Cx_MspInit().

\#define DISCOVERY_I2Cx_TIMING 0x40B32537

Definition at line 215 of file stm32f072b_discovery.h.
Referenced by I2Cx_Init().

\#define DISCOVERY_SPIx SPI2

Definition at line 171 of file stm32f072b_discovery.h.
Referenced by SPIx_Init().

\#define DISCOVERY_SPIx_AF GPIO_AF0_SPI2

Definition at line 174 of file stm32f072b_discovery.h.
Referenced by SPIx_MspInit().

\#define DISCOVERY_SPIx_CLOCK_ENABLE () __HAL_RCC_SPI2_CLK_ENABLE()

Definition at line 172 of file stm32f072b_discovery.h.
Referenced by SPIx_MspInit().

\#define DISCOVERY_SPIx_GPIO_CLK_DISABLE () __HAL_RCC_GPIOB_CLK_DISABLE()

Definition at line 176 of file stm32f072b_discovery.h.
```c
#define DISCOVERY_SPIx_GPIO_CLK_ENABLE () __HAL_RCC_GPIOB_CLK_ENABLE()

Definition at line 175 of file stm32f072b_discovery.h.
Referenced by SPIx_MspInit().

#define DISCOVERY_SPIx_GPIO_PORT GPIOB /* GPIOB */

Definition at line 173 of file stm32f072b_discovery.h.
Referenced by SPIx_MspInit().

#define DISCOVERY_SPIx_MISO_PIN GPIO_PIN_14 /* PB.14 */

Definition at line 178 of file stm32f072b_discovery.h.
Referenced by SPIx_MspInit().

#define DISCOVERY_SPIx_MOSI_PIN GPIO_PIN_15 /* PB.15 */

Definition at line 179 of file stm32f072b_discovery.h.
Referenced by SPIx_MspInit().

#define DISCOVERY_SPIx_SCK_PIN GPIO_PIN_13 /* PB.13 */

Definition at line 177 of file stm32f072b_discovery.h.
Referenced by SPIx_MspInit().

#define I2Cx_TIMEOUT_MAX ((uint32_t)0x10000)
```
Definition at line 213 of file `stm32f072b_discovery.h`.

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

```c
#define SPIx_TIMEOUT_MAX ((uint32_t)0x1000)
```

Definition at line 185 of file `stm32f072b_discovery.h`.

Generated on Wed Jul 5 2017 09:43:20 for STM32F072B-Discovery
BSP User Manual by `doxygen` 1.7.6.1
# STM32F072B-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>

## Exported Constants

| STM32F072B_DISCOVERY EEPROM |
# Define EEPROM parameters

```c
#define EEPROM_PAGESIZE 4
#define EEPROM_MAX_SIZE 0x2000 /* 64Kbit*/
#define EEPROM_LONG_TIMEOUT ((uint32_t)(1000))
#define EEPROM_MAX_TRIALS_NUMBER 300
#define EEPROM_OK 0
#define EEPROM_FAIL 1
#define EEPROM_TIMEOUT 2
```
Define Documentation

```c
#define EEPROM_FAIL   1
```

Definition at line 76 of file `stm32f072b_discovery_eeprom.h`.

Referenced by `BSP_EEPROM_Init()`, `BSP_EEPROM_ReadBuffer()`, `BSP_EEPROM_WaitEepromStandbyState()`, and `BSP_EEPROM_WriteBuffer()`.

```c
#define EEPROM_LONG_TIMEOUT  ((uint32_t)(1000))
```

Definition at line 70 of file `stm32f072b_discovery_eeprom.h`.

Referenced by `BSP_EEPROM_ReadBuffer()`, `BSP_EEPROM_WaitEepromStandbyState()`, and `BSP_EEPROM_WriteBuffer()`.

```c
#define EEPROM_MAX_SIZE  0x2000 /* 64Kbit*/
```

Definition at line 66 of file `stm32f072b_discovery_eeprom.h`.

```c
#define EEPROM_MAX_TRIALS_NUMBER  300
```

Definition at line 73 of file `stm32f072b_discovery_eeprom.h`.

Referenced by `BSP_EEPROM_Init()`, and `BSP_EEPROM_WaitEepromStandbyState()`.

```c
#define EEPROM_OK   0
```

Definition at line 75 of file `stm32f072b_discovery_eeprom.h`. 
Referenced by `BSP_EEPROM_Init()`, `BSP_EEPROM_ReadBuffer()`, `BSP_EEPROM_WaitEepromStandbyState()`, `BSP_EEPROM_WriteBuffer()`, and `BSP_EEPROM_WritePage()`.

```c
#define EEPROM_PAGESIZE 4
```

Definition at line 65 of file `stm32f072b_discovery_eeprom.h`.

Referenced by `BSP_EEPROM_WriteBuffer()`.

```c
#define EEPROM_TIMEOUT 2
```

Definition at line 77 of file `stm32f072b_discovery_eeprom.h`.

Referenced by `BSP_EEPROM_ReadBuffer()`, `BSP_EEPROM_WaitEepromStandbyState()`, and `BSP_EEPROM_WriteBuffer()`.
STM32F072B-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>
<tbody>
<tr>
<td><strong>Link Operations</strong> functions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

STM32F072B_DISCOVERY Common
### Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>GYRO_IO_Init</code> (void)</td>
<td></td>
<td>Configures the GYRO 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></td>
<td>Writes one byte to the GYRO.</td>
</tr>
<tr>
<td>void</td>
<td><code>GYRO_IO_Read</code> (uint8_t *pBuffer, uint8_t ReadAddr, uint16_t NumByteToRead)</td>
<td></td>
<td>Reads a block of data from the GYROSCOPE.</td>
</tr>
<tr>
<td>void</td>
<td><code>EEPROM_IO_Init</code> (void)</td>
<td></td>
<td>Initializes peripherals used by the I2C EEPROM driver.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><code>EEPROM_IO_WriteData</code> (uint16_t DevAddress, uint16_t MemAddress, uint32_t pBuffer, uint32_t BufferSize)</td>
<td></td>
<td>Write data to I2C EEPROM driver in using DMA channel.</td>
</tr>
<tr>
<td>uint32_t</td>
<td><code>EEPROM_IO_ReadData</code> (uint16_t DevAddress, uint16_t MemAddress, uint32_t pBuffer, uint32_t BufferSize)</td>
<td></td>
<td>Read data from I2C EEPROM driver in using DMA channel.</td>
</tr>
<tr>
<td>HAL_StatusTypeDef</td>
<td><code>EEPROM_IO_IsDeviceReady</code> (uint16_t DevAddress, uint32_t Trials)</td>
<td></td>
<td>Checks if target device is ready for communication.</td>
</tr>
</tbody>
</table>
Function Documentation

`void EEPROM_IO_Init ( void )`

Initializes peripherals used by the I2C EEPROM driver.

**Return values:**

None

Definition at line 729 of file `stm32f072b_discovery.c`.

References `I2Cx_Init()`.

Referenced by `BSP_EEPROM_Init()`.

`HAL_StatusTypeDef EEPROM_IO_IsDeviceReady ( uint16_t DevAddress, uint32_t Trials )`

Checks if target device is ready for communication.

**Note:**

This function is used with Memory devices

**Parameters:**

- **DevAddress**  Target device address
- **Trials**  Number of trials

**Return values:**

`HAL`  status

Definition at line 781 of file `stm32f072b_discovery.c`.

References `I2cHandle`, and `I2Cx_TIMEOUT_MAX`. 
Referenced by **BSP_EEPROM_Init()**, and **BSP_EEPROM_WaitEepromStandbyState()**.

```c
uint32_t EEPROM_IO_ReadData ( uint16_t DevAddress,
                            uint16_t MemAddress,
                            uint32_t pBuffer,
                            uint32_t BufferSize )
```

Read data from I2C EEPROM driver in using DMA channel.

**Parameters:**
- **DevAddress** Target device address
- **MemAddress** Internal memory address
- **pBuffer** Pointer to data buffer
- **BufferSize** Amount of data to be read

**Return values:**
- **HAL** status

Definition at line **762** of file **stm32f072b_discovery.c**.

References **I2Cx_ReadBuffer()**, and **I2Cx_ReadData()**.

Referenced by **BSP_EEPROM_ReadBuffer()**.

```c
uint32_t EEPROM_IO_WriteData ( uint16_t DevAddress,
                               uint16_t MemAddress,
                               uint32_t pBuffer,
                               uint32_t BufferSize )
```

Write data to I2C EEPROM driver in using DMA channel.
Parameters:

- **DevAddress**: Target device address
- **MemAddress**: Internal memory address
- **pBuffer**: Pointer to data buffer
- **BufferSize**: Amount of data to be sent

Return values:

- **HAL** status

Definition at line 742 of file `stm32f072b_discovery.c`.

References `I2Cx_WriteBuffer()`, and `I2Cx_WriteData()`.

Referenced by `BSP_EEPROM_WritePage()`.

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

Configures the GYRO SPI interface.

Return values:

- **None**

Definition at line 624 of file `stm32f072b_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 693 of file `stm32f072b_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 GYRO.

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

**Return values:**
- None

Definition at line 658 of file `stm32f072b_discovery.c`. 
References GYRO_CS_HIGH, GYRO_CS_LOW, MULTIPLEBYTE_CMD, and SPIx_WriteRead().
# STM32F072B-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>
</table>

## Private Variables

`STM32F072B_DISCOVERY EEPROM`
## Variables

<table>
<thead>
<tr>
<th>AccessSpecifier</th>
<th>Type</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>__IO</td>
<td>uint16_t</td>
<td>EEPROMAddress</td>
</tr>
<tr>
<td>__IO</td>
<td>uint32_t</td>
<td>EEPROMTimeout</td>
</tr>
<tr>
<td>__IO</td>
<td>uint16_t</td>
<td>EEPROMDataRead</td>
</tr>
<tr>
<td>__IO</td>
<td>uint8_t*</td>
<td>EEPROMDataWritePointer</td>
</tr>
<tr>
<td>__IO</td>
<td>uint8_t</td>
<td>EEPROMDataNum</td>
</tr>
</tbody>
</table>
Variable Documentation

__IO uint16_t EEPROMAddress = 0

Definition at line 92 of file stm32f072b_discovery_eeprom.c.

Referenced by BSP_EEPROM_Init(), BSP_EEPROM_ReadBuffer(), BSP_EEPROM_WaitEepromStandbyState(), and BSP_EEPROM_WritePage().

__IO uint8_t EEPROMDataNum

Definition at line 96 of file stm32f072b_discovery_eeprom.c.

Referenced by BSP_EEPROM_WriteBuffer().

__IO uint16_t EEPROMDataRead

Definition at line 94 of file stm32f072b_discovery_eeprom.c.

Referenced by BSP_EEPROM_ReadBuffer(), and HAL_I2C_MemRxCpltCallback().

__IO uint8_t* EEPROMDataWritePointer

Definition at line 95 of file stm32f072b_discovery_eeprom.c.

Referenced by BSP_EEPROM_WritePage(), and HAL_I2C_MemTxCpltCallback().

