B-L475E-IOT01 BSP User Manual

LOW LEVEL Private Function Prototypes

LOW LEVEL

Generated on Thu Mar 16 2017 10:38:33 for B-L475E-IOT01 BSP User Manual by doxygen 1.7.6.1

B-L475E-IOT01 BSP User Manual

QSPI_Info Struct Reference

QSPI Exported Types

```
#include <stm32l475e_iot01_qspi.h>
```

Data Fields

uint32_t	**FlashSize**
uint32_t	**EraseSectorSize**
uint32_t	**EraseSectorsNumber**
uint32_t	**ProgPageSize**
uint32_t	**ProgPagesNumber**

Detailed Description

Definition at line **83** of file **stm32l475e_iot01_qspi.h**.

Field Documentation

uint32_t QSPI_Info::EraseSectorSize

Size of sectors for the erase operation

Definition at line **85** of file **stm32l475e_iot01_qspi.h**.

Referenced by **BSP_QSPI_GetInfo()**.

uint32_t QSPI_Info::EraseSectorsNumber

Number of sectors for the erase operation

Definition at line **86** of file **stm32l475e_iot01_qspi.h**.

Referenced by **BSP_QSPI_GetInfo()**.

uint32_t QSPI_Info::FlashSize

Size of the flash

Definition at line **84** of file **stm32l475e_iot01_qspi.h**.

Referenced by **BSP_QSPI_GetInfo()**.

uint32_t QSPI_Info::ProgPageSize

Size of pages for the program operation

Definition at line **87** of file **stm32l475e_iot01_qspi.h**.

Referenced by **BSP_QSPI_GetInfo()**.

uint32_t QSPI_Info::ProgPagesNumber

Number of pages for the program operation

Definition at line **88** of file **stm32l475e_iot01_qspi.h**.

Referenced by **BSP_QSPI_GetInfo()**.

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

- **stm32l475e_iot01_qspi.h**

B-L475E-IOT01 BSP User Manual

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- EraseSectorSize : **QSPI_Info**
- EraseSectorsNumber : **QSPI_Info**
- FlashSize : **QSPI_Info**
- ProgPageSize : **QSPI_Info**
- ProgPagesNumber : **QSPI_Info**

B-L475E-IOT01 BSP User Manual

- EraseSectorSize : **QSPI_Info**
- EraseSectorsNumber : **QSPI_Info**
- FlashSize : **QSPI_Info**
- ProgPageSize : **QSPI_Info**
- ProgPagesNumber : **QSPI_Info**

B-L475E-IOT01 BSP User Manual

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

- _ -

- __STM32L475E_IOT01_BSP_VERSION : **stm32l475e_iot01.c**
- __STM32L475E_IOT01_BSP_VERSION_MAIN : **stm32l475e_iot01.c**
- __STM32L475E_IOT01_BSP_VERSION_RC : **stm32l475e_iot01.c**
- __STM32L475E_IOT01_BSP_VERSION_SUB1 : **stm32l475e_iot01.c**
- __STM32L475E_IOT01_BSP_VERSION_SUB2 : **stm32l475e_iot01.c**

B-L475E-IOT01 BSP User Manual

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

- a -

- ACCELERO_ERROR : **stm32l475e_iot01_accelero.h**
- ACCELERO_OK : **stm32l475e_iot01_accelero.h**
- ACCELERO_StatusTypeDef : **stm32l475e_iot01_accelero.h**
- ACCELERO_TIMEOUT : **stm32l475e_iot01_accelero.h**
- AccelerometerDrv : **stm32l475e_iot01_accelero.c**

B-L475E-IOT01 BSP User Manual

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

- b -

- BSP_ACCELERO_AccGetXYZ() : **stm32l475e_iot01_accelero.c** , **stm32l475e_iot01_accelero.h**
- BSP_ACCELERO_DeInit() : **stm32l475e_iot01_accelero.h** , **stm32l475e_iot01_accelero.c**
- BSP_ACCELERO_Init() : **stm32l475e_iot01_accelero.c** , **stm32l475e_iot01_accelero.h**
- BSP_ACCELERO_LowPower() : **stm32l475e_iot01_accelero.h** , **stm32l475e_iot01_accelero.c**
- BSP_COM_DeInit() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_COM_Init() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_GetVersion() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_GYRO_DeInit() : **stm32l475e_iot01_gyro.h** , **stm32l475e_iot01_gyro.c**
- BSP_GYRO_GetXYZ() : **stm32l475e_iot01_gyro.c** , **stm32l475e_iot01_gyro.h**
- BSP_GYRO_Init() : **stm32l475e_iot01_gyro.c** , **stm32l475e_iot01_gyro.h**
- BSP_GYRO_LowPower() : **stm32l475e_iot01_gyro.c** , **stm32l475e_iot01_gyro.h**
- BSP_HSENSOR_Init() : **stm32l475e_iot01_hsensor.c** , **stm32l475e_iot01_hsensor.h**
- BSP_HSENSOR_ReadHumidity() : **stm32l475e_iot01_hsensor.c**

, **stm32l475e_iot01_hsensor.h**
- BSP_HSENSOR_ReadID() : **stm32l475e_iot01_hsensor.c** , **stm32l475e_iot01_hsensor.h**
- BSP_LED_DeInit() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_LED_Init() : **stm32l475e_iot01.h** , **stm32l475e_iot01.c**
- BSP_LED_Off() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_LED_On() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_LED_Toggle() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_MAGNETO_DeInit() : **stm32l475e_iot01_magneto.c** , **stm32l475e_iot01_magneto.h**
- BSP_MAGNETO_GetXYZ() : **stm32l475e_iot01_magneto.c** , **stm32l475e_iot01_magneto.h**
- BSP_MAGNETO_Init() : **stm32l475e_iot01_magneto.c** , **stm32l475e_iot01_magneto.h**
- BSP_MAGNETO_LowPower() : **stm32l475e_iot01_magneto.c** , **stm32l475e_iot01_magneto.h**
- BSP_PB_DeInit() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_PB_GetState() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_PB_Init() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_PSENSOR_Init() : **stm32l475e_iot01_psensor.c** , **stm32l475e_iot01_psensor.h**
- BSP_PSENSOR_ReadID() : **stm32l475e_iot01_psensor.c** , **stm32l475e_iot01_psensor.h**
- BSP_PSENSOR_ReadPressure() : **stm32l475e_iot01_psensor.c** , **stm32l475e_iot01_psensor.h**
- BSP_QSPI_DeInit() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_EnableMemoryMappedMode() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_EnterDeepPowerDown() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_Erase_Block() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_Erase_Chip() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_Erase_Sector() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_GetInfo() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_GetStatus() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_Init() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_LeaveDeepPowerDown() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_MspDeInit() : **stm32l475e_iot01_qspi.c**

- BSP_QSPI_MspInit() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_Read() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_ResumeErase() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_SuspendErase() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_Write() : **stm32l475e_iot01_qspi.c**
- BSP_TSENSOR_Init() : **stm32l475e_iot01_tsensor.c** , **stm32l475e_iot01_tsensor.h**
- BSP_TSENSOR_ReadTemp() : **stm32l475e_iot01_tsensor.c** , **stm32l475e_iot01_tsensor.h**
- BUTTON_IRQn : **stm32l475e_iot01.c**
- BUTTON_MODE_EXTI : **stm32l475e_iot01.h**
- BUTTON_MODE_GPIO : **stm32l475e_iot01.h**
- BUTTON_PIN : **stm32l475e_iot01.c**
- BUTTON_PORT : **stm32l475e_iot01.c**
- Button_TypeDef : **stm32l475e_iot01.h**
- BUTTON_USER : **stm32l475e_iot01.h**
- ButtonMode_TypeDef : **stm32l475e_iot01.h**
- BUTTONn : **stm32l475e_iot01.h**

B-L475E-IOT01 BSP User Manual

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

- c -

- COM1 : **stm32l475e_iot01.h**
- COM2 : **stm32l475e_iot01.h**
- COM_RX_AF : **stm32l475e_iot01.c**
- COM_RX_PIN : **stm32l475e_iot01.c**
- COM_RX_PORT : **stm32l475e_iot01.c**
- COM_TX_AF : **stm32l475e_iot01.c**
- COM_TX_PIN : **stm32l475e_iot01.c**
- COM_TX_PORT : **stm32l475e_iot01.c**
- COM_TypeDef : **stm32l475e_iot01.h**
- COM_USART : **stm32l475e_iot01.c**
- COMn : **stm32l475e_iot01.h**

B-L475E-IOT01 BSP User Manual

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

- d -

- DISCOVERY_COM1 : **stm32l475e_iot01.h**
- DISCOVERY_COM1_CLK_DISABLE : **stm32l475e_iot01.h**
- DISCOVERY_COM1_CLK_ENABLE : **stm32l475e_iot01.h**
- DISCOVERY_COM1_IRQn : **stm32l475e_iot01.h**
- DISCOVERY_COM1_RX_AF : **stm32l475e_iot01.h**
- DISCOVERY_COM1_RX_GPIO_CLK_DISABLE : **stm32l475e_iot01.h**
- DISCOVERY_COM1_RX_GPIO_CLK_ENABLE : **stm32l475e_iot01.h**
- DISCOVERY_COM1_RX_GPIO_PORT : **stm32l475e_iot01.h**
- DISCOVERY_COM1_RX_PIN : **stm32l475e_iot01.h**
- DISCOVERY_COM1_TX_AF : **stm32l475e_iot01.h**
- DISCOVERY_COM1_TX_GPIO_CLK_DISABLE : **stm32l475e_iot01.h**
- DISCOVERY_COM1_TX_GPIO_CLK_ENABLE : **stm32l475e_iot01.h**
- DISCOVERY_COM1_TX_GPIO_PORT : **stm32l475e_iot01.h**
- DISCOVERY_COM1_TX_PIN : **stm32l475e_iot01.h**
- DISCOVERY_COMx_CLK_DISABLE : **stm32l475e_iot01.h**
- DISCOVERY_COMx_CLK_ENABLE : **stm32l475e_iot01.h**
- DISCOVERY_COMx_RX_GPIO_CLK_DISABLE : **stm32l475e_iot01.h**

- DISCOVERY_COMx_RX_GPIO_CLK_ENABLE : **stm32l475e_iot01.h**
- DISCOVERY_COMx_TX_GPIO_CLK_DISABLE : **stm32l475e_iot01.h**
- DISCOVERY_COMx_TX_GPIO_CLK_ENABLE : **stm32l475e_iot01.h**
- DISCOVERY_DMAx_CLK_ENABLE : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_CLK_DISABLE : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_CLK_ENABLE : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_ER_IRQn : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_EV_IRQn : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_FORCE_RESET : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_RELEASE_RESET : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_SCL_PIN : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_SCL_SDA_AF : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_SCL_SDA_GPIO_CLK_DISABLE : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_SCL_SDA_GPIO_CLK_ENABLE : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_SCL_SDA_GPIO_PORT : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_SDA_PIN : **stm32l475e_iot01.h**

B-L475E-IOT01 BSP User Manual

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

- g -

- GPIO_PIN : **stm32l475e_iot01.c**
- GPIO_PORT : **stm32l475e_iot01.c**
- GYRO_ERROR : **stm32l475e_iot01_gyro.h**
- GYRO_OK : **stm32l475e_iot01_gyro.h**
- GYRO_StatusTypeDef : **stm32l475e_iot01_gyro.h**
- GYRO_TIMEOUT : **stm32l475e_iot01_gyro.h**
- GyroscopeDrv : **stm32l475e_iot01_gyro.c**

B-L475E-IOT01 BSP User Manual

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

- h -

- hDiscoUart : **stm32l475e_iot01.c**
- hI2cHandler : **stm32l475e_iot01.c**
- Hsensor_drv : **stm32l475e_iot01_hsensor.c**
- HSENSOR_ERROR : **stm32l475e_iot01_hsensor.h**
- HSENSOR_OK : **stm32l475e_iot01_hsensor.h**
- HSENSOR_Status_TypDef : **stm32l475e_iot01_hsensor.h**
- HTS221_I2C_ADDRESS : **stm32l475e_iot01.h**

B-L475E-IOT01 BSP User Manual

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

- i -

- I2Cx_DeInit() : **stm32l475e_iot01.c**
- I2Cx_Error() : **stm32l475e_iot01.c**
- I2Cx_Init() : **stm32l475e_iot01.c**
- I2Cx_IsDeviceReady() : **stm32l475e_iot01.c**
- I2Cx_MspDeInit() : **stm32l475e_iot01.c**
- I2Cx_MspInit() : **stm32l475e_iot01.c**
- I2Cx_ReadMultiple() : **stm32l475e_iot01.c**
- I2Cx_WriteMultiple() : **stm32l475e_iot01.c**

B-L475E-IOT01 BSP User Manual

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

- l -

- LED2 : **stm32l475e_iot01.h**
- LED2_GPIO_CLK_DISABLE : **stm32l475e_iot01.h**
- LED2_GPIO_CLK_ENABLE : **stm32l475e_iot01.h**
- LED2_GPIO_PORT : **stm32l475e_iot01.h**
- LED2_PIN : **stm32l475e_iot01.h**
- LED_GREEN : **stm32l475e_iot01.h**
- Led_TypeDef : **stm32l475e_iot01.h**
- LEDn : **stm32l475e_iot01.h**
- LEDx_GPIO_CLK_DISABLE : **stm32l475e_iot01.h**
- LEDx_GPIO_CLK_ENABLE : **stm32l475e_iot01.h**
- LPS22HB_I2C_ADDRESS : **stm32l475e_iot01.h**

B-L475E-IOT01 BSP User Manual

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

- m -

- MAGNETO_ERROR : **stm32l475e_iot01_magneto.h**
- MAGNETO_OK : **stm32l475e_iot01_magneto.h**
- MAGNETO_StatusTypeDef : **stm32l475e_iot01_magneto.h**
- MAGNETO_TIMEOUT : **stm32l475e_iot01_magneto.h**
- MagnetoDrv : **stm32l475e_iot01_magneto.c**

B-L475E-IOT01 BSP User Manual

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

- p -

- Psensor_drv : **stm32l475e_iot01_psensor.c**
- PSENSOR_ERROR : **stm32l475e_iot01_psensor.h**
- PSENSOR_OK : **stm32l475e_iot01_psensor.h**
- PSENSOR_Status_TypDef : **stm32l475e_iot01_psensor.h**

B-L475E-IOT01 BSP User Manual

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

- q -

- QSPI_AutoPollingMemReady() : **stm32l475e_iot01_qspi.c**
- QSPI_BUSY : **stm32l475e_iot01_qspi.h**
- QSPI_ERROR : **stm32l475e_iot01_qspi.h**
- QSPI_HIGH_PERF_DISABLE : **stm32l475e_iot01_qspi.c**
- QSPI_HIGH_PERF_ENABLE : **stm32l475e_iot01_qspi.c**
- QSPI_HighPerfMode() : **stm32l475e_iot01_qspi.c**
- QSPI_NOT_SUPPORTED : **stm32l475e_iot01_qspi.h**
- QSPI_OK : **stm32l475e_iot01_qspi.h**
- QSPI_QUAD_DISABLE : **stm32l475e_iot01_qspi.c**
- QSPI_QUAD_ENABLE : **stm32l475e_iot01_qspi.c**
- QSPI_QuadMode() : **stm32l475e_iot01_qspi.c**
- QSPI_ResetMemory() : **stm32l475e_iot01_qspi.c**
- QSPI_SUSPENDED : **stm32l475e_iot01_qspi.h**
- QSPI_WriteEnable() : **stm32l475e_iot01_qspi.c**
- QSPIHandle : **stm32l475e_iot01_qspi.c**

B-L475E-IOT01 BSP User Manual

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

- s -

- SENSOR_IO_DeInit() : **stm32l475e_iot01.c**
- SENSOR_IO_Delay() : **stm32l475e_iot01.c**
- SENSOR_IO_Init() : **stm32l475e_iot01.c**
- SENSOR_IO_IsDeviceReady() : **stm32l475e_iot01.c**
- SENSOR_IO_Read() : **stm32l475e_iot01.c**
- SENSOR_IO_ReadMultiple() : **stm32l475e_iot01.c**
- SENSOR_IO_Write() : **stm32l475e_iot01.c**
- SENSOR_IO_WriteMultiple() : **stm32l475e_iot01.c**

B-L475E-IOT01 BSP User Manual

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

- t -

- tsensor_drv : **stm32l475e_iot01_tsensor.c**
- TSENSOR_ERROR : **stm32l475e_iot01_tsensor.h**
- TSENSOR_I2C_ADDRESS : **stm32l475e_iot01.h**
- TSENSOR_OK : **stm32l475e_iot01_tsensor.h**
- TSENSOR_Status_TypDef : **stm32l475e_iot01_tsensor.h**

B-L475E-IOT01 BSP User Manual

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

- u -

- USER_BUTTON_EXTI_IRQn : **stm32l475e_iot01.h**
- USER_BUTTON_GPIO_CLK_DISABLE : **stm32l475e_iot01.h**
- USER_BUTTON_GPIO_CLK_ENABLE : **stm32l475e_iot01.h**
- USER_BUTTON_GPIO_PORT : **stm32l475e_iot01.h**
- USER_BUTTON_PIN : **stm32l475e_iot01.h**

B-L475E-IOT01 BSP User Manual

- b -

- BSP_ACCELERO_AccGetXYZ() : **stm32l475e_iot01_accelero.c** , **stm32l475e_iot01_accelero.h**
- BSP_ACCELERO_DeInit() : **stm32l475e_iot01_accelero.h** , **stm32l475e_iot01_accelero.c**
- BSP_ACCELERO_Init() : **stm32l475e_iot01_accelero.c** , **stm32l475e_iot01_accelero.h**
- BSP_ACCELERO_LowPower() : **stm32l475e_iot01_accelero.h** , **stm32l475e_iot01_accelero.c**
- BSP_COM_DeInit() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_COM_Init() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_GetVersion() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_GYRO_DeInit() : **stm32l475e_iot01_gyro.h** , **stm32l475e_iot01_gyro.c**
- BSP_GYRO_GetXYZ() : **stm32l475e_iot01_gyro.c** , **stm32l475e_iot01_gyro.h**
- BSP_GYRO_Init() : **stm32l475e_iot01_gyro.c** , **stm32l475e_iot01_gyro.h**
- BSP_GYRO_LowPower() : **stm32l475e_iot01_gyro.c** , **stm32l475e_iot01_gyro.h**
- BSP_HSENSOR_Init() : **stm32l475e_iot01_hsensor.c** , **stm32l475e_iot01_hsensor.h**
- BSP_HSENSOR_ReadHumidity() : **stm32l475e_iot01_hsensor.c** , **stm32l475e_iot01_hsensor.h**