__IO uint32_t EEPROMTimeout = EEPROM_LONG_TIMEOUT
Definition at line 93 of file stm32f072b_discovery_eeprom.c.

Referenced by BSP_EEPROM_ReadBuffer(), BSP_EEPROM_WaitEepromStandbyState(), and BSP_EEPROM_WriteBuffer().
# STM32F072B-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>
<tr>
<td><strong>Enumerations</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Exported Types

| STM32F072B_DISCOVERY_GYROSCOPE |
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 `stm32f072b_discovery_gyroscope.h`.
STM32F072B-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>

STM32F072B_DISCOVERY GYROSCOPE
Variables

| static GYRO_DrvTypeDef * GyroscopeDrv |
Variable Documentation

GYRO_DrvTypeDef* GyroscopeDrv [static]

Definition at line 54 of file stm32f072b_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 Jul 5 2017 09:43:20 for STM32F072B-Discovery BSP User Manual by doxygen 1.7.6.1
# STM32F072B-Discovery BSP User Manual

## Bus Operations functions

STM32F072B_DISCOVERY Common
## Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</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 uint32_t</td>
<td><strong>I2Cx_WriteData</strong> (uint8_t Addr, uint16_t Reg, uint8_t Value)</td>
<td>Write a value in a register of the device through BUS.</td>
</tr>
<tr>
<td>static uint32_t</td>
<td><strong>I2Cx_WriteBuffer</strong> (uint8_t Addr, uint16_t Reg, uint8_t *pBuffer, uint16_t Length)</td>
<td>Write a value in a register of the device through BUS.</td>
</tr>
<tr>
<td>static uint32_t</td>
<td><strong>I2Cx_ReadData</strong> (uint8_t Addr, uint16_t Reg, uint8_t *Value)</td>
<td>Read a register of the device through BUS.</td>
</tr>
<tr>
<td>static uint32_t</td>
<td><strong>I2Cx_ReadBuffer</strong> (uint8_t Addr, uint16_t Reg, uint8_t *pBuffer, uint16_t Length)</td>
<td>Reads multiple data on the BUS.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>I2Cx_Error</strong> (uint8_t Addr)</td>
<td>Discovery I2Cx error treatment function.</td>
</tr>
<tr>
<td>static void</td>
<td><strong>SPIx_Init</strong> (void)</td>
<td>SPI1 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>SPI1 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>
<tr>
<td>static void</td>
<td><strong>I2Cx_MspInit</strong> (I2C_HandleTypeDef *hi2c)</td>
<td>Discovery I2Cx MSP Initialization.</td>
</tr>
</tbody>
</table>
Function Documentation

static void I2Cx_Error ( uint8_t Addr ) [static]

Discovery I2Cx error treatment function.

Return values:
None

Definition at line 503 of file stm32f072b_discovery.c.

References I2cHandle, and I2Cx_Init().

Referenced by I2Cx_ReadBuffer(), I2Cx_ReadData(), I2Cx_WriteBuffer(), and I2Cx_WriteData().

static void I2Cx_Init ( void ) [static]

Discovery I2Cx Bus initialization.

Return values:
None

Definition at line 386 of file stm32f072b_discovery.c.

References DISCOVERY_I2Cx, DISCOVERY_I2Cx_TIMING, I2cHandle, and I2Cx_MspInit().

Referenced by EEPROM_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 314 of file stm32f072b_discovery.c.

References DISCOVERY_EEPROM_DMA_CHANNEL_RX, DISCOVERY_EEPROM_DMA_CHANNEL_TX, DISCOVERY_EEPROM_DMA_CLK_ENABLE, DISCOVERY_EEPROM_DMA_PREPRIO, DISCOVERY_EEPROM_DMA_RX_IRQn, DISCOVERY_EEPROM_DMA_SUBPRIO, DISCOVERY_EEPROM_DMA_TX_IRQn, 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 uint32_t I2Cx_ReadBuffer ( uint8_t Addr, uint16_t Reg, uint8_t * pBuffer, uint16_t Length )

Reads multiple data on the BUS.

Parameters:

- Addr: I2C Address
- Reg: Reg Address
- pBuffer: pointer to read data buffer
- Length: length of the data
Read a register of the device through BUS.

**Parameters:**
- **Addr**: Device address on BUS
- **Reg**: The target register address to read

**Return values:**
- **Value**: read register value

Definition at line 459 of file `stm32f072b_discovery.c`.

References **I2cHandle, I2Cx_Error(), and I2Cx_TIMEOUT_MAX**.

Referenced by **EEPROM_IO_ReadData()**.

```c
static uint32_t I2Cx_ReadData ( uint8_t Addr,
                               uint16_t Reg,
                               uint8_t * Value ) [static]
```

```c
static uint32_t I2Cx_WriteBuffer ( uint8_t Addr,
                                  uint16_t Reg,
                                  uint8_t * pBuffer,
                                  uint16_t Length ) [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
- **pBuffer** The target register value to be written
- **Length** buffer size to be written

**Return values:**
- None

Definition at line 437 of file *stm32f072b_discovery.c*.

References **I2cHandle**, and **I2Cx_Error()**.

Referenced by **EEPROM_IO_WriteData()**.

```c
static uint32_t I2Cx_WriteData ( uint8_t Addr,
                               uint16_t Reg,
                               uint8_t Value )
```

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 413 of file *stm32f072b_discovery.c*.

References **I2cHandle**, **I2Cx_Error()**, and **I2Cx_TIMEOUT_MAX**.
Referenced by `EEPROM_IO_WriteData()`.

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

SPI1 error treatment function.

Return values:

None

Definition at line 573 of file `stm32f072b_discovery.c`.

References SpiHandle, and `SPIx_Init()`.

Referenced by `SPIx_WriteRead()`.
```

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

SPI1 Bus initialization.

Return values:

None

Definition at line 519 of file `stm32f072b_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 ) [static]

SPI MSP Init.

Parameters:

hspi SPI handle
```
**Return values:**

None

Definition at line 588 of file stm32f072b_discovery.c.

References DISCOVERY_SPIx_AF, DISCOVERY_SPIx_CLOCK_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 553 of file stm32f072b_discovery.c.

References SpiHandle, SPIx_Error(), and SpixTimeout.

Referenced by GYRO_IO_Read(), and GYRO_IO_Write().
STM32F072B-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>

**STM32F072B_DISCOVERY LED**

Exposed Constants

Define for STM32F072B_DISCOVERY board. [More...](#)
Defines

#define LEDn 4
#define LED3_PIN GPIO_PIN_6
#define LED3_GPIO_PORT GPIOC
#define LED3_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define LED3_GPIO_CLK_DISABLE() __HAL_RCC_GPIOC_CLK_DISABLE()
#define LED4_PIN GPIO_PIN_8
#define LED4_GPIO_PORT GPIOC
#define LED4_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define LED4_GPIO_CLK_DISABLE() __HAL_RCC_GPIOC_CLK_DISABLE()
#define LED5_PIN GPIO_PIN_9
#define LED5_GPIO_PORT GPIOC
#define LED5_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define LED5_GPIO_CLK_DISABLE() __HAL_RCC_GPIOC_CLK_DISABLE()
#define LED6_PIN GPIO_PIN_7
#define LED6_GPIO_PORT GPIOC
#define LED6_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define LED6_GPIO_CLK_DISABLE() __HAL_RCC_GPIOC_CLK_DISABLE()
#define LEDx_GPIO_CLK_ENABLE(__LED__) __HAL_RCC_GPIOC_CLK_ENABLE()
#define LEDx_GPIO_CLK_DISABLE(__LED__) __HAL_RCC_GPIOC_CLK_DISABLE()
Detailed Description

Define for STM32F072B_DISCOVERY board.
Define Documentation

```c
#define LED3_GPIO_CLK_DISABLE () __HAL_RCC_GPIOC_CLK_DISABLE()
Definition at line 115 of file stm32f072b_discovery.h.
```

```c
#define LED3_GPIO_CLK_ENABLE () __HAL_RCC_GPIOC_CLK_ENABLE()
Definition at line 114 of file stm32f072b_discovery.h.
```

```c
#define LED3_GPIO_PORT GPIOC
Definition at line 113 of file stm32f072b_discovery.h.
```

```c
#define LED3_PIN GPIO_PIN_6
Definition at line 112 of file stm32f072b_discovery.h.
```

```c
#define LED4_GPIO_CLK_DISABLE () __HAL_RCC_GPIOC_CLK_DISABLE()
Definition at line 120 of file stm32f072b_discovery.h.
```

```c
#define LED4_GPIO_CLK_ENABLE () __HAL_RCC_GPIOC_CLK_ENABLE()
Definition at line 119 of file stm32f072b_discovery.h.
```

```c
#define LED4_GPIO_PORT GPIOC
Definition at line 118 of file stm32f072b_discovery.h.
```
```c
#define LED4_PIN GPIO_PIN_8

Definition at line 117 of file stm32f072b_discovery.h.

#define LED5_GPIO_CLK_DISABLE () __HAL_RCC_GPIOC_CLK_DISABLE() 

Definition at line 125 of file stm32f072b_discovery.h.

#define LED5_GPIO_CLK_ENABLE () __HAL_RCC_GPIOC_CLK_ENABLE() 

Definition at line 124 of file stm32f072b_discovery.h.

#define LED5_GPIO_PORT GPIOC 

Definition at line 123 of file stm32f072b_discovery.h.

#define LED5_PIN GPIO_PIN_9 

Definition at line 122 of file stm32f072b_discovery.h.

#define LED6_GPIO_CLK_DISABLE () __HAL_RCC_GPIOC_CLK_DISABLE() 

Definition at line 130 of file stm32f072b_discovery.h.

#define LED6_GPIO_CLK_ENABLE () __HAL_RCC_GPIOC_CLK_ENABLE() 

Definition at line 129 of file stm32f072b_discovery.h.
```
#define LED6_GPIO_PORT GPIOC

Definition at line 128 of file stm32f072b_discovery.h.

#define LED6_PIN GPIO_PIN_7

Definition at line 127 of file stm32f072b_discovery.h.

#define LEDn 4

Definition at line 110 of file stm32f072b_discovery.h.

#define LEDx_GPIO_CLK_DISABLE ( __LED__ )

Value:

```
```

Definition at line 137 of file stm32f072b_discovery.h.

#define LEDx_GPIO_CLK_ENABLE ( __LED__ )

Value:

```
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__) == LED5) LED5_GPIO_CLK_ENABLE(); else 
if( (LED__) == LED6) LED6_GPIO_CLK_ENABLE();
while(0)

Definition at line 132 of file stm32f072b_discovery.h.

Referenced by BSP_LED_Init().