- BSP_HSENSOR_ReadID() : **stm32l475e_iot01_hsensor.c** , **stm32l475e_iot01_hsensor.h**
- BSP_LED_DeInit() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_LED_Init() : **stm32l475e_iot01.h** , **stm32l475e_iot01.c**
- BSP_LED_Off() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_LED_On() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_LED_Toggle() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_MAGNETO_DeInit() : **stm32l475e_iot01_magneto.c** , **stm32l475e_iot01_magneto.h**
- BSP_MAGNETO_GetXYZ() : **stm32l475e_iot01_magneto.c** , **stm32l475e_iot01_magneto.h**
- BSP_MAGNETO_Init() : **stm32l475e_iot01_magneto.c** , **stm32l475e_iot01_magneto.h**
- BSP_MAGNETO_LowPower() : **stm32l475e_iot01_magneto.c** , **stm32l475e_iot01_magneto.h**
- BSP_PB_DeInit() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_PB_GetState() : **stm32l475e_iot01.c** , **stm32l475e_iot01.h**
- BSP_PB_Init() : **stm32l475e_iot01.h** , **stm32l475e_iot01.c**
- BSP_PSENSOR_Init() : **stm32l475e_iot01_psensor.c** , **stm32l475e_iot01_psensor.h**
- BSP_PSENSOR_ReadID() : **stm32l475e_iot01_psensor.c** , **stm32l475e_iot01_psensor.h**
- BSP_PSENSOR_ReadPressure() : **stm32l475e_iot01_psensor.c** , **stm32l475e_iot01_psensor.h**
- BSP_QSPI_DeInit() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_EnableMemoryMappedMode() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_EnterDeepPowerDown() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_Erase_Block() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_Erase_Chip() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_Erase_Sector() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_GetInfo() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_GetStatus() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_Init() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_LeaveDeepPowerDown() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_MspDeInit() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_MspInit() : **stm32l475e_iot01_qspi.c**

- BSP_QSPI_Read() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_ResumeErase() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_SuspendErase() : **stm32l475e_iot01_qspi.c**
- BSP_QSPI_Write() : **stm32l475e_iot01_qspi.c**
- BSP_TSENSOR_Init() : **stm32l475e_iot01_tsensor.h** , **stm32l475e_iot01_tsensor.c**
- BSP_TSENSOR_ReadTemp() : **stm32l475e_iot01_tsensor.h** , **stm32l475e_iot01_tsensor.c**

- i -

- I2Cx_DeInit() : **stm32l475e_iot01.c**
- I2Cx_Error() : **stm32l475e_iot01.c**
- I2Cx_Init() : **stm32l475e_iot01.c**
- I2Cx_IsDeviceReady() : **stm32l475e_iot01.c**
- I2Cx_MspDeInit() : **stm32l475e_iot01.c**
- I2Cx_MspInit() : **stm32l475e_iot01.c**
- I2Cx_ReadMultiple() : **stm32l475e_iot01.c**
- I2Cx_WriteMultiple() : **stm32l475e_iot01.c**

- q -

- QSPI_AutoPollingMemReady() : **stm32l475e_iot01_qspi.c**
- QSPI_HighPerfMode() : **stm32l475e_iot01_qspi.c**
- QSPI_QuadMode() : **stm32l475e_iot01_qspi.c**
- QSPI_ResetMemory() : **stm32l475e_iot01_qspi.c**
- QSPI_WriteEnable() : **stm32l475e_iot01_qspi.c**

- s -

- SENSOR_IO_DeInit() : **stm32l475e_iot01.c**
- SENSOR_IO_Delay() : **stm32l475e_iot01.c**
- SENSOR_IO_Init() : **stm32l475e_iot01.c**
- SENSOR_IO_IsDeviceReady() : **stm32l475e_iot01.c**
- SENSOR_IO_Read() : **stm32l475e_iot01.c**
- SENSOR_IO_ReadMultiple() : **stm32l475e_iot01.c**
- SENSOR_IO_Write() : **stm32l475e_iot01.c**
- SENSOR_IO_WriteMultiple() : **stm32l475e_iot01.c**

Generated on Thu Mar 16 2017 10:38:33 for B-L475E-IOT01 BSP User Manual by doxygen 1.7.6.1

B-L475E-IOT01 BSP User Manual

- AccelerometerDrv : **stm32l475e_iot01_accelero.c**
- BUTTON_IRQn : **stm32l475e_iot01.c**
- BUTTON_PIN : **stm32l475e_iot01.c**
- BUTTON_PORT : **stm32l475e_iot01.c**
- COM_RX_AF : **stm32l475e_iot01.c**
- COM_RX_PIN : **stm32l475e_iot01.c**
- COM_RX_PORT : **stm32l475e_iot01.c**
- COM_TX_AF : **stm32l475e_iot01.c**
- COM_TX_PIN : **stm32l475e_iot01.c**
- COM_TX_PORT : **stm32l475e_iot01.c**
- COM_USART : **stm32l475e_iot01.c**
- GPIO_PIN : **stm32l475e_iot01.c**
- GPIO_PORT : **stm32l475e_iot01.c**
- GyroscopeDrv : **stm32l475e_iot01_gyro.c**
- hDiscoUart : **stm32l475e_iot01.c**
- hI2cHandler : **stm32l475e_iot01.c**
- Hsensor_drv : **stm32l475e_iot01_hsensor.c**
- MagnetoDrv : **stm32l475e_iot01_magneto.c**
- Psensor_drv : **stm32l475e_iot01_psensor.c**
- QSPIHandle : **stm32l475e_iot01_qspi.c**
- tsensor_drv : **stm32l475e_iot01_tsensor.c**

B-L475E-IOT01 BSP User Manual

- ACCELERO_StatusTypeDef : **stm32l475e_iot01_accelero.h**
- Button_TypeDef : **stm32l475e_iot01.h**
- ButtonMode_TypeDef : **stm32l475e_iot01.h**
- COM_TypeDef : **stm32l475e_iot01.h**
- GYRO_StatusTypeDef : **stm32l475e_iot01_gyro.h**
- HSENSOR_Status_TypDef : **stm32l475e_iot01_hsensor.h**
- Led_TypeDef : **stm32l475e_iot01.h**
- MAGNETO_StatusTypeDef : **stm32l475e_iot01_magneto.h**
- PSENSOR_Status_TypDef : **stm32l475e_iot01_psensor.h**
- TSENSOR_Status_TypDef : **stm32l475e_iot01_tsensor.h**

B-L475E-IOT01 BSP User Manual

- ACCELERO_ERROR : **stm32l475e_iot01_accelero.h**
- ACCELERO_OK : **stm32l475e_iot01_accelero.h**
- ACCELERO_TIMEOUT : **stm32l475e_iot01_accelero.h**
- BUTTON_MODE_EXTI : **stm32l475e_iot01.h**
- BUTTON_MODE_GPIO : **stm32l475e_iot01.h**
- BUTTON_USER : **stm32l475e_iot01.h**
- COM1 : **stm32l475e_iot01.h**
- COM2 : **stm32l475e_iot01.h**
- GYRO_ERROR : **stm32l475e_iot01_gyro.h**
- GYRO_OK : **stm32l475e_iot01_gyro.h**
- GYRO_TIMEOUT : **stm32l475e_iot01_gyro.h**
- HSENSOR_ERROR : **stm32l475e_iot01_hsensor.h**
- HSENSOR_OK : **stm32l475e_iot01_hsensor.h**
- LED2 : **stm32l475e_iot01.h**
- LED_GREEN : **stm32l475e_iot01.h**
- MAGNETO_ERROR : **stm32l475e_iot01_magneto.h**
- MAGNETO_OK : **stm32l475e_iot01_magneto.h**
- MAGNETO_TIMEOUT : **stm32l475e_iot01_magneto.h**
- PSENSOR_ERROR : **stm32l475e_iot01_psensor.h**
- PSENSOR_OK : **stm32l475e_iot01_psensor.h**
- TSENSOR_ERROR : **stm32l475e_iot01_tsensor.h**
- TSENSOR_OK : **stm32l475e_iot01_tsensor.h**

B-L475E-IOT01 BSP User Manual

- _ -

- __STM32L475E_IOT01_BSP_VERSION : **stm32l475e_iot01.c**
- __STM32L475E_IOT01_BSP_VERSION_MAIN : **stm32l475e_iot01.c**
- __STM32L475E_IOT01_BSP_VERSION_RC : **stm32l475e_iot01.c**
- __STM32L475E_IOT01_BSP_VERSION_SUB1 : **stm32l475e_iot01.c**
- __STM32L475E_IOT01_BSP_VERSION_SUB2 : **stm32l475e_iot01.c**