Generated on Wed Jul 5 2017 09:43:19 for STM32F072B-Discovery
BSP User Manual by doxygen 1.7.6.1
# STM32F072B-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></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Firmware Directory Reference**
| directory | Drivers |

Generated on Wed Jul 5 2017 09:43:20 for STM32F072B-Discovery BSP User Manual by `doxygen` 1.7.6.1
## STM32F072B-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
# STM32F072B-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 | STM32F072B-Discovery
---|---

Generated on Wed Jul 5 2017 09:43:20 for STM32F072B-Discovery
BSP User Manual by doxygen 1.7.6.1
STM32F072B-Discovery BSP User Manual

STM32F072B-Discovery Directory Reference
### Files

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>stm32f072b_discovery.c</code></td>
<td>This file provides set of firmware functions to manage Leds, push-button available on STM32F072B-Discovery Kit from STMicroelectronics.</td>
</tr>
<tr>
<td><code>stm32f072b_discovery.h</code></td>
<td>This file contains definitions for STM32F072B-Discovery's Leds, push-buttons hardware resources.</td>
</tr>
<tr>
<td><code>stm32f072b_discovery_eeprom.c</code></td>
<td>This file provides a set of functions needed to manage an I2C M24LR64 EEPROM memory.</td>
</tr>
<tr>
<td><code>stm32f072b_discovery_eeprom.h</code></td>
<td>This file contains all the functions prototypes for the stm32072b_discovery_eeprom.c firmware driver.</td>
</tr>
<tr>
<td><code>stm32f072b_discovery_gyroscope.c</code></td>
<td>This file provides a set of functions needed to manage the l3gd20 MEMS accelerometer available on STM32F072B-Discovery Kit.</td>
</tr>
<tr>
<td><code>stm32f072b_discovery_gyroscope.h</code></td>
<td>This file contains definitions for</td>
</tr>
</tbody>
</table>
stm32f072b_discovery.h

Go to the documentation of this file.

00001    /**
00002    ******************************************
00003    * @file    stm32f072b_discovery.h
00004    * @author  MCD Application Team
00005    * @brief   This file contains definitions
00006    *          for STM32F072B-Discovery's Leds, push-
00007    *          buttons hardware resources.
00008    ******************************************
00009    *
00010    * <h2><center>&copy; COPYRIGHT(c) 2016 STM
00011    * microelectronics</center></h2>
00012    *
00013    * Redistribution and use in source and binary
00014    * forms, with or without modification,
00015    * are permitted provided that the following
00016    * conditions are met:
00017    * 1. Redistributions of source code must
00018    *    retain the above copyright notice,
00019    * 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 __STM32F072B_DISCOVERY_H
#define __STM32F072B_DISCOVERY_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes -----------------------------------------------*/
#include "stm32f0xx_hal.h"

/**
@addtogroup BSP
@{*/

/**
@defgroup STM32F072B_DISCOVERY STM32F072B_DISCOVERY
@{*/

/**
@defgroup STM32F072B_DISCOVERY_Common STM32F072B_DISCOVERY_Common
@{*/

/**
@brief LED Types Definition
*/
typedef enum
{
  */

/**
@defgroup STM32F072B_DISCOVERY_Exported_Types Exported Types
@{*/

/**
  @brief LED Types Definition
*/
typedef enum
{
LED3 = 0,
LED4 = 1,
LED5 = 2,
LED6 = 3,
/* Color led aliases */
LED_RED = LED3,
LED_ORANGE = LED4,
LED_GREEN = LED5,
LED_BLUE = LED6
Led_TypeDef;
/**
 * @brief BUTTON Types Definition
 */
typedef enum
{
  BUTTON_USER = 0
} Button_TypeDef;
typedef enum
{
  BUTTON_MODE_GPIO = 0,
  BUTTON_MODE_EXTI = 1
} ButtonMode_TypeDef;
/**
 * @defgroup STM32F072B_DISCOVERY_Exported_Constants Exported Constants
 */
/* @defgroup STM32F072B_DISCOVERY_Exported_Constants Exported Constants */
* @}*/
* */
/** @defgroup STM32F072B_DISCOVERY_Exported_Constants Exported Constants */
* @}*/
* */
#if !defined (USE_STM32072B_DISCO)
#define USE_STM32072B_DISCO
#endif
/**	@defgroup STM32F072B_DISCOVERY_LED STM32F072B_DISCOVERY LED
 *
 */
#define LEDn 4
#define LED3_PIN GPIO_PIN_6
#define LED3_GPIO_PORT GPIOC
#define LED3_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define LED3_GPIO_CLK_DISABLE() __HAL_RCC_GPIOC_CLK_DISABLE()
#define LED4_PIN GPIO_PIN_8
#define LED4_GPIO_PORT GPIOC
#define LED4_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define LED4_GPIO_CLK_DISABLE() __HAL_RCC_GPIOC_CLK_DISABLE()
#define LED5_PIN GPIO_PIN_9
#define LED5_GPIO_PORT GPIOC
#define LED5_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define LED5_GPIO_CLK_DISABLE() __HAL_RCC_GPIOC_CLK_DISABLE()
#define LED6_PIN GPI
#define LED6_GPIO_PORT GPIOC

#define LED6_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define LED6_GPIO_CLK_DISABLE() __HAL_RCC_GPIOC_CLK_DISABLE()

#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();
} while(0)

#define LEDx_GPIO_CLK_DISABLE(__LED__) ((((__LED__) == LED3) ? LED3_GPIO_CLK_DISABLE() : ((__LED__) == LED4) ? LED4_GPIO_CLK_DISABLE() : ((__LED__) == LED5) ? LED5_GPIO_CLK_DISABLE() : ((__LED__) == LED6) ? LED6_GPIO_CLK_DISABLE() : 0 ))

/**
 * @}
 */

/**
 * @defgroup STM32F072B_DISCOVERY_BUTTON STM32F072B_DISCOVERY_BUTTON
 */

/** @defgroup STM32F072B_DISCOVERY_BUTTON STM32F072B_DISCOVERY_BUTTON */
#define BUTTONn 1

/**
 * @brief USER push-button
 */
#define USER_BUTTON_PIN GPIO_PIN_0 /* PA0 */
#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_IRQHandler EXTI0_1_IRQHandler
#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 )

/**
 * @}
 */
/**
 * @defgroup STM32F072B_DISCOVERY_BUS STM32F072B_DISCOVERY BUS
 */
#if defined(HAL_SPI_MODULE_ENABLED)
/*##################### SPI1 ###############
#####################*/
#define DISCOVERY_SPIx SPI2
#define DISCOVERY_SPIx_CLOCK_ENABLE() __HAL_RCC_SPI2_CLK_ENABLE()
#endif

#define DISCOVERY_SPIx_CLOCK_ENABLE() __HAL_RCC_SPI2_CLK_ENABLE()
```c
#define DISCOVERY_SPIx_GPIO_PORT GPIOB
     /* GPIOB */
#define DISCOVERY_SPIx_AF GPIO_AF0_SPI2
#define DISCOVERY_SPIx_GPIO_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_ENABLE()
#define DISCOVERY_SPIx_GPIO_CLK_DISABLE() __HAL_RCC_GPIOB_CLK_DISABLE()
#define DISCOVERY_SPIx_SCK_PIN GPIO_PIN_13 /* PB.13 */
#define DISCOVERY_SPIx_MISO_PIN GPIO_PIN_14 /* PB.14 */
#define DISCOVERY_SPIx_MOSI_PIN GPIO_PIN_15 /* PB.15 */

/* 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)
#endif /* HAL_SPI_MODULE_ENABLED */
#if defined(HAL_I2C_MODULE_ENABLED)
/*##################### I2C2 ###############

/**
 * @brief Definition for I2C Interface pins (I2C2 used)
 */
#define DISCOVERY_I2Cx I2C2
```
#define DISCOVERY_I2Cx_CLK_ENABLE() __HAL_RCC_I2C2_CLK_ENABLE()
#define DISCOVERY_I2Cx_CLK_DISABLE() __HAL_RCC_I2C2_CLK_DISABLE()
#define DISCOVERY_I2Cx_FORCE_RESET() __HAL_RCC_I2C2_FORCE_RESET()
#define DISCOVERY_I2Cx_RELEASE_RESET() __HAL_RCC_I2C2_RELEASE_RESET()

#define DISCOVERY_I2Cx_SCL_PIN GPIO_PIN_10 /* PB.10 */
#define DISCOVERY_I2Cx_SDA_PIN GPIO_PIN_11 /* PB.11 */

#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_AF1_I2C2

#ifdef __cplusplus
extern "C" {
#endif

/* 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 (0xFFFF)

#ifdef __cplusplus
}
#endif

#define I2Cx_TIMEOUT_MAX ((uint32_t)0x10000)
#define DISCOVERY_I2Cx_TIMING 0x40B32537

#define DISCOVERY_EEPROM_I2C_ADDRESS_A01 0xA0

#endif /* HAL_I2C_MODULE_ENABLED */

/**
 * @}
 */

/**
 * @defgroup STM32F072B_DISCOVERY_COMPONENT
 STM32F072B_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
*/
#define DUMMY_BYTE ((uint8_t)0x00)

/*	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												GPIOC
    /* GPIOC */
#define GYRO_CS_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define GYRO_CS_GPIO_CLK_DISABLE() __HAL_RCC_GPIOC_CLK_DISABLE()
#define GYRO_CS_PIN																				GPIO_PIN_0
    /* PC.00 */
#define GYRO_INT_GPIO_PORT															GPIOC
    /* GPIOC */
#define GYRO_INT_GPIO_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_ENABLE()
#define GYRO_INT_GPIO_CLK_DISABLE() __HAL_RCC_GPIOC_CLK_DISABLE()
#define GYRO_INT1_PIN																				GPIO_PIN_1
#define GYRO_INT1_EXTI_IRQn														EXTI0_1_IRQn
#define GYRO_INT2_PIN																				GPIO_PIN_2
#define GYRO_INT2_EXTI_IRQn														EXTI2_3_IRQn

/*##################### EEPROM ##############*/
/**
 * @brief I2C EEPROM Interface pins
 */
#define DISCOVERY_EEPROM_DMA_DMA1
#define DISCOVERY_EEPROM_DMA_CHANNEL_TX_DMA1_Channel4
#define DISCOVERY_EEPROM_DMA_CHANNEL_RX
DMA1_Channel5

#define DISCOVERY_EEPROM_DMA_CLK_ENABLE()
__HAL_RCC_DMA1_CLK_ENABLE()

#define DISCOVERY_EEPROM_DMA_CLK_DISABLE()
__HAL_RCC_DMA1_CLK_DISABLE()

#define DISCOVERY_EEPROM_DMA_TX_IRQn
DMA1_Channel4_5_6_7_IRQn

#define DISCOVERY_EEPROM_DMA_RX_IRQn
DMA1_Channel4_5_6_7_IRQn

#define DISCOVERY_EEPROM_DMA_PREPRIO
0

#define DISCOVERY_EEPROM_DMA_SUBPRIO
0

/**
 * @}
 */

/**
 * @}
 */

/**
 * @defgroup STM32F072B_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 Mode);
uint32_t BSP_PB_GetState(Button_TypeDef Button);
/**
/*
 * __cplusplus
 */
#endif

#ifndef __cplusplus
#endif /* __STM32F072B_DISCOVERY_H */

/*************************************************
(C) COPYRIGHT STMicroelectronics
*****END OF FILE*****/

Generated on Wed Jul 5 2017 09:43:19 for STM32F072B-Discovery
BSP User Manual by doxygen 1.7.6.1
stm32f072b_discovery.c

Go to the documentation of this file.

```c
00001 /**
00002 **************************************************************************************************************************************
00003 * @file     stm32f072b_discovery.c
00004 * @author   MCD Application Team
00005 * @brief    This file provides set of firmware functions to manage Leds, push-button available on STM32F072B-Discovery Kit from STMicroelectronics.
00006 **************************************************************************************************************************************
00007 *
00008 * @attention
00009 *
00010 * <h2><center>&copy; COPYRIGHT(c) 2016 STMicroelectronics</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.
/* Includes -----------------------------------*/
#include "stm32f072b_discovery.h"
/**	@addtogroup BSP
 * @{

/**	@addtogroup STM32F072B_DISCOVERY
 * @{

/**	@addtogroup STM32F072B_DISCOVERY_Common
 * @brief This file provides firmware functions to manage Leds, push-buttons, and temperature sensor (LM75) available on STM32F072B-DISCO evaluation board from STMicroelectronics.
 * @{

/**	@defgroup STM32F072B_DISCOVERY_Private_Constants
 * Private Constants
 * @{

/**	@brief STM32F072B DISCO BSP Driver version number V2.1.6
 * */
#define __STM32F072B_DISCO_BSP_VERSION_MAIN (0x02) /*!< [31:24] main version */
#define __STM32F072B_DISCO_BSP_VERSION_SUB1 (0x01) /*!< [23:16] sub1 version */
#define __STM32F072B_DISCO_BSP_VERSION_SUB2
```c
00065 #define __STM32F072B_DISCO_BSP_VERSION_RC
00066 #define __STM32F072B_DISCO_BSP_VERSION
00067 ((__STM32F072B_DISCO_BSP_VERSION_MAIN << 24) |
00068  ((__STM32F072B_DISCO_BSP_VERSION_SUB1 << 16) |
00069  ((__STM32F072B_DISCO_BSP_VERSION_SUB2 << 8) |
00070  ((__STM32F072B_DISCO_BSP_VERSION_RC))
00071 /**
00072  * @}*/
00074 /** @defgroup STM32F072B_DISCOVERY_Private_Variables Private Variables */
00076  */
00078 GPIO_TypeDef* LED_PORT[LEDn] = {LED3_GPIO_PORT, LED4_GPIO_PORT, LED5_GPIO_PORT, LED6_GPIO_PORT};
00080 const uint16_t LED_PIN[LEDn] = {LED3_PIN, LED4_PIN, LED5_PIN, LED6_PIN};
00082 GPIO_TypeDef* BUTTON_PORT[BUTTONn] = {USER_B UTTON_GPIO_PORT};
00084 const uint16_t BUTTON_PIN[BUTTONn] = {USER_B
UTTON_PIN};

const uint8_t BUTTON_IRQn[BUTTONn] = {USER_BUTTON_EXTI_IRQn};

/**
 * @brief BUS variables
 */
#if defined(HAL_I2C_MODULE_ENABLED)
I2C_HandleTypeDef I2cHandle;
#endif

#if defined(HAL_SPI_MODULE_ENABLED)
uint32_t SpixTimeout = SPIx_TIMEOUT_MAX;
/*<! Value of Timeout when SPI communication fails */
#else
SPI_HandleTypeDef SpiHandle;
#endif

/**
 * @defgroup STM32F072B_DISCOVERY_BUS_Operations_Functions Bus Operations functions
 */
#if defined(HAL_I2C_MODULE_ENABLED)
/* I2Cx bus function */
static void I2Cx_Init(void);
static uint32_t I2Cx_WriteData(uint8_t Addr, uint16_t Reg, uint8_t Value);
static uint32_t I2Cx_WriteBuffer(uint8_t Addr, uint16_t Reg, uint8_t *pBuffer, uint16_t Length);
static uint32_t I2Cx_ReadData(uint8_t Addr, uint16_t Reg, uint8_t *Value);
00118  static uint32_t  I2Cx_ReadBuffer(uint8_t Addr,
     uint16_t Reg, uint8_t *pBuffer, uint16_t Length);
00119  static void  I2Cx_Error(uint8_t Addr);
00120  #endif /* HAL_I2C_MODULE_ENABLED */
00121
00122  #if defined(HAL_SPI_MODULE_ENABLED)
00123  /* SPIx bus function */
00124  static void  SPIx_Init(void);
00125  static uint8_t  SPIx_WriteRead(uint8_t byte);
00126  static void  SPIx_Error(void);
00127  static void  SPIx_MspInit(SPI_HandleTypeDef *hspi);
00128
00129  /**
00130   * @}
00131  */
00132
00133  /** @defgroup STM32F072B_DISCOVERY_LINK_Operations_Functions Link Operations functions
00134   * @{
00135   */
00136
00137  /* Link function for GYRO peripheral */
00138  void  GYRO_IO_Init(void);
00139  void  GYRO_IO_Write(uint8_t* pBuffer, uint8_t WriteAddr, uint16_t NumByteToWrite);
00140  void  GYRO_IO_Read(uint8_t* pBuffer, uint8_t ReadAddr, uint16_t NumByteToRead);
00141  #endif
00142
00143  #if defined(HAL_I2C_MODULE_ENABLED)
00144  /* Link function for I2C EEPROM peripheral */
00145  void  EEPROM_IO_Init(void);
uint32_t EEPROM_IO_WriteData(uint16_t DevAddress, uint16_t MemAddress, uint32_t pBuffer, uint32_t BufferSize);
uint32_t EEPROM_IO_ReadData(uint16_t DevAddress, uint16_t MemAddress, uint32_t pBuffer, uint32_t BufferSize);
HAL_StatusTypeDef EEPROM_IO_IsDeviceReady(uint16_t DevAddress, uint32_t Trials);

#ifdef /* HAL_I2C_MODULE_ENABLED */
#endif /* HAL_I2C_MODULE_ENABLED */

/**
 * @}
 */
/**
 * @addtogroup STM32F072B_DISCOVERY_Exported_Functions
 * @{
 */

/**
 * @brief This method returns the STM32F072B DISCO BSP Driver revision
 * @retval version : 0xXYZR (8bits for each decimal, R for RC)
 */
uint32_t BSP_GetVersion(void) {
    return __STM32F072B_DISCO_BSP_VERSION;
}