- b -

- BUTTONn : **stm32l475e_iot01.h**

- c -

- COMn : **stm32l475e_iot01.h**

- d -

- DISCOVERY_COM1 : **stm32l475e_iot01.h**
- DISCOVERY_COM1_CLK_DISABLE : **stm32l475e_iot01.h**
- DISCOVERY_COM1_CLK_ENABLE : **stm32l475e_iot01.h**
- DISCOVERY_COM1_IRQn : **stm32l475e_iot01.h**

- DISCOVERY_COM1_RX_AF : **stm32l475e_iot01.h**
- DISCOVERY_COM1_RX_GPIO_CLK_DISABLE : **stm32l475e_iot01.h**
- DISCOVERY_COM1_RX_GPIO_CLK_ENABLE : **stm32l475e_iot01.h**
- DISCOVERY_COM1_RX_GPIO_PORT : **stm32l475e_iot01.h**
- DISCOVERY_COM1_RX_PIN : **stm32l475e_iot01.h**
- DISCOVERY_COM1_TX_AF : **stm32l475e_iot01.h**
- DISCOVERY_COM1_TX_GPIO_CLK_DISABLE : **stm32l475e_iot01.h**
- DISCOVERY_COM1_TX_GPIO_CLK_ENABLE : **stm32l475e_iot01.h**
- DISCOVERY_COM1_TX_GPIO_PORT : **stm32l475e_iot01.h**
- DISCOVERY_COM1_TX_PIN : **stm32l475e_iot01.h**
- DISCOVERY_COMx_CLK_DISABLE : **stm32l475e_iot01.h**
- DISCOVERY_COMx_CLK_ENABLE : **stm32l475e_iot01.h**
- DISCOVERY_COMx_RX_GPIO_CLK_DISABLE : **stm32l475e_iot01.h**
- DISCOVERY_COMx_RX_GPIO_CLK_ENABLE : **stm32l475e_iot01.h**
- DISCOVERY_COMx_TX_GPIO_CLK_DISABLE : **stm32l475e_iot01.h**
- DISCOVERY_COMx_TX_GPIO_CLK_ENABLE : **stm32l475e_iot01.h**
- DISCOVERY_DMAx_CLK_ENABLE : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_CLK_DISABLE : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_CLK_ENABLE : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_ER_IRQn : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_EV_IRQn : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_FORCE_RESET : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_RELEASE_RESET : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_SCL_PIN : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_SCL_SDA_AF : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_SCL_SDA_GPIO_CLK_DISABLE : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_SCL_SDA_GPIO_CLK_ENABLE :

stm32l475e_iot01.h
- DISCOVERY_I2Cx_SCL_SDA_GPIO_PORT : **stm32l475e_iot01.h**
- DISCOVERY_I2Cx_SDA_PIN : **stm32l475e_iot01.h**

- h -

- HTS221_I2C_ADDRESS : **stm32l475e_iot01.h**

- l -

- LED2_GPIO_CLK_DISABLE : **stm32l475e_iot01.h**
- LED2_GPIO_CLK_ENABLE : **stm32l475e_iot01.h**
- LED2_GPIO_PORT : **stm32l475e_iot01.h**
- LED2_PIN : **stm32l475e_iot01.h**
- LEDn : **stm32l475e_iot01.h**
- LEDx_GPIO_CLK_DISABLE : **stm32l475e_iot01.h**
- LEDx_GPIO_CLK_ENABLE : **stm32l475e_iot01.h**
- LPS22HB_I2C_ADDRESS : **stm32l475e_iot01.h**

- q -

- QSPI_BUSY : **stm32l475e_iot01_qspi.h**
- QSPI_ERROR : **stm32l475e_iot01_qspi.h**
- QSPI_HIGH_PERF_DISABLE : **stm32l475e_iot01_qspi.c**
- QSPI_HIGH_PERF_ENABLE : **stm32l475e_iot01_qspi.c**
- QSPI_NOT_SUPPORTED : **stm32l475e_iot01_qspi.h**
- QSPI_OK : **stm32l475e_iot01_qspi.h**
- QSPI_QUAD_DISABLE : **stm32l475e_iot01_qspi.c**
- QSPI_QUAD_ENABLE : **stm32l475e_iot01_qspi.c**
- QSPI_SUSPENDED : **stm32l475e_iot01_qspi.h**

- t -

- TSENSOR_I2C_ADDRESS : **stm32l475e_iot01.h**

- u -

- USER_BUTTON_EXTI_IRQn : **stm32l475e_iot01.h**

- USER_BUTTON_GPIO_CLK_DISABLE : **stm32l475e_iot01.h**
- USER_BUTTON_GPIO_CLK_ENABLE : **stm32l475e_iot01.h**
- USER_BUTTON_GPIO_PORT : **stm32l475e_iot01.h**
- USER_BUTTON_PIN : **stm32l475e_iot01.h**

B-L475E-IOT01 BSP User Manual

Defines | Functions | Variables

stm32l475e_iot01.c File Reference

STM32L475E-IOT01 board support package. More...

```
#include "stm32l475e_iot01.h"
```

Go to the source code of this file.

Defines

#define	**__STM32L475E_IOT01_BSP_VERSION_MAIN** (0x01) STM32L475E IOT01 BSP Driver version number V1.0.0.
#define	**__STM32L475E_IOT01_BSP_VERSION_SUB1** (0x00)
#define	**__STM32L475E_IOT01_BSP_VERSION_SUB2** (0x00)
#define	**__STM32L475E_IOT01_BSP_VERSION_RC** (0x00)
#define	**__STM32L475E_IOT01_BSP_VERSION**

Functions

static void	**I2Cx_MspInit** (I2C_HandleTypeDef *i2c_handler) Initializes I2C MSP.
static void	**I2Cx_MspDeInit** (I2C_HandleTypeDef *i2c_handler) DeInitializes I2C MSP.
static void	**I2Cx_Init** (I2C_HandleTypeDef *i2c_handler) Initializes I2C HAL.
static void	**I2Cx_DeInit** (I2C_HandleTypeDef *i2c_handler) DeInitializes I2C HAL.
static HAL_StatusTypeDef	**I2Cx_ReadMultiple** (I2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t *Buffer, uint16_t Length) Reads multiple data.
static HAL_StatusTypeDef	**I2Cx_WriteMultiple** (I2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t *Buffer, uint16_t Length) Writes a value in a register of the device through BUS in using DMA mode.
static HAL_StatusTypeDef	**I2Cx_IsDeviceReady** (I2C_HandleTypeDef *i2c_handler, uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.
static void	**I2Cx_Error** (I2C_HandleTypeDef *i2c_handler, uint8_t Addr) Manages error callback by re-initializing I2C.

void	**SENSOR_IO_Init** (void) Initializes Sensors low level.
void	**SENSOR_IO_DeInit** (void) DeInitializes Sensors low level.
void	**SENSOR_IO_Write** (uint8_t Addr, uint8_t Reg, uint8_t Value) Writes a single data.
uint8_t	**SENSOR_IO_Read** (uint8_t Addr, uint8_t Reg) Reads a single data.
uint16_t	**SENSOR_IO_ReadMultiple** (uint8_t Addr, uint8_t Reg, uint8_t *Buffer, uint16_t Length) Reads multiple data with I2C communication channel from TouchScreen.
void	**SENSOR_IO_WriteMultiple** (uint8_t Addr, uint8_t Reg, uint8_t *Buffer, uint16_t Length) Writes multiple data with I2C communication channel from MCU to TouchScreen.
HAL_StatusTypeDef	**SENSOR_IO_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.
void	**SENSOR_IO_Delay** (uint32_t Delay) Delay function used in TouchScreen low level driver.
uint32_t	**BSP_GetVersion** (void) This method returns the STM32L475E IOT01 BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LEDs.

void	**BSP_LED_DeInit** (**Led_TypeDef** Led) DeInit LEDs.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** ButtonMode) Configures button GPIO and EXTI Line.
void	**BSP_PB_DeInit** (**Button_TypeDef** Button) Push Button DeInit.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) Configures COM port.
void	**BSP_COM_DeInit** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) DeInit COM port.

Variables

const uint32_t	**GPIO_PIN** [**LEDn**] = {**LED2_PIN**}
GPIO_TypeDef *	**GPIO_PORT** [**LEDn**] = {**LED2_GPIO_PORT**}
GPIO_TypeDef *	**BUTTON_PORT** [**BUTTONn**] = {**USER_BUTTON_GPIO_PORT**}
const uint16_t	**BUTTON_PIN** [**BUTTONn**] = {**USER_BUTTON_PIN**}
const uint16_t	**BUTTON_IRQn** [**BUTTONn**] = {**USER_BUTTON_EXTI_IRQn**}
USART_TypeDef *	**COM_USART** [**COMn**] = {**DISCOVERY_COM1**}
GPIO_TypeDef *	**COM_TX_PORT** [**COMn**] = {**DISCOVERY_COM1_TX_GPIO_PORT**}
GPIO_TypeDef *	**COM_RX_PORT** [**COMn**] = {**DISCOVERY_COM1_RX_GPIO_PORT**}
const uint16_t	**COM_TX_PIN** [**COMn**] = {**DISCOVERY_COM1_TX_PIN**}
const uint16_t	**COM_RX_PIN** [**COMn**] = {**DISCOVERY_COM1_RX_PIN**}
const uint16_t	**COM_TX_AF** [**COMn**] = {**DISCOVERY_COM1_TX_AF**}
const uint16_t	**COM_RX_AF** [**COMn**] = {**DISCOVERY_COM1_RX_AF**}
I2C_HandleTypeDef	**hI2cHandler**
UART_HandleTypeDef	**hDiscoUart**

Detailed Description

STM32L475E-IOT01 board support package.