/**
 * @brief Configures LED GPIO.
 * @param Led Specifies the Led to be configured.
 * This parameter can be one of following parameters:
 * @arg LED3
 */
```c
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_PuPd_DOWN */
    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.
- **@retval** None

This parameter can be one of following parameters:

- **@arg** LED3
- **@arg** LED4
- **@arg** LED5
void BSP_LED_On(Led_TypeDef Led)
{
    HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[Led], GPIO_PIN_SET);
}

void BSP_LED_Off(Led_TypeDef Led)
{
    HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[Led], GPIO_PIN_RESET);
}

void BSP_LED_Toggle(Led_TypeDef Led)
{
    HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[Led], GPIO_PIN_TOGGLE);
}

/*
@brief  Toggles the selected LED.
@param  Led Specifies the Led to be toggled.
      This parameter can be one of following parameters:
      @arg LED3
      @arg LED4
      @arg LED5
      @arg LED6
      @retval None
*/

void BSP_LED_Toggle(Led_TypeDef Led)
{
    HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[Led], GPIO_PIN_TOGGLE);
}

/*
@brief  Toggles the selected LED.
@param  Led Specifies the Led to be toggled.
      This parameter can be one of following parameters:
      @arg LED3
      @arg LED4
      @arg LED5
      @arg LED6
      @retval None
*/

void BSP_LED_Toggle(Led_TypeDef Led)
{
    HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[Led], GPIO_PIN_TOGGLE);
}
void BSP_LED_Toggle(Led_TypeDef Led) {
    HAL_GPIO_TogglePin(LED_PORT[Led], LED_PIN[Led]);
}

/**
 * @brief Configures Button GPIO and EXTI Line.
 * @param Button Specifies the Button to be configured.
 * This parameter should be: BUTTON_USER
 * @param Mode Specifies Button mode.
 * This parameter can be one of following parameters:
 * @arg BUTTON_MODE_GPIO: Button will be used as simple IO
 * @arg BUTTON_MODE_EXTI: Button will be connected to EXTI line with interrupt
 * @retval None
 */
void BSP_PB_Init(Button_TypeDef Button, ButtonMode_TypeDef Mode) {
    GPIO_InitTypeDef GPIO_InitStruct;
    /* Enable the BUTTON Clock */
    BUTTONx_GPIO_CLK_ENABLE(Button);
    __HAL_RCC_SYSCFG_CLK_ENABLE();
    GPIO_InitStruct.Pin = BUTTON_PIN[Button];
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
    }
if (Mode == BUTTON_MODE_GPIO)
{
    /* Configure Button pin as input */
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
    GPIO_InitStruct.Pull = GPIO_PULLDOWN;
    HAL_GPIO_Init(BUTTON_PORT[Button], &GPIO_InitStruct);
}

if (Mode == BUTTON_MODE EXTI)
{
    /* Configure Button pin as input with External interrupt */
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    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]), 0x03, 0x00);
    HAL_NVIC_EnableIRQ((IRQn_Type)(BUTTON_IRQn[Button]));
}

/**
 * @brief Returns the selected Push Button state.
 * @param Button Specifies the Button to be checked.
 * @retval The Button GPIO pin value.
uint32_t BSP_PB_GetState(Button_TypeDef Button) {
    return HAL_GPIO_ReadPin(BUTTON_PORT[Button], BUTTON_PIN[Button]);
}

/**
 * @}
 */

/**
 * @addtogroup STM32F072B_DISCOVERY_BUS_Operations_Functions
 * @{
 */

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

#if defined(HAL_I2C_MODULE_ENABLED)
/*******************************
I2C Routines
***********************************/

/**
@brief Discovery I2Cx MSP Initialization
@param hi2c I2C handle
@retval None
*/

static void I2Cx_MspInit(I2C_HandleTypeDef *hi2c) {
    GPIO_InitTypeDef GPIO_InitStruct;
    static DMA_HandleTypeDef hdma_tx;
    static DMA_HandleTypeDef hdma_rx;
/* Enable GPIO clock */
DISCOVERY_I2Cx_GPIO_CLK_ENABLE();

/* Configure I2C Tx and Rx as alternate function */
GPIO_InitStruct.Pin = (DISCOVERY_I2Cx_SCL_PIN | DISCOVERY_I2Cx_SDA_PIN);
GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
GPIO_InitStruct.Alternate = DISCOVERY_I2Cx_AF;
HAL_GPIO_Init(DISCOVERY_I2Cx_GPIO_PORT, &GPIO_InitStruct);

/* Enable the DMA clock */
DISCOVERY_EEPROM_DMA_CLK_ENABLE();

/* Configure the DMA channel for the EEPROM peripheral TX direction */
/* Configure the DMA channel */
hdma_tx.Instance = DISCOVERY_EEPROM_DMA_CHANNEL_TX;
/* Set the parameters to be configured */
hdma_tx.Init.Direction = DMA_MEMORY_TO_PERIPH;
hdma_tx.InitPeriphInc = DMA_PINC_DISABLE;
hdma_tx.Init.MemInc = DMA_MINC_ENABLE;
hdma_tx.InitPeriphDataAlignment = DMA_PDATAALIGN_BYTE;
hdma_tx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
hdma_tx.Init.Mode = DMA_NO
RMAL;
00344    hdma_tx.Init.Priority    = DMA_PRIORITY_VERY_HIGH;
00345
00346    /* Associate the initialized hdma_rx handle to the the husart handle*/
00347    __HAL_LINKDMA(hi2c, hdmatx, hdma_tx);
00348
00349    /* Configure the DMA Stream */
00350    HAL_DMA_Init(&hdma_tx);
00351
00352    /* Configure and enable DMA TX Channel interrupt */
00353    HAL_NVIC_SetPriority((IRQn_Type)(DISCOVERY_EEPROM_DMA_TX_IRQn), DISCOVERY_EEPROM_DMA_PREPrio,
00354                      DISCOVERY_EEPROM_DMA_SUBPrio);
00355    HAL_NVIC_EnableIRQ((IRQn_Type)(DISCOVERY_EEPROM_DMA_TX_IRQn));
00356
00357    /* Configure the DMA channel for the EEPROM peripheral RX direction */
00358    /* Configure the DMA channel */
00359    hdma_rx.Instance    = DISCOVERY_EEPROM_DMA_CHANNEL_RX;
00360    /* Set the parameters to be configured */
00361    hdma_rx.Init.Direction    = DMA_PERIPHER_TO_MEMORY;
00362    hdma_rx.Init.PeriphInc    = DMA_PERIPHER_DISABLE;
00363    hdma_rx.Init.MemInc    = DMA_MEMORY_DISABLE;
00364    hdma_rx.Init.PeriphDataAlignment    = DMA_PTRALIGNBYTE;
00365    hdma_rx.Init.MemDataAlignment    = DMA_MEMORY_ALIGN_BYTE;
00366    hdma_rx.Init.Mode    = DMA_NOPRINT;
hdma_rx.Init.Priority = DMA_PRIORITYVERY_HIGH;

/* Associate the initialized hdma_rx handle to the the usart handle*/
__HAL_LINKDMA(hi2c, hdmarx, hdma_rx);

/* Configure the DMA Stream */
HAL_DMA_Init(&hdma_rx);

/* Configure and enable DMA RX Channel interrupt */
HAL_NVIC_SetPriority((IRQn_Type)(DISCOVERY_EEPROM_DMA_RX_IRQn), DISCOVERY_EEPROM_DMA_PREPRIO, DISCOVERY_EEPROM_DMA_SUBPRIO);
HAL_NVIC_EnableIRQ((IRQn_Type)(DISCOVERY_EEPROM_DMA_RX_IRQn));

/* Enable 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.Timing = DISCOVERY_I2Cx_TIMING;
        I2cHandle.Init.OwnAddress1 = 0;
        I2cHandle.Init.AddressingMode = I2C_ADDRESSING_MODE_7;
        HAL_I2C_Init(&I2cHandle);
    }
}

DRESSINGMODE_7BIT;

00394  I2cHandle.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
00395  I2cHandle.Init.OwnAddress2 = 0;
00396  I2cHandle.Init.OwnAddress2Masks = I2C_OA2_NOMASK;
00397  I2cHandle.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
00398  I2cHandle.Init.NoStretchMode = I2C_NO_STRETCH_DISABLE;
00399
00400  /* Init the I2C */
00401  I2Cx_MspInit(&I2cHandle);
00402  HAL_I2C_Init(&I2cHandle);
00403  }
00404  }
00405
00406  /**
00407  * @brief Write a value in a register of the device through BUS.
00408  * @param Addr Device address on BUS Bus.
00409  * @param Reg The target register address to write
00410  * @param Value The target register value to be written
00411  * @retval None
00412  */
00413  static uint32_t I2Cx_WriteData(uint8_t Addr, uint16_t Reg, uint8_t Value)
00414  {
00415  HAL_StatusTypeDef status = HAL_OK;
00416
00417  status = HAL_I2C_Mem_Write(&I2cHandle, Addr, Reg, I2C_MEMADD_SIZE_16BIT, &Value, 1, I2Cx_TIM_EOUT_MAX);
00418
/* Check the communication status */
if(status != HAL_OK)
{
    /* Execute user timeout callback */
    I2Cx_Error(Addr);
    return HAL_ERROR;
}
return HAL_OK;

/**
 * @brief Write 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
 * @param pBuffer The target register value to be written
 * @param Length buffer size to be written
 * @retval None
 */
static uint32_t I2Cx_WriteBuffer(uint8_t Addr, uint16_t Reg, uint8_t *pBuffer, uint16_t Length)
{
    HAL_StatusTypeDef status = HAL_OK;
    status = HAL_I2C_Mem_Write_DMA(&I2cHandle, Addr, Reg, I2C_MEMADD_SIZE_16BIT, pBuffer, Length);
    if(status != HAL_OK)
    {
        /* Execute user timeout callback */
        I2Cx_Error(Addr);
return HAL_ERROR;
}
return HAL_OK;

/**
 * @brief Read a register of the device through BUS
 * @param Addr Device address on BUS
 * @param Reg The target register address to read
 * @retval Value: read register value
 */
static uint32_t I2Cx_ReadData(uint8_t Addr, uint16_t Reg, uint8_t *Value)
{
    HAL_StatusTypeDef status = HAL_OK;

    status = HAL_I2C_Mem_Read(&I2cHandle, Addr, Reg, I2C_MEMADD_SIZE_16BIT, Value, 1, I2Cx_TIMEO
UT_MAX);

    /* Check the communication status */
    if(status != HAL_OK)
    {
        /* Execute user timeout callback */
        I2Cx_Error(Addr);
        return HAL_ERROR;
    }
    return HAL_OK;
}

/**
 * @brief Reads multiple data on the BUS.
 * @param Addr I2C Address
 * @param Reg Reg Address
 * @param pBuffer pointer to read data bu
static uint32_t I2Cx_ReadBuffer(uint8_t Addr, uint16_t Reg, uint8_t *pBuffer, uint16_t Length) {
    HAL_StatusTypeDef status = HAL_OK;
    status = HAL_I2C_Mem_Read_DMA(&I2cHandle, Addr, Reg, I2C_MEMADD_SIZE_16BIT, pBuffer, Length);
    if (status != HAL_OK) {
        I2Cx_Error(Addr);
        return HAL_ERROR;
    }
    return HAL_OK;
}

/**
 * @brief Discovery I2Cx error treatment function
 * @retval None
 */
static void I2Cx_Error (uint8_t Addr) {
    /* De-initialize the I2C communication BUS */
    HAL_I2C_DeInit(&I2cHandle);
    /* Re-Initialize the I2C communication BUS */
I2Cx_Init();

#if defined(HAL_SPI_MODULE_ENABLED)
	/*******************************
SPI	Routine
s***********************************/
	/**
	* @brief SPI1 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:
l3gd20 SPI interface max baudrate is 10MHz for write/read
PCLK2 frequency is set to 90 MHz */
	SpiHandle.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_8;
	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 = SPIDATASIZE_8
 SpiHandle.Init.FirstBit = SPI_FIRSTBIT_MASTER;
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;
}
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 hspi SPI handle
* @retval None
*
static void SPIx_MspInit(SPI_HandleTypeDef * hspi)
{
    GPIO_InitTypeDef    GPIO_InitStructure;

    /* Enable SPI2 clock */
    DISCOVERY_SPIx_CLOCK_ENABLE();

    /* enable SPI2 gpio clock */
    DISCOVERY_SPIx_GPIO_CLK_ENABLE();

    /* configure SPI5 SCK, MOSI and MISO */
    GPIO_InitStructure.Pin = (DISCOVERY_SPIx_SCK_PIN | DISCOVERY_SPIx_MOSI_PIN | DISCOVERY_SPIx_
MISO_PIN);
00600    GPIO_InitStructure.Mode = GPIO_MODE_AF_PP;
00601    GPIO_InitStructure.Pull = GPIO_NOPULL; /*
GPIOPULLDOWN*/
00602    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_HIGH;
00603    GPIO_InitStructure.Alternate = DISCOVERY_SPIx_AF;
00604    HAL_GPIO_Init(DISCOVERY_SPIx_GPIO_PORT, &GPIO_InitStructure);
00605 }
00606
00607 /**
00608    * @}
00609    */
00610
00611 /** @addtogroup STM32F072B_DISCOVERY_LINK_Operations_Functions
00612    * @{
00613    */
00614
00615 /*********************************************************************************/
00616
00617 LINK OPERATIONS
00618 *********************************************************************************/
00619 /* LINK GYRO
00620 *********************************************************************************/
00621 /* @brief Configures the GYRO SPI interface.
00622    * @retval None
00623    */
00624 void GYRO_IO_Init(void)
00625 {
00626    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;
GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_HIGH;
GPIO_InitStructure.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GYRO_INT_GPIO_PORT, &GPIO_InitStructure);

SPIx_Init();

/**
 * @brief Writes one byte to the GYRO.
 * @param pBuffer pointer to the buffer containing the data to be written to the GYRO.
```c
void GYRO_IO_Write(uint8_t* pBuffer, uint8_t WriteAddr, uint16_t NumByteToWrite)
{
    /* Configure the MS bit:
      - When 0, the address will remain unchanged in multiple read/write commands.
      - When 1, the address will be auto incremented in multiple read/write commands.
    */
    if (NumByteToWrite > 0x01)
    {
        WriteAddr |= (uint8_t)MULTIPLEBYTE_CMD;
    }
    /* Set chip select Low at the start of the transmission */
    GYRO_CS_LOW();
    /* Send the Address of the indexed register */
    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 */
}```
transmission */
00683  GYRO_CS_HIGH();
00684 }
00685 
00686 /**
00687  * @brief  Reads a block of data from the GYROSCOPE.
00688  * @param pBuffer pointer to the buffer that receives the data read from the GYROSCOPE.
00689  * @param ReadAddr GYROSCOPE's internal address to read from.
00690  * @param NumByteToRead number of bytes to read from the GYROSCOPE.
00691  * @retval None
00692 */
00693 void GYRO_IO_Read(uint8_t* pBuffer, uint8_t ReadAddr, uint16_t NumByteToRead)
00694 {
00695   if(NumByteToRead > 0x01)
00696   {
00697     ReadAddr |= (uint8_t)(READWRITE_CMD | MULTIPLEBYTE_CMD);
00698   }
00699 else
00700   {
00701     ReadAddr |= (uint8_t)READWRITE_CMD;
00702   }
00703   /* Set chip select Low at the start of the transmission */
00704  GYRO_CS_LOW();
00705 
00706   /* Send the Address of the indexed register */
00707  SPIx_WriteRead(ReadAddr);
00708 
00709   /* 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 */