Author:
MCD Application Team

Version:
V1.0.0

Date:
17-March-2017

Attention:

OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file **stm32l475e_iot01.c**.

B-L475E-IOT01 BSP User Manual

Defines | Enumerations | Functions

stm32l475e_iot01.h File Reference

STM32L475E IOT01 board support package. More...

```
#include "stm32l4xx_hal.h"
```

Go to the source code of this file.

Defines

#define **LEDn** ((uint8_t)1)
Define for STM32L475E_IOT01 board.

#define **LED2_PIN** GPIO_PIN_14

#define **LED2_GPIO_PORT** GPIOB

#define **LED2_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_CLK_E

#define **LED2_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_CLK_D

#define **LEDx_GPIO_CLK_ENABLE**(__INDEX__) do{if((__INDEX__)
LED2_GPIO_CLK_ENABLE();}while(0)

#define **LEDx_GPIO_CLK_DISABLE**(__INDEX__) do{if((__INDEX__
LED2_GPIO_CLK_DISABLE();}while(0)

#define **BUTTONn** ((uint8_t)1)

#define **USER_BUTTON_PIN** GPIO_PIN_13
Wakeup push-button.

#define **USER_BUTTON_GPIO_PORT** GPIOC

#define **USER_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_GPIC

#define **USER_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_GPI

#define **USER_BUTTON_EXTI_IRQn** EXTI15_10_IRQn

#define **COMn** ((uint8_t)1)

#define **DISCOVERY_COM1** USART1
Definition for COM port1, connected to USART1.

#define **DISCOVERY_COM1_CLK_ENABLE**() __HAL_RCC_USART

#define **DISCOVERY_COM1_CLK_DISABLE**() __HAL_RCC_USART

#define **DISCOVERY_COM1_TX_PIN** GPIO_PIN_6

#define **DISCOVERY_COM1_TX_GPIO_PORT** GPIOB

#define **DISCOVERY_COM1_TX_GPIO_CLK_ENABLE**() __HAL_RC

#define **DISCOVERY_COM1_TX_GPIO_CLK_DISABLE**() __HAL_R

#define **DISCOVERY_COM1_TX_AF** GPIO_AF7_USART1

#define **DISCOVERY_COM1_RX_PIN** GPIO_PIN_7

#define **DISCOVERY_COM1_RX_GPIO_PORT** GPIOB

#define **DISCOVERY_COM1_RX_GPIO_CLK_ENABLE**() __HAL_RC

#define **DISCOVERY_COM1_RX_GPIO_CLK_DISABLE**() __HAL_R

```
#define DISCOVERY_COM1_RX_AF   GPIO_AF7_USART1
#define DISCOVERY_COM1_IRQn   USART1_IRQn
#define DISCOVERY_COMx_CLK_ENABLE(__INDEX__)   do { if((__
                {DISCOVERY_COM1_CLK_ENABLE();}} while(0)
#define DISCOVERY_COMx_CLK_DISABLE(__INDEX__)   do { if((__
                {DISCOVERY_COM1_CLK_DISABLE();}} while(0)
#define DISCOVERY_COMx_TX_GPIO_CLK_ENABLE(__INDEX__)
                {DISCOVERY_COM1_TX_GPIO_CLK_ENABLE();}} while(0)
#define DISCOVERY_COMx_TX_GPIO_CLK_DISABLE(__INDEX__)
                {DISCOVERY_COM1_TX_GPIO_CLK_DISABLE();}} while(0)
#define DISCOVERY_COMx_RX_GPIO_CLK_ENABLE(__INDEX__)
                {DISCOVERY_COM1_RX_GPIO_CLK_ENABLE();}} while(0)
#define DISCOVERY_COMx_RX_GPIO_CLK_DISABLE(__INDEX__)
                {DISCOVERY_COM1_RX_GPIO_CLK_DISABLE();}} while(0)
#define DISCOVERY_I2Cx   I2C2
#define DISCOVERY_I2Cx_CLK_ENABLE()   __HAL_RCC_I2C2_CLI
#define DISCOVERY_I2Cx_CLK_DISABLE()   __HAL_RCC_I2C2_CL
#define DISCOVERY_DMAx_CLK_ENABLE()   __HAL_RCC_DMA1_
#define DISCOVERY_I2Cx_SCL_SDA_GPIO_CLK_ENABLE()   __HA
#define DISCOVERY_I2Cx_SCL_SDA_GPIO_CLK_DISABLE()   __H
#define DISCOVERY_I2Cx_FORCE_RESET()   __HAL_RCC_I2C2_F
#define DISCOVERY_I2Cx_RELEASE_RESET()   __HAL_RCC_I2C2_
#define DISCOVERY_I2Cx_SCL_PIN   GPIO_PIN_10
#define DISCOVERY_I2Cx_SDA_PIN   GPIO_PIN_11
#define DISCOVERY_I2Cx_SCL_SDA_GPIO_PORT   GPIOB
#define DISCOVERY_I2Cx_SCL_SDA_AF   GPIO_AF4_I2C2
#define DISCOVERY_I2Cx_EV_IRQn   I2C2_EV_IRQn
#define DISCOVERY_I2Cx_ER_IRQn   I2C2_ER_IRQn
#define LPS22HB_I2C_ADDRESS   (uint8_t)0xBA
#define HTS221_I2C_ADDRESS   (uint8_t)0xBE
#define TSENSOR_I2C_ADDRESS   HTS221_I2C_ADDRESS
```

Enumerations

enum	**Led_TypeDef** { **LED2** = 0, **LED_GREEN** = LED2 }
enum	**Button_TypeDef** { **BUTTON_USER** = 0 }
enum	**ButtonMode_TypeDef** { **BUTTON_MODE_GPIO** = 0, **BUTTON_MODE_EXTI** = 1 }
enum	**COM_TypeDef** { **COM1** = 0, **COM2** = 0 }

Functions

uint32_t	**BSP_GetVersion** (void) This method returns the STM32L475E IOT01 BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LEDs.
void	**BSP_LED_DeInit** (**Led_TypeDef** Led) DeInit LEDs.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** ButtonMode) Configures button GPIO and EXTI Line.
void	**BSP_PB_DeInit** (**Button_TypeDef** Button) Push Button DeInit.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *husart) Configures COM port.
void	**BSP_COM_DeInit** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) DeInit COM port.

Detailed Description

STM32L475E IOT01 board support package.

Author:
MCD Application Team

Version:
V1.0.0

Date:
17-March-2017

Attention:

OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file **stm32l475e_iot01.h**.

---

stm32l475e_iot01_accelero.c File Reference

This file provides a set of functions needed to manage the accelerometer sensor. More...

```
#include "stm32l475e_iot01_accelero.h"
```

Go to the source code of this file.

Functions

ACCELERO_StatusTypeDef	**BSP_ACCELERO_Init** (void) Initialize the ACCELERO.
void	**BSP_ACCELERO_DeInit** (void) DeInitialize the ACCELERO.
void	**BSP_ACCELERO_LowPower** (uint16_t status) Set/Unset the ACCELERO in low power mode.
void	**BSP_ACCELERO_AccGetXYZ** (int16_t *pDataXYZ) Get XYZ acceleration values.

Variables

static ACCELERO_DrvTypeDef *	**AccelerometerDrv**

Detailed Description

This file provides a set of functions needed to manage the accelerometer sensor.

Author:
MCD Application Team

Version:
V1.0.0

Date:
17-March-2017

Attention:

OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file **stm32l475e_iot01_accelero.c**.

B-L475E-IOT01 BSP User Manual

Enumerations | Functions

stm32l475e_iot01_accelero.h File Reference

This file provides a set of functions needed to manage the accelerometer sensor. More...

```
#include "stm32l475e_iot01.h" #include
"../Components/lsm6dsl/lsm6dsl.h"
```

Go to the source code of this file.

Enumerations

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

Functions

ACCELERO_StatusTypeDef	**BSP_ACCELERO_Init** (void) Initialize the ACCELERO.
void	**BSP_ACCELERO_DeInit** (void) DeInitialize the ACCELERO.
void	**BSP_ACCELERO_LowPower** (uint16_t status) Set/Unset the ACCELERO in low power mode.
void	**BSP_ACCELERO_AccGetXYZ** (int16_t *pDataXYZ) Get XYZ acceleration values.

Detailed Description

This file provides a set of functions needed to manage the accelerometer sensor.

Author:
MCD Application Team

Version:
V1.0.0

Date:
17-March-2017

Attention:

OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file **stm32l475e_iot01_accelero.h**.