#if defined(HAL_I2C_MODULE_ENABLED)
/********************************** LINK I2C EEPROM **********************************/
/**
 * @brief Initializes peripherals used by the I2C EEPROM driver.
 * @retval None
 */
void EEPROM_IO_Init(void) {
    I2Cx_Init();
}
/**
 * @brief Write data to I2C EEPROM driver in using DMA channel
 * @param DevAddress Target device address
 * @param MemAddress Internal memory address
 * @param pBuffer Pointer to data buffer
 * @param BufferSize Amount of data to be sent
 */
uint32_t EEPROM_IO_WriteData(uint16_t DevAddress, uint16_t MemAddress, uint32_t pBuffer, uint32_t BufferSize)
{
    if (BufferSize == 1) {
        return (I2Cx_WriteData(DevAddress, MemAddress, *((uint8_t*)pBuffer)));
    } else {
        return (I2Cx_WriteBuffer(DevAddress, MemAddress, (uint8_t*)pBuffer, (uint32_t)BufferSize)) ;
    }
}

/**
 * @brief   Read data from I2C EEPROM driver in using DMA channel
 * @param   DevAddress Target device address
 * @param   MemAddress Internal memory address
 * @param   pBuffer Pointer to data buffer
 * @param   BufferSize Amount of data to be read
 * @retval  HAL status
 */

uint32_t EEPROM_IO_ReadData(uint16_t DevAddress, uint16_t MemAddress, uint32_t pBuffer, uint32_t BufferSize)
{
    if (BufferSize == 1) {
        return (I2Cx_ReadData(DevAddress, MemAdd
ress, (uint8_t*)pBuffer));
00767 } else
00768 { return (I2Cx_ReadBuffer(DevAddress, MemAddress, (uint8_t*)pBuffer, (uint32_t)BufferSize));
00771 }
00772 }
00773 */
00774 */ @brief Checks if target device is ready for communication.
00775 */ @note This function is used with Memory devices
00776 */ @param DevAddress Target device address
00777 */ @param Trials Number of trials
00778 */ @retval HAL status
00779 */
00780 */ HAL_StatusTypeDef EEPROM_IO_IsDeviceReady(uint16_t DevAddress, uint32_t Trials)
00781 { HAL_Delay(5);
00782 return (HAL_I2C_IsDeviceReady(&I2cHandle, DevAddress, Trials, I2Cx_TIMEOUT_MAX));
00785 }
00786 */Hal_I2C_MODULE_ENABLED */
00787 */
00788 */ @
00789 */ /
00790 */
00791 */
00792 */ @
00793 */ /
00794 */
00795 */
00796 */ @
00797 /*
/**
 * @file    stm32f072b_discovery_eeprom.h
 * @author  MCD Application Team
 * @brief   This file contains all the functions prototypes for
 *          the stm32072b_discovery_eeprom.c firmware driver.
 *
 * @attention
 *
 * <h2><center>&copy; COPYRIGHT(c) 2016 STM microelectronics</center></h2>
 * 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.

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

#ifndef __STM32072B_DISCOVERY_EEPROM_H
#define __STM32072B_DISCOVERY_EEPROM_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes -----------------------------------------------
-----------------------------------------------*/
#include "stm32f072b_discovery.h"

/** @addtogroup BSP
@{ */

/** @addtogroup STM32F072B_DISCOVERY
@{ */

/** @defgroup STM32F072B_DISCOVERY_EEPROM STM32F072B_DISCOVERY EEPROM
@{ */

#define EEPROM_PAGESIZE             4
#define EEPROM_MAX_SIZE             0x2000 / 64Kbit*/

/** @defgroup STM32F072B_DISCOVERY_EEPROM_Exported_Constants Exported Constants
@{ */

/** EEPROM hardware address and page size */
#define EEPROM_PAGESIZE 4
#define EEPROM_MAX_SIZE 0x2000
*/
/* Maximum Timeout values for flags and events waiting loops.
This timeout is based on systick set to 1ms*/

#define EEPROM_LONG_TIMEOUT ((uint32_t)(1000))

#define EEPROM_MAX_TRIALS_NUMBER 300

#define EEPROM_OK 0
#define EEPROM_FAIL 1
#define EEPROM_TIMEOUT 2

/** USER Callbacks: This function is declared
as __weak in EEPROM driver and
should be implemented into user application.
BSP_EEPROM_TIMEOUT_UserCallback() function is called whenever a timeout condition
occurs during communication (waiting on an event that doesn't occur, bus
events, busy devices ...). */
__weak uint32_t BSP_EEPROM_TIMEOUT_UserCallback(void);
/* Link function for I2C EEPROM peripheral */

void EEPROM_IO_Init(void);
uint32_t EEPROM_IO_WriteData(uint16_t DevAddress, uint16_t MemAddress, uint32_t pBuffer, uint32_t BufferSize);
uint32_t EEPROM_IO_ReadData(uint16_t DevAddress, uint16_t MemAddress, uint32_t pBuffer, uint32_t BufferSize);
HAL_StatusTypeDef EEPROM_IO_IsDeviceReady(uint16_t DevAddress, uint32_t Trials);
/**
 * @}*/
/**
 * @}*/
/**
 * @}*/
/**
 * @}*/
*/

#ifdef __cplusplus
} #endif
#endif /* __STM32072B_DISCOVERY_EEPROM_H */
#endif

/****************************
(C) COPYRIGHT STMicroelectronics
*****END OF FILE*****/
/**
 * @file    stm32f072b_discovery_eeprom.c
 * @author  MCD Application Team
 * @brief   This file provides a set of functions needed to manage an I2C M24LR64 EEPROM memory.

 * Notes:
 * - This driver is intended for stm32F0xx families devices only.
 * - The I2C EEPROM memory (M24LR64) is available on RF EEPROM daughter board (ANT7-M24LR-A) provided with the STM32F072B-Discovery board.
 * To use this driver you have to connect the ANT7-M24LR-A to CN3 connector.

 * It implements a high level comm
unication layer for read and write

00017 * from/to this memory. The needed
STM32F0xx hardware resources (I2C and
00018 * GPIO) are defined in stm32f072b
_discovery.h file, and the initialization is
00019 * performed in EEPROM_IO_Init() f
unction declared in stm32f072b_discovery.c
00020 * file.
00021 * You can easily tailor this driv
er to any other development board,
00022 * by just adapting the defines fo
r hardware resources and
00023 * EEPROM_IO_Init() function.
00024 *
00025 * @note In this driver, basic rea
d and write functions (EEPROM_ReadBuffer()  
00026 * and EEPROM_WritePage()) u
se Polling mode to perform the data transfer
00027 * to/from EEPROM memory.
00028 *
00029 * +-----------------------------------
------------------------------+
00030 * | Pin assignment for M
24LR64 EEPROM |
00031 * +-----------------------------------+
00032 * | STM32F0xx I2C Pins
| EEPROM | Pin |
00033 * +-----------------------------------+
00034 * |
| E0(GND) | 1 (0V) |
00035 * |
| AC0 | 2 |
00036 * |
| AC1 | 3 |
00037 * |
<table>
<thead>
<tr>
<th>VSS</th>
<th>4 (0V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDA</td>
<td>5</td>
</tr>
<tr>
<td>SCL</td>
<td>6</td>
</tr>
<tr>
<td>E1(GND)</td>
<td>7 (0V)</td>
</tr>
<tr>
<td>VDD</td>
<td>8 (3.3V)</td>
</tr>
</tbody>
</table>

---

@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, 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.

******************************************
************************************

#include "stm32f072b_discovery_eeprom.h"

/** @addtogroup BSP */
/**
** Includes ----------------------------------
"------------------------------------------*/

#include "stm32f072b_discovery_eeprom.h"

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

/** @addtogroup STM32F072B_DISCOVERY */
/** @{*/
/**
 * @addtogroup STM32F072B_DISCOVERY_EEPROM
 *
 * This file includes the I2C EEPROM driver of STM32072B-EVAL board.
 */

/**
 * @defgroup STM32072B_DISCOVERY_EEPROM_Private_Variables Private Variables
 */

__IO uint16_t EEPROMAddress = 0;
__IO uint32_t EEPROMTimeout = EEPROM_LONG_TIMEOUT;
__IO uint16_t EEPROMDataRead;
__IO uint8_t* EEPROMDataWritePointer;
__IO uint8_t EEPROMDataNum;

/**
 * @addtogroup STM32F072B_DISCOVERY_EEPROM_Exported_Functions
 */

/**
 * @brief Initializes peripherals used by the I2C EEPROM driver.
 *
 * @note There are 2 different versions of M24LR64 (A01 & A02).
 * Then try to connect on 1st one (EEPROM_I2C_ADDRESS_A01)
 * and if problem, check the 2nd one (EEPROM_I2C_ADDRESS_A02)
 */
uint32_t BSP_EEPROM_Init(void) {
    /* I2C Initialization */
    EEPROM_IO_Init();
    /* Select the EEPROM address for A01 and check if OK*/
    EEPROMAddress = DISCOVERY_EEPROM_I2C_ADDRESS_A01;
    if (EEPROM_IO_IsDeviceReady(EEPROMAddress, EEPROM_MAX_TRIALS_NUMBER) != HAL_OK) {
        return EEPROM_FAIL;
    }
    return EEPROM_OK;
}

/**
 * @brief Reads a block of data from the EEPROM.
 * @param pBuffer pointer to the buffer that receives the data read from
 * the EEPROM.
 * @param ReadAddr EEPROM's internal address to start reading from.
 * @param NumByteToRead pointer to the variable holding number of bytes to
 * be read from the EEPROM.
 * @return EEPROM_OK if operation is correctly performed, else return value
 * different from EEPROM_OK (0)
 */
variable in order know when the transfer is complete.

@retval EEPROM_OK (0) if operation is correctly performed, else return value different from EEPROM_OK (0) or the timeout user callback.

```c
uint32_t BSP_EEPROM_ReadBuffer(uint8_t* pBuffer, uint16_t ReadAddr, uint16_t* NumByteToRead) {
    uint32_t buffersize = *NumByteToRead;
    EEPROMDataRead = *NumByteToRead;

    if (EEPROM_IO_ReadData(EEPROMAddress, ReadAddr, (uint32_t) pBuffer, buffersize) != EEPROM_OK ) {
        return EEPROM_FAIL;
    } else {
        EEPROMTimeout = HAL_GetTick();
        while (EEPROMDataRead > 0) {
            if((HAL_GetTick() - EEPROMTimeout) < EEPROM_LONG_TIMEOUT)
```
BSP_EEPROM_TIMEOUT_UserCallback();

return EEPROM_TIMEOUT;

} /* If all operations OK, return EEPROM_OK (0) */

return EEPROM_OK;

} /*
If all operations OK, return EEPROM_OK (0) */

/**
 * @brief Writes more than one byte to the EEPROM with a single WRITE cycle.
 * @note The number of bytes (combined to write start address) must not
 * cross the EEPROM page boundary. This function can only write into
 * the boundaries of an EEPROM page.