---

stm32l475e_iot01_gyro.c File Reference

This file provides a set of functions needed to manage the gyroscope sensor. More...

```
#include "stm32l475e_iot01_gyro.h"
```

Go to the source code of this file.

Functions

uint8_t	**BSP_GYRO_Init** (void) Initialize Gyroscope.
void	**BSP_GYRO_DeInit** (void) DeInitialize Gyroscope.
void	**BSP_GYRO_LowPower** (uint16_t status) Set/Unset Gyroscope in low power mode.
void	**BSP_GYRO_GetXYZ** (float *pfData) Get XYZ angular acceleration from the Gyroscope.

Variables

static GYRO_DrvTypeDef * **GyroscopeDrv**

Detailed Description

This file provides a set of functions needed to manage the gyroscope sensor.

Author:
MCD Application Team

Version:
V1.0.0

Date:
17-March-2017

Attention:

OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file **stm32l475e_iot01_gyro.c**.

B-L475E-IOT01 BSP User Manual

Enumerations | Functions

stm32l475e_iot01_gyro.h File Reference

This file contains definitions for the **stm32l475e_iot01_gyro.c**. More...

```
#include "stm32l475e_iot01.h" #include
"../Components/lsm6dsl/lsm6dsl.h"
```

Go to the source code of this file.

Enumerations

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

Functions

uint8_t	**BSP_GYRO_Init** (void) Initialize Gyroscope.
void	**BSP_GYRO_DeInit** (void) DeInitialize Gyroscope.
void	**BSP_GYRO_LowPower** (uint16_t status) Set/Unset Gyroscope in low power mode.
void	**BSP_GYRO_GetXYZ** (float *pfData) Get XYZ angular acceleration from the Gyroscope.

Detailed Description

This file contains definitions for the **stm32l475e_iot01_gyro.c**.

Author:
MCD Application Team

Version:
V1.0.0

Date:
17-March-2017

Attention:

OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file **stm32l475e_iot01_gyro.h**.

---

stm32l475e_iot01_hsensor.c File Reference

This file provides a set of functions needed to manage the humidity sensor. More...

```
#include "stm32l475e_iot01_hsensor.h"
```

Go to the source code of this file.

Functions

uint32_t	**BSP_HSENSOR_Init** (void) Initializes peripherals used by the I2C Humidity Sensor driver.
uint8_t	**BSP_HSENSOR_ReadID** (void) Read ID of HTS221.
float	**BSP_HSENSOR_ReadHumidity** (void) Read Humidity register of HTS221.

Variables

static HSENSOR_DrvTypeDef *	**Hsensor_drv**

Detailed Description

This file provides a set of functions needed to manage the humidity sensor.

Author:
MCD Application Team

Version:
V1.0.0

Date:
17-March-2017

Attention:

OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file **stm32l475e_iot01_hsensor.c**.

B-L475E-IOT01 BSP User Manual

Enumerations | Functions

stm32l475e_iot01_hsensor.h File Reference

This file provides a set of functions needed to manage the humidity sensor. More...

```
#include "stm32l475e_iot01.h" #include
"../Components/hts221/hts221.h"
```

Go to the source code of this file.

Enumerations

enum	**HSENSOR_Status_TypDef** { **HSENSOR_OK** = 0, **HSENSOR_ERROR** } HSENSOR Status. More...

Functions

uint32_t	**BSP_HSENSOR_Init** (void) Initializes peripherals used by the I2C Humidity Sensor driver.
uint8_t	**BSP_HSENSOR_ReadID** (void) Read ID of HTS221.
float	**BSP_HSENSOR_ReadHumidity** (void) Read Humidity register of HTS221.

Detailed Description

This file provides a set of functions needed to manage the humidity sensor.

Author:
MCD Application Team

Version:
V1.0.0

Date:
17-March-2017

Attention:

OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file **stm32l475e_iot01_hsensor.h**.

B-L475E-IOT01 BSP User Manual

stm32l475e_iot01_magneto.c File Reference

This file provides a set of functions needed to manage the magnetometer sensor. More...

```
#include "stm32l475e_iot01_magneto.h"
```

Go to the source code of this file.

Functions

MAGNETO_StatusTypeDef	**BSP_MAGNETO_Init** (void) Initialize a magnetometer sensor.
void	**BSP_MAGNETO_DeInit** (void) DeInitialize the MAGNETO.
void	**BSP_MAGNETO_LowPower** (uint16_t status) Set/Unset the MAGNETO in low power mode.
void	**BSP_MAGNETO_GetXYZ** (int16_t *pDataXYZ) Get XYZ magnetometer values.

Variables

static MAGNETO_DrvTypeDef *	**MagnetoDrv**

Detailed Description

This file provides a set of functions needed to manage the magnetometer sensor.

Author:
MCD Application Team

Version:
V1.0.0

Date:
17-March-2017

Attention:

OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file **stm32l475e_iot01_magneto.c**.

---

stm32l475e_iot01_magneto.h File Reference

This file provides a set of functions needed to manage the magnetometer sensor. More...

```
#include "stm32l475e_iot01.h" #include
"../Components/lis3mdl/lis3mdl.h"
```

Go to the source code of this file.

Enumerations

enum	**MAGNETO_StatusTypeDef** { **MAGNETO_OK** = 0, **MAGNETO_ERROR** = 1, **MAGNETO_TIMEOUT** = 2 }

Functions

MAGNETO_StatusTypeDef	**BSP_MAGNETO_Init** (void) Initialize a magnetometer sensor.
void	**BSP_MAGNETO_DeInit** (void) DeInitialize the MAGNETO.
void	**BSP_MAGNETO_LowPower** (uint16_t status) Set/Unset the MAGNETO in low power mode.
void	**BSP_MAGNETO_GetXYZ** (int16_t *pDataXYZ) Get XYZ magnetometer values.

Detailed Description

This file provides a set of functions needed to manage the magnetometer sensor.

Author:
MCD Application Team

Version:
V1.0.0

Date:
17-March-2017

Attention:

OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file **stm32l475e_iot01_magneto.h**.

B-L475E-IOT01 BSP User Manual

stm32l475e_iot01_psensor.c File Reference

This file provides a set of functions needed to manage the pressure sensor. More...

```
#include "stm32l475e_iot01_psensor.h"
```

Go to the source code of this file.

Functions

uint32_t	**BSP_PSENSOR_Init** (void) Initializes peripherals used by the I2C Pressure Sensor driver.
uint8_t	**BSP_PSENSOR_ReadID** (void) Read ID of LPS22HB.
float	**BSP_PSENSOR_ReadPressure** (void) Read Pressure register of LPS22HB.

Variables

static PSENSOR_DrvTypeDef * **Psensor_drv**

Detailed Description

This file provides a set of functions needed to manage the pressure sensor.

Author:
MCD Application Team

Version:
V1.0.0

Date:
17-March-2017

Attention:

OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file **stm32l475e_iot01_psensor.c**.

B-L475E-IOT01 BSP User Manual

Enumerations | Functions

stm32l475e_iot01_psensor.h File Reference

This file provides a set of functions needed to manage the pressure sensor. More...

```
#include "stm32l475e_iot01.h" #include
"../Components/lps22hb/lps22hb.h"
```

Go to the source code of this file.

Enumerations

enum	**PSENSOR_Status_TypDef** { **PSENSOR_OK** = 0, **PSENSOR_ERROR** } PSENSOR Status. More...

Functions

uint32_t	**BSP_PSENSOR_Init** (void) Initializes peripherals used by the I2C Pressure Sensor driver.
uint8_t	**BSP_PSENSOR_ReadID** (void) Read ID of LPS22HB.
float	**BSP_PSENSOR_ReadPressure** (void) Read Pressure register of LPS22HB.

Detailed Description

This file provides a set of functions needed to manage the pressure sensor.

Author:
MCD Application Team

Version:
V1.0.0

Date:
17-March-2017

Attention:

OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file **stm32l475e_iot01_psensor.h**.

B-L475E-IOT01 BSP User Manual

stm32l475e_iot01_qspi.c File Reference

This file includes a standard driver for the MX25R6435F QSPI memory mounted on STM32L475E IOT01 board. More...

```
#include "stm32l475e_iot01_qspi.h"
```

Go to the source code of this file.