 * This function doesn't check on boundaries condition (in this driver
 * the function BSP_EEPROM_WriteBuffer() which calls BSP_EEPROM_WritePage() is
 * responsible of checking on Page boundaries).

 * @param pBuffer pointer to the buffer containing the data to be written to
 * the EEPROM.

 * @param WriteAddr EEPROM's internal address to write to.

 * @param NumByteToWrite pointer to the variable holding number of bytes to
 * be written into the EEPROM.

 * @param...
The variable pointed by NumByteToWrite is reset to 0 when all the data are written to the EEPROM. Application should monitor this variable in order to know when the transfer is complete.

This function just configure the communication and enable the DMA channel to transfer data. Meanwhile, the user application may perform other tasks in parallel.

@retval EEPROM_OK (0) if operation is correctly performed, else return value different from EEPROM_OK (0) or the timeout user callback.

/* Set the pointer to the Number of data to be written. This pointer will be used by the DMA Transfer Completer interrupt Handler in order to reset the variable to 0. User should check on this variable in order to know if the DMA transfer has been complete or not. */

EEPROMDataWritePointer = NumByteToWrite;
status = EEPROM_IO_WriteData(EEPROMAddress, WriteAddr, (uint32_t) pBuffer, buffersize);
/* If all operations OK, return EEPROM_OK */
(0) */
00213    return status;
00214 }
00215
00216 /**
00217     * @brief Writes buffer of data to the I2C EEPROM.
00218     * @param pBuffer pointer to the buffer containing the data to be written
00219     *         to the EEPROM.
00220     * @param WriteAddr EEPROM's internal address to write to.
00221     * @param NumByteToWrite number of bytes to write to the EEPROM.
00222     * @retval EEPROM_OK (0) if operation is correctly performed, else return value
00223     *         different from EEPROM_OK (0) or the timeout user callback.
00224  */
00225 uint32_t BSP_EEPROM_WriteBuffer(uint8_t* pBuffer, uint16_t WriteAddr, uint16_t NumByteToWrite)
00226 {
00227     uint16_t numofpage = 0, numofsingle = 0, count = 0;
00228     uint16_t addr = 0;
00229
00230     addr = WriteAddr % EEPROM_PAGESIZE;
00231     count = EEPROM_PAGESIZE - addr;
00232     numofpage = NumByteToWrite / EEPROM_PAGESIZE;
00233     numofsingle = NumByteToWrite % EEPROM_PAGESIZE;
00234
00235     /*!
00236     * @brief If WriteAddr is EEPROM_PAGESIZE aligned
00237     */
00238     if(addr == 0)
00239     {

/!*< If NumByteToWrite < EEPROM_PAGESIZE */
if(numofpage == 0)
{
    /* Store the number of data to be written */
    EEPROMDataNum = numofsingle;
    /* Start writing data */
    if (BSP_EEPROM_WritePage(pBuffer, WriteAddr, (uint8_t*)&EEPROMDataNum)) != EEPROM_OK)
    {
        return EEPROM_FAIL;
    }
}
/* Wait transfer through DMA to be complete */
EEPROMTimeout = HAL_GetTick();
while (EEPROMDataNum > 0)
{
    if((HAL_GetTick() - EEPROMTimeout) < EEPROM_LONG_TIMEOUT) {
        BSP_EEPROM_TIMEOUT_UserCallback();
        return EEPROM_TIMEOUT;
    }
}
if (BSP_EEPROM_WaitEepromStandbyState()) != EEPROM_OK) return EEPROM_FAIL;

/*<! If NumByteToWrite > EEPROM_PAGESIZE */
else
{
    while(numofpage--)
    {
        /* Store the number of data to be written */
        EEPROMDataNum = EEPROM_PAGESIZE;
    }
}
if (BSP_EEPROM_WritePage(pBuffer, WriteAddr, (uint8_t*)(&EEPROMDataNum)) != EEPROM_OK)
{
    return EEPROM_FAIL;
}

/* Wait transfer through DMA to be complete */
EEPROMTimeout = HAL_GetTick();
while (EEPROMDataNum > 0)
{
    if(((HAL_GetTick() - EEPROMTimeout) < EEPROM_LONG_TIMEOUT) {
        BSP_EEPROM_TIMEOUT_UserCallback();
        return EEPROM_TIMEOUT;
    }
}

if (BSP_EEPROM_WaitEepromStandbyState() != EEPROM_OK) return EEPROM_FAIL;

WriteAddr += EEPROM_PAGESIZE;
pBuffer += EEPROM_PAGESIZE;

if(numofsingle!=0)
{
    /* Store the number of data to be written */
    EEPROMDataNum = numofsingle;

    if (BSP_EEPROM_WritePage(pBuffer, WriteAddr, (uint8_t*)(&EEPROMDataNum)) != EEPROM_OK)
    {
        return EEPROM_FAIL;
    }
}

/* Wait transfer through DMA to be c
00293     EEPROMTimeout = HAL_GetTick();
00294     while (EEPROMDataNum > 0)
00295         {
00296             if((HAL_GetTick() - EEPROMTimeout) < EEPROM_LONG_TIMEOUT) {
00297                 BSP_EEPROM_TIMEOUT_UserCallback();
00298                 return EEPROM_TIMEOUT;
00299             }
00300         }
00301     if (BSP_EEPROM_WaitEepromStandbyState () != EEPROM_OK) return EEPROM_FAIL;
00302 }
00303 }
00304 }
00305 /*!< If WriteAddr is not EEPROM_PAGESIZE aligned */
00306 else
00307 {
00308     /*!< If NumByteToWrite < EEPROM_PAGESIZE */
00309         if(numofpage== 0)
00310             {
00311                 /*!< If the number of data to be written is more than the remaining space
00312                     in the current page: */
00313                 if (NumByteToWrite > count)
00314                     {
00315                     /* Store the number of data to be written */
00316                     EEPROMDataNum = count;
00317                 /*!< Write the data contained in same page */
00318                     if (BSP_EEPROM_WritePage(pBuffer, WriteAddr, (uint8_t*)(&EEPROMDataNum)) != EEPROM_OK) 
00319                         {
00320                             

00320    return EEPROM_FAIL;
00321 } /* Wait transfer through DMA to be complete */
00322 EEPROMTimeout = HAL_GetTick();
00323 while (EEPROMDataNum > 0)
00324 {
00325      if((HAL_GetTick() - EEPROMTimeout) < EEPROM_LONG_TIMEOUT) {
00326          BSP_EEPROM_TIMEOUT_UserCallback();
00327          return EEPROM_TIMEOUT;
00328      }
00329  }
00330 } /* Store the number of data to be written */
00331 EEPROMDataNum = (NumByteToWrite - count);
00332 /*!< Write the remaining data in the following page */
00333 if (BSP_EEPROM_WritePage((uint8_t*)((pBuffer + count), (WriteAddr + count), (uint8_t*)(&EEPROMDataNum)) != EEPROM_OK)
00334 {
00335     return EEPROM_FAIL;
00336  }
00337  /* Wait transfer through DMA to be complete */
00338 EEPROMTimeout = HAL_GetTick();
00339 while (EEPROMDataNum > 0)
00340 {
00341      if((HAL_GetTick() - EEPROMTimeout) < EEPROM_LONG_TIMEOUT) {
00342          BSP_EEPROM_TIMEOUT_UserCallback(
00346    return EEPROM_TIMEOUT;
00347  }
00348  }
00349  if (BSP_EEPROM_WaitEepromStandbyState()
00350  ) != EEPROM_OK) return EEPROM_FAIL;
00351  }
00352  else
00353  { /* Store the number of data to be written */
00354    EEPROMDataNum = numofsingle;
00355  
00356  if (BSP_EEPROM_WritePage(pBuffer, WriteAddr, (uint8_t*)(&EEPROMDataNum))
00357  ) != EEPROM_OK)
00358  { return EEPROM_FAIL;
00359  }
00360  /* Wait transfer through DMA to be complete */
00361    EEPROMTimeout = HAL_GetTick();
00362  while (EEPROMDataNum > 0)
00363  { if((HAL_GetTick() - EEPROMTimeout)
00364  < EEPROM_LONG_TIMEOUT) {
00365    BSP_EEPROM_TIMEOUT_UserCallback();
00366    return EEPROM_TIMEOUT;
00367  }
00368  
00369  if (BSP_EEPROM_WaitEepromStandbyState()
00370  ) != EEPROM_OK) return EEPROM_FAIL;
00371  }
00372  else
00373  /*! If NumByteToWrite > EEPROM_PAGESIZE */
00374  /*
{  
    NumByteToWrite -= count;
    numofpage = NumByteToWrite / EEPROM_PAGESIZE;
    numofsingle = numofsingle % EEPROM_PAGESIZE;
    
    if(count != 0) {
        /* Store the number of data to be written */
        EEPROMDataNum = count;
        if (BSP_EEPROM_WritePage(pBuffer, WriteAddr, (uint8_t*)(&EEPROMDataNum)) != EEPROM_OK) {
            return EEPROM_FAIL;
        }
        /* Wait transfer through DMA to be complete */
        EEPROMTimeout = HAL_GetTick();
        while (EEPROMDataNum > 0) {
            if(((HAL_GetTick() - EEPROMTimeout) < EEPROM_LONG_TIMEOUT) {
                BSP_EEPROM_TIMEOUT_UserCallback();
                return EEPROM_TIMEOUT;
            }
        }
        if (BSP_EEPROM_WaitEepromStandbyState() != EEPROM_OK) return EEPROM_FAIL;
    }
    WriteAddr += count;
    pBuffer += count;
    }
    
    while(numofpage--) {
    }
/* Store the number of data to be written */
EEPROMDataNum = EEPROM_PAGESIZE;

if (BSP_EEPROM_WritePage(pBuffer, WriteAddr, (uint8_t*)(&EEPROMDataNum)) != EEPROM_OK) {
    return EEPROM_FAIL;
}

/* Wait transfer through DMA to be complete */
EEPROMTimeout = HAL_GetTick();
while (EEPROMDataNum > 0) {
    if((HAL_GetTick() - EEPROMTimeout) < EEPROM_LONG_TIMEOUT) {
        BSP_EEPROM_TIMEOUT_UserCallback();
        return EEPROM_TIMEOUT;
    }
}

if (BSP_EEPROM_WaitEepromStandbyState() != EEPROM_OK) return EEPROM_FAIL;

WriteAddr += EEPROM_PAGESIZE;
pBuffer += EEPROM_PAGESIZE;

if(numofsingle != 0) {
    /* Store the number of data to be written */
    EEPROMDataNum = numofsingle;

    if (BSP_EEPROM_WritePage(pBuffer, WriteAddr, (uint8_t*)(&EEPROMDataNum)) != EEPROM_OK) {
        return EEPROM_FAIL;
    }
}
/* Wait transfer through DMA to be complete */
EEPROMTimeout = HAL_GetTick();
while (EEPROMDataNum > 0)
{
    if((HAL_GetTick() - EEPROMTimeout) < EEPROM_LONG_TIMEOUT) {
        BSP_EEPROM_TIMEOUT_UserCallback();
        return EEPROM_TIMEOUT;
    }
}
if (BSP_EEPROM_WaitEepromStandbyState() != EEPROM_OK) return EEPROM_FAIL;

/* If all operations OK, return EEPROM_OK (0) */
return EEPROM_OK;

/**
 * @brief Wait for EEPROM Standby state.
 *
 * @note This function allows to wait and check that EEPROM has finished the last operation. It is mostly used after Write operation: after receiving the buffer to be written, the EEPROM may need additional time to actually perform the write operation. During this time, it doesn't answer to I2C packets addressed to it. Once the write operation is complete
the EEPROM responds to its address.