Defines

#define	**QSPI_QUAD_DISABLE** 0x0
#define	**QSPI_QUAD_ENABLE** 0x1
#define	**QSPI_HIGH_PERF_DISABLE** 0x0
#define	**QSPI_HIGH_PERF_ENABLE** 0x1

Functions

static uint8_t	**QSPI_ResetMemory** (QSPI_HandleTypeDef *hqspi) This function reset the QSPI memory.
static uint8_t	**QSPI_WriteEnable** (QSPI_HandleTypeDef *hqspi) This function send a Write Enable and wait it is effective.
static uint8_t	**QSPI_AutoPollingMemReady** (QSPI_HandleTypeDef *hqspi, uint32_t Timeout) This function read the SR of the memory and wait the EOP.
static uint8_t	**QSPI_QuadMode** (QSPI_HandleTypeDef *hqspi, uint8_t Operation) This function enables/disables the Quad mode of the memory.
static uint8_t	**QSPI_HighPerfMode** (QSPI_HandleTypeDef *hqspi, uint8_t Operation) This function enables/disables the high performance mode of the memory.
uint8_t	**BSP_QSPI_Init** (void) Initializes the QSPI interface.
uint8_t	**BSP_QSPI_DeInit** (void) De-Initializes the QSPI interface.
uint8_t	**BSP_QSPI_Read** (uint8_t *pData, uint32_t ReadAddr, uint32_t Size) Reads an amount of data from the QSPI memory.
uint8_t	**BSP_QSPI_Write** (uint8_t *pData, uint32_t WriteAddr, uint32_t Size) Writes an amount of data to the QSPI memory.
uint8_t	**BSP_QSPI_Erase_Block** (uint32_t BlockAddress) Erases the specified block of the QSPI memory.
uint8_t	**BSP_QSPI_Erase_Sector** (uint32_t Sector) Erases the specified sector of the QSPI memory.

uint8_t	**BSP_QSPI_Erase_Chip** (void) Erases the entire QSPI memory.
uint8_t	**BSP_QSPI_GetStatus** (void) Reads current status of the QSPI memory.
uint8_t	**BSP_QSPI_GetInfo** (**QSPI_Info** *pInfo) Return the configuration of the QSPI memory.
uint8_t	**BSP_QSPI_EnableMemoryMappedMode** (void) Configure the QSPI in memory-mapped mode.
uint8_t	**BSP_QSPI_SuspendErase** (void) This function suspends an ongoing erase command.
uint8_t	**BSP_QSPI_ResumeErase** (void) This function resumes a paused erase command.
uint8_t	**BSP_QSPI_EnterDeepPowerDown** (void) This function enter the QSPI memory in deep power down mode.
uint8_t	**BSP_QSPI_LeaveDeepPowerDown** (void) This function leave the QSPI memory from deep power down mode.
__weak void	**BSP_QSPI_MspInit** (void) Initializes the QSPI MSP.
__weak void	**BSP_QSPI_MspDeInit** (void) De-Initializes the QSPI MSP.

Variables

QSPI_HandleTypeDef	**QSPIHandle**

Detailed Description

This file includes a standard driver for the MX25R6435F QSPI memory mounted on STM32L475E IOT01 board.

Author:
MCD Application Team

Version:
V1.0.0

Date:
17-March-2017

```
  ==========================================
==================================
                     ##### How to use this dri
ver #####
  ==========================================
==================================
  [..]
   (#) This driver is used to drive the MX25R6
435F QSPI external
       memory mounted on STM32L475E IOT01 boar
d.

   (#) This driver need a specific component d
river (MX25R6435F) to be included with.

   (#) Initialization steps:
       (++) Initialize the QPSI external memor
y using the BSP_QSPI_Init() function. This
            function includes the MSP layer ha
rdware resources initialization and the
            QSPI interface with the external m
emory. The BSP_QSPI_DeInit() can be used
            to deactivate the QSPI interface.
```

```
   (#) QSPI memory operations
       (++) QSPI memory can be accessed with r
ead/write operations once it is
            initialized.
            Read/write operation can be perfor
med with AHB access using the functions
            BSP_QSPI_Read()/BSP_QSPI_Write().
       (++) The function to the QSPI memory in
 memory-mapped mode is possible after
            the call of the function BSP_QSPI_
EnableMemoryMappedMode().
       (++) The function BSP_QSPI_GetInfo() re
turns the configuration of the QSPI memory.
            (see the QSPI memory data sheet)
       (++) Perform erase block operation usin
g the function BSP_QSPI_Erase_Block() and by
            specifying the block address. You
can perform an erase operation of the whole
            chip by calling the function BSP_Q
SPI_Erase_Chip().
       (++) The function BSP_QSPI_GetStatus()
returns the current status of the QSPI memory.

            (see the QSPI memory data sheet)
       (++) Perform erase sector operation usi
ng the function BSP_QSPI_Erase_Sector()
            which is not blocking. So the func
tion BSP_QSPI_GetStatus() should be used
            to check if the memory is busy, an
d the functions BSP_QSPI_SuspendErase()/
            BSP_QSPI_ResumeErase() can be used
 to perform other operations during the
            sector erase.
       (++) Deep power down of the QSPI memory
 is managed with the call of the functions
            BSP_QSPI_EnterDeepPowerDown()/BSP_
```

```
QSPI_LeaveDeepPowerDown()
```

Attention:

Definition in file **stm32l475e_iot01_qspi.c**.

B-L475E-IOT01 BSP User Manual

stm32l475e_iot01_qspi.h File Reference

This file contains the common defines and functions prototypes for the **stm32l475e_iot01_qspi.c** driver. More...

```
#include "stm32l4xx_hal.h" #include
"../Components/mx25r6435f/mx25r6435f.h"
```

Go to the source code of this file.

Data Structures

struct **QSPI_Info**

Defines

#define **QSPI_OK** ((uint8_t)0x00)
#define **QSPI_ERROR** ((uint8_t)0x01)
#define **QSPI_BUSY** ((uint8_t)0x02)
#define **QSPI_NOT_SUPPORTED** ((uint8_t)0x04)
#define **QSPI_SUSPENDED** ((uint8_t)0x08)

Functions

uint8_t	**BSP_QSPI_Init** (void) Initializes the QSPI interface.
uint8_t	**BSP_QSPI_DeInit** (void) De-Initializes the QSPI interface.
uint8_t	**BSP_QSPI_Read** (uint8_t *pData, uint32_t ReadAddr, uint32_t Size) Reads an amount of data from the QSPI memory.
uint8_t	**BSP_QSPI_Write** (uint8_t *pData, uint32_t WriteAddr, uint32_t Size) Writes an amount of data to the QSPI memory.
uint8_t	**BSP_QSPI_Erase_Block** (uint32_t BlockAddress) Erases the specified block of the QSPI memory.
uint8_t	**BSP_QSPI_Erase_Sector** (uint32_t Sector) Erases the specified sector of the QSPI memory.
uint8_t	**BSP_QSPI_Erase_Chip** (void) Erases the entire QSPI memory.
uint8_t	**BSP_QSPI_GetStatus** (void) Reads current status of the QSPI memory.
uint8_t	**BSP_QSPI_GetInfo** (**QSPI_Info** *pInfo) Return the configuration of the QSPI memory.
uint8_t	**BSP_QSPI_EnableMemoryMappedMode** (void) Configure the QSPI in memory-mapped mode.
uint8_t	**BSP_QSPI_SuspendErase** (void) This function suspends an ongoing erase command.
uint8_t	**BSP_QSPI_ResumeErase** (void) This function resumes a paused erase command.
uint8_t	**BSP_QSPI_EnterDeepPowerDown** (void) This function enter the QSPI memory in deep power down mode.
uint8_t	**BSP_QSPI_LeaveDeepPowerDown** (void) This function leave the QSPI memory from deep power

	down mode.
__weak void	**BSP_QSPI_MspInit** (void) Initializes the QSPI MSP.
__weak void	**BSP_QSPI_MspDeInit** (void) De-Initializes the QSPI MSP.

Detailed Description

This file contains the common defines and functions prototypes for the **stm32l475e_iot01_qspi.c** driver.

Author:
MCD Application Team

Version:
V1.0.0

Date:
17-March-2017

Attention:

Definition in file **stm32l475e_iot01_qspi.h**.

stm32l475e_iot01_tsensor.c File Reference

This file provides a set of functions needed to manage the temperature sensor. More...

```
#include "stm32l475e_iot01_tsensor.h"
```

Go to the source code of this file.

Functions

uint32_t	**BSP_TSENSOR_Init** (void) Initializes peripherals used by the I2C Temperature Sensor driver.
float	**BSP_TSENSOR_ReadTemp** (void) Read Temperature register of TS751.

Variables

static TSENSOR_DrvTypeDef * **tsensor_drv**

Detailed Description

This file provides a set of functions needed to manage the temperature sensor.

Author:
MCD Application Team

Version:
V1.0.0

Date:
17-March-2017

Attention:

OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file **stm32l475e_iot01_tsensor.c**.

B-L475E-IOT01 BSP User Manual

Enumerations | Functions

stm32l475e_iot01_tsensor.h File Reference

This file provides a set of functions needed to manage the temperature sensor. More...

```
#include "stm32l475e_iot01.h" #include
"../Components/hts221/hts221.h"
```

Go to the source code of this file.

Enumerations

enum	**TSENSOR_Status_TypDef** { **TSENSOR_OK** = 0, **TSENSOR_ERROR** } TSENSOR Status. More...

Functions

uint32_t	**BSP_TSENSOR_Init** (void) Initializes peripherals used by the I2C Temperature Sensor driver.
float	**BSP_TSENSOR_ReadTemp** (void) Read Temperature register of TS751.

Detailed Description

This file provides a set of functions needed to manage the temperature sensor.

Author:
MCD Application Team

Version:
V1.0.0

Date:
17-March-2017

Attention:

OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Definition in file **stm32l475e_iot01_tsensor.h**.