* @retval EEPROM_OK (0) if operation is correctly performed, else return value different from EEPROM_OK (0) or the timeout user callback.

*/

uint32_t BSP_EEPROM_WaitEepromStandbyState( void)
{
    HAL_StatusTypeDef status;
    EEPROMTimeout = HAL_GetTick();

    do {
        if((HAL_GetTick() - EEPROMTimeout) < EEPROM_LONG_TIMEOUT)
        {
            BSP_EEPROM_TIMEOUT_UserCallback();
            return EEPROM_TIMEOUT;
        }
        status = EEPROM_IO_IsDeviceReady(EEPROMAddress, EEPROM_MAX_TRIALS_NUMBER);
    }while (status == HAL_BUSY);

    /* Check if the maximum allowed number of trials has been reached */
    if (status != HAL_OK)
    {
        /* If the maximum number of trials has been reached, exit the function */
        BSP_EEPROM_TIMEOUT_UserCallback();
        return EEPROM_FAIL;
    }
}
return EEPROM_OK;
}

/**
 * @brief Memory Tx Transfer completed call backs.
 * @param hi2c I2C handle
 * @retval None
 */
void HAL_I2C_MemTxCpltCallback(I2C_HandleTypeDef *hi2c) {

*EEPROMDataWritePointer = 0;
}

/**
 * @brief Memory Rx Transfer completed call backs.
 * @param hi2c I2C handle
 * @retval None
 */
void HAL_I2C_MemRxCpltCallback(I2C_HandleTypeDef *hi2c) {

EEPROMDataRead = 0;
}

/**
 * @brief Basic management of the timeout situation.
 * @retval None
 */
__weak uint32_t BSP_EEPROM_TIMEOUT_UserCallback(void) {

/* Block communication and all processes */
while (1) {
}
/**
 * @}
 */
/**
 * @}
 */
/**
 * @}
 */
/**
 * @}
 */
/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
stm32f072b_discovery_gyroscope.h

Go to the documentation of this file.

```c
/**
 * @file    stm32f072b_discovery_gyroscope.h
 *
 * @author  MCD Application Team
 * @brief   This file contains definitions for stm32f072b_discovery_gyroscope.c
 *          firmware driver.
 * @attention
 *
 * <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,
 *    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 __STM32F072B_DISCOVERY_GYRO_H
#define __STM32F072B_DISCOVERY_GYRO_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes -----------------------------------------------*/
#include "stm32f072b_discovery.h"

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

/** @addtogroup BSP *
@{ */

/** @addtogroup STM32F072B_DISCOVERY *
@{ */

/** @defgroup STM32F072B_DISCOVERY_GYRO STM32F072B_DISCOVERY_GYROSCOPE *
@{ */

/** @defgroup STM32F072B_DISCOVERY_GYRO_Expo Exported Types *
@{ */
typedef enum {

GYRO_OK = 0,

*/

/** @defgroup STM32F072B_DISCOVERY_GYRO_Expo Exported Types */
@endgroup */

/** @addtogroup STM32F072B_DISCOVERY */
@endgroup */

/** @addtogroup BSP */
@endgroup */

#endif

*/
GYRO_ERROR = 1,
GYRO_TIMEOUT = 2
}

/* 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_InterruptTypeDef *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
#ifdef __STM32F072B_DISCOVERY_GYROSCOPE_H
*/
#endif
/****************************************************************************
(C) COPYRIGHT STMicroelectronics *****END OF FILE*****/
stm32f072b_discovery_gyroscope.c

Go to the documentation of this file.

```c
/**
 * @file    stm32f072b_discovery_gyroscope.c
 *
 * @author  MCD Application Team
 *
 * @brief   This file provides a set of functions needed to manage the l3gd20 MEMS accelerometer available on
 *          STM32F072B-Discovery Kit.
 *
 * @attention
 *
 * <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,
 *    this list of conditions and the fol
lowing 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.
#include "stm32f072b_discovery_gyroscope.h"

/**
 * @addtogroup BSP
 *
 */

/**
 * @addtogroup STM32F072B_DISCOVERY
 *
 */

/**
 * @addtogroup STM32F072B_DISCOVERY_GYRO
 *
 */

/**
 * @defgroup STM32F072B_DISCOVERY_GYRO_Private_Variables Private Variables
 *
 */

static GYRO_DrvTypeDef *GyroscopeDrv;

/**
 * @defgroup STM32F072B_DISCOVERY_GYRO_Exported_Functions
 *
 */

/**
 * @brief Set GYRO Initialization.
 * @retval GYRO_OK if no problem during initialization
 */
```c
uint8_t BSP_GYRO_Init(void)
{
    uint8_t ret = GYRO_ERROR;
    uint16_t ctrl = 0x0000;
    GYRO_InitTypeDef L3GD20_InitStructure;
    GYRO_FilterConfigTypeDef L3GD20_FilterStructure={0,0};

    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_FULLSCALE_500;

        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 | \nL3GD20_InitStructure.Full_Scale) << 8);

/* L3gd20 Init */
GyroscopeDrv->Init(ctrl);

L3GD20_FilterStructure.HighPassFilter_Mode_Selection =L3GD20_HPM_NORMAL_MODE_RES;
L3GD20_FilterStructure.HighPassFilter_CutOff_Frequency = L3GD20_HPFCF_0;

ctrl = (uint8_t)((L3GD20_FilterStructure.HighPassFilter_Mode_Selection | \nL3GD20_FilterStructure.HighPassFilter_CutOff_Frequency));

GyroscopeDrv->FilterConfig(ctrl) ;
GyroscopeDrv->FilterCmd(L3GD20_HIGHPASSFILTER_ENABLE);

ret = GYRO_OK;
}
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 GYRO
 * @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)
typedef *pIntConfig)
00153 {
00154  uint16_t interruptconfig = 0x0000;
00155
00156  if (GyroscopeDrv->ConfigIT != NULL)
00157  {
00158    /* Configure latch Interrupt request and
     * axe interrupts */
00159    interruptconfig |= ((uint8_t)(pIntConfig
00160    ->Latch_Request | \n00161    pIntConfig
00162    ->Interrupt_Axes) << 8);
00163
00164    GyroscopeDrv->ConfigIT(interruptconfig);
00165  }
00166 }
00167
00168 /**
00169  * @brief Enable INT1 or INT2 interrupt
00170  * @param IntPin Interrupt pin
00171  *   This parameter can be:
00172  *     @arg L3GD20_INT1
00173  *     @arg L3GD20_INT2
00174  * @retval None
00175 */
00176 void BSP_GYRO_EnableIT(uint8_t IntPin)
00177 {
00178  if (GyroscopeDrv->EnableIT != NULL)
00179  {
00180    GyroscopeDrv->EnableIT(IntPin);
00181  }
00182 }
00183
/**
 * @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);
    }
}
## STM32F072B-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></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Modules**
STM32F072B-Discovery BSP User Manual

<table>
<thead>
<tr>
<th>Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F072B_DISCOVERY</td>
</tr>
</tbody>
</table>
# Modules

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F072B_DISCOVERY Common</td>
<td>This file provides firmware functions to manage Leds, push-buttons, and temperature sensor (LM75) available on STM32F072B-DISCO evaluation board from STMicroelectronics.</td>
</tr>
<tr>
<td>STM32F072B_DISCOVERY EEPROM</td>
<td>This file includes the I2C EEPROM driver of STM32072B-EVAL board.</td>
</tr>
<tr>
<td>STM32F072B_DISCOVERY GYROSCOPE</td>
<td>Generated on Wed Jul 5 2017 09:43:19 for STM32F072B-Discovery BSP User Manual by doxygen 1.7.6.1</td>
</tr>
</tbody>
</table>
STM32F072B-Discovery BSP User Manual

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

**STM32F072B_DISCOVERY Common**

This file provides firmware functions to manage Leds, push-buttons, and temperature sensor (LM75) available on STM32F072B-DISCO evaluation board from STMicroelectronics. More...
## Modules

<table>
<thead>
<tr>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private Constants</td>
</tr>
<tr>
<td>Private Variables</td>
</tr>
<tr>
<td>Bus Operations functions</td>
</tr>
<tr>
<td>Link Operations functions</td>
</tr>
<tr>
<td>Exported Types</td>
</tr>
<tr>
<td>Exported Constants</td>
</tr>
<tr>
<td>Exported Functions</td>
</tr>
</tbody>
</table>
Detailed Description

This file provides firmware functions to manage Leds, push-buttons, and temperature sensor (LM75) available on STM32F072B-DISCO evaluation board from STMicroelectronics.
### Exported Constants

<table>
<thead>
<tr>
<th>STM32F072B_DISCOVERY Common</th>
<th>Modules</th>
<th>Files</th>
<th>Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modules</td>
<td>Modules</td>
<td>Files</td>
<td>Directories</td>
</tr>
</tbody>
</table>

**STM32F072B-Discovery BSP User Manual**
# Modules

<table>
<thead>
<tr>
<th>MODULE NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32F072B_DISCOVERY LED</td>
<td>Define for STM32F072B_DISCOVERY board.</td>
</tr>
<tr>
<td>STM32F072B_DISCOVERY BUTTON</td>
<td></td>
</tr>
<tr>
<td>STM32F072B_DISCOVERY BUS</td>
<td></td>
</tr>
<tr>
<td>STM32F072B_DISCOVERY COMPONENT</td>
<td></td>
</tr>
</tbody>
</table>

Generated on Wed Jul 5 2017 09:43:19 for STM32F072B-Discovery BSP User Manual by doxygen 1.7.6.1
STM32F072B-Discovery BSP User Manual

STM32F072B_DISCOVERY EEPROM

STM32F072B_DISCOVERY

This file includes the I2C EEPROM driver of STM32072B-EVAL board. More...
## Modules

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Private Variables</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Exported Constants</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Exported Functions</strong></td>
<td></td>
</tr>
</tbody>
</table>
Detailed Description

This file includes the I2C EEPROM driver of STM32072B-EVAL board.
STM32F072B-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>Modules</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

STM32F072B_DISCOVERY GYROSCOPE

STM32F072B_DISCOVERY
<table>
<thead>
<tr>
<th>Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private Variables</td>
</tr>
<tr>
<td>Exported Types</td>
</tr>
<tr>
<td>Exported Functions</td>
</tr>
</tbody>
</table>

Generated on Wed Jul 5 2017 09:43:20 for STM32F072B-Discovery
BSP User Manual by doxygen 1.7.6.1