B-L475E-IOT01 BSP User Manual

Modules

Here is a list of all modules:

- BSP
  - STM32L475E_IOT01
    - LOW LEVEL
      - LOW LEVEL Private Def
      - LOW LEVEL Variables
      - LOW LEVEL Private Function Prototypes
      - LOW LEVEL Private Functions
      - LOW LEVEL Exported Types
      - LOW LEVEL Exported Constants
      - LOW LEVEL Exported Functions
    - ACCELERO
      - ACCELERO Private Variables
      - ACCELERO Private Functions
      - ACCELERO Exported Types
      - ACCELERO Exported Functions
    - GYROSCOPE
      - GYROSCOPE Private Variables
      - GYROSCOPE Private Functions
      - GYROSCOPE Exported Constants
      - GYROSCOPE Exported Functions
    - HUMIDITY
      - HUMIDITY Private Variables
      - HUMIDITY Private Functions
      - HUMIDITY Exported Types
      - HUMIDITY Exported Functions
    - MAGNETO
      - MAGNETO Private Variables

    - MAGNETO Private Functions
    - MAGNETO Exported Types
    - MAGNETO Exported Functions
- PRESSURE
    - PRESSURE Private Variables
    - PRESSURE Private Functions
    - PRESSURE Exported Types
    - PRESSURE Exported Functions
- QSPI
    - QSPI Private Constants
    - QSPI Private Variables
    - QSPI Private Functions
    - QSPI Exported Constants
    - QSPI Exported Types
    - QSPI Exported Functions
- TEMPERATURE
    - TEMPERATURE Private Variables
    - TEMPERATURE Private Functions
    - TEMPERATURE Exported Types
    - TEMPERATURE Exported Constants

B-L475E-IOT01 BSP User Manual

Data Structures

Here are the data structures with brief descriptions:

QSPI_Info

B-L475E-IOT01 BSP User Manual

File List

Here is a list of all files with brief descriptions:

stm32l475e_iot01.c [code]	STM32L475E-IOT01 board support package
stm32l475e_iot01.h [code]	STM32L475E IOT01 board support package
stm32l475e_iot01_accelero.c [code]	This file provides a set of functions needed to manage the accelerometer sensor
stm32l475e_iot01_accelero.h [code]	This file provides a set of functions needed to manage the accelerometer sensor
stm32l475e_iot01_gyro.c [code]	This file provides a set of functions needed to manage the gyroscope sensor
stm32l475e_iot01_gyro.h [code]	This file contains definitions for the **stm32l475e_iot01_gyro.c**
stm32l475e_iot01_hsensor.c [code]	This file provides a set of functions needed to manage the humidity sensor

stm32l475e_iot01_hsensor.h [code]	This file provides a set of functions needed to manage the humidity sensor
stm32l475e_iot01_magneto.c [code]	This file provides a set of functions needed to manage the magnetometer sensor
stm32l475e_iot01_magneto.h [code]	This file provides a set of functions needed to manage the magnetometer sensor
stm32l475e_iot01_psensor.c [code]	This file provides a set of functions needed to manage the pressure sensor
stm32l475e_iot01_psensor.h [code]	This file provides a set of functions needed to manage the pressure sensor
stm32l475e_iot01_qspi.c [code]	This file includes a standard driver for the MX25R6435F QSPI memory mounted on STM32L475E IOT01 board
stm32l475e_iot01_qspi.h [code]	This file contains the common defines and functions prototypes for the **stm32l475e_iot01_qspi.c** driver
stm32l475e_iot01_tsensor.c [code]	This file provides a set of functions needed to manage the temperature sensor

stm32l475e_iot01_tsensor.h [code]	This file provides a set of functions needed to manage the temperature sensor

B-L475E-IOT01 BSP User Manual

Directories

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

- Drivers
  - BSP
    - B-L475E-IOT01

B-L475E-IOT01 BSP User Manual

Modules | Functions

LOW LEVEL

STM32L475E_IOT01

Modules

- LOW LEVEL Private Def
- LOW LEVEL Variables
- LOW LEVEL Private Function Prototypes
- LOW LEVEL Private Functions
- LOW LEVEL Exported Types
- LOW LEVEL Exported Constants
- LOW LEVEL Exported Functions

Functions

void	**SENSOR_IO_Init** (void) Initializes Sensors low level.
void	**SENSOR_IO_DeInit** (void) DeInitializes Sensors low level.
void	**SENSOR_IO_Write** (uint8_t Addr, uint8_t Reg, uint8_t Value) Writes a single data.
uint8_t	**SENSOR_IO_Read** (uint8_t Addr, uint8_t Reg) Reads a single data.
uint16_t	**SENSOR_IO_ReadMultiple** (uint8_t Addr, uint8_t Reg, uint8_t *Buffer, uint16_t Length) Reads multiple data with I2C communication channel from TouchScreen.
void	**SENSOR_IO_WriteMultiple** (uint8_t Addr, uint8_t Reg, uint8_t *Buffer, uint16_t Length) Writes multiple data with I2C communication channel from MCU to TouchScreen.
HAL_StatusTypeDef	**SENSOR_IO_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.
void	**SENSOR_IO_Delay** (uint32_t Delay) Delay function used in TouchScreen low level driver.

Function Documentation

void SENSOR_IO_DeInit (void)

DeInitializes Sensors low level.

Return values:
None

Definition at line **565** of file **stm32l475e_iot01.c**.

References **hI2cHandler**, and **I2Cx_DeInit()**.

void SENSOR_IO_Delay (uint32_t Delay)

Delay function used in TouchScreen low level driver.

Parameters:
Delay,: Delay in ms

Return values:
None

Definition at line **642** of file **stm32l475e_iot01.c**.

void SENSOR_IO_Init (void)

Initializes Sensors low level.

Return values:
None

Definition at line **556** of file **stm32l475e_iot01.c**.

References **hI2cHandler**, and **I2Cx_Init()**.

Referenced by **BSP_TSENSOR_Init()**.

HAL_StatusTypeDef SENSOR_IO_IsDeviceReady (uint16_t DevAd
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 **632** of file **stm32l475e_iot01.c**.

References **hI2cHandler**, and **I2Cx_IsDeviceReady()**.

uint8_t SENSOR_IO_Read (uint8_t Addr,
uint8_t Reg
)

Reads a single data.

Parameters:

Addr,: I2C address
Reg,: Reg address

Return values:

Data to be read

Definition at line **588** of file **stm32l475e_iot01.c**.

References **hI2cHandler**, and **I2Cx_ReadMultiple()**.

uint16_t SENSOR_IO_ReadMultiple (
uint8_t Addr,
uint8_t Reg,
uint8_t * Buffer,
uint16_t Length
)

Reads multiple data with I2C communication channel from TouchScreen.

Parameters:

Addr,: I2C address
Reg,: Register address
Buffer,: Pointer to data buffer
Length,: Length of the data

Return values:

HAL status

Definition at line **606** of file **stm32l475e_iot01.c**.

References **hI2cHandler**, and **I2Cx_ReadMultiple()**.

void SENSOR_IO_Write (
uint8_t Addr,
uint8_t Reg,
uint8_t Value
)

Writes a single data.

Parameters:

Addr,: I2C address

Reg,: Reg address

Value,: Data to be written

Return values:

None

Definition at line **577** of file **stm32l475e_iot01.c**.

References **hI2cHandler**, and **I2Cx_WriteMultiple()**.

```
void SENSOR_IO_WriteMultiple ( uint8_t    Addr,
                               uint8_t    Reg,
                               uint8_t *  Buffer,
                               uint16_t   Length
                             )
```

Writes multiple data with I2C communication channel from MCU to TouchScreen.

Parameters:

Addr,: I2C address

Reg,: Register address

Buffer,: Pointer to data buffer

Length,: Length of the data

Return values:

None

Definition at line **620** of file **stm32l475e_iot01.c**.

References **hI2cHandler**, and **I2Cx_WriteMultiple()**.

B-L475E-IOT01 BSP User Manual

Data Structure Index

Q

Q

QSPI_Info

Q

Generated on Thu Mar 16 2017 10:38:33 for B-L475E-IOT01 BSP User Manual by doxygen 1.7.6.1

B-L475E-IOT01 BSP User Manual

QSPI Exported Types

QSPI

Data Structures

struct	**QSPI_Info**

B-L475E-IOT01 BSP User Manual

stm32l475e_iot01_qspi.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l475e_iot01_qspi.h
00004   * @author  MCD Application Team
00005   * @version V1.0.0
00006   * @date    17-March-2017
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm32l475e_iot01_qspi.c driver.
00009   ******************************************************************************
00010   * @attention
00011   *
00012   * <h2><center>&copy; COPYRIGHT(c) 2017 STMicroelectronics</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without modification,
00015   * are permitted provided that the following conditions are met:
00016   *   1. Redistributions of source code must retain the above copyright notice,
```

```
00017   *      this list of conditions and the fol
lowing disclaimer.
00018   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00019   *      this list of conditions and the fol
lowing disclaimer in the documentation
00020   *      and/or other materials provided wit
h the distribution.
00021   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00022   *      may be used to endorse or promote p
roducts derived from this software
00023   *      without specific prior written perm
ission.
00024   *
00025   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00026   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00027   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00028   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00029   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00030   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
00031   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00032   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00033   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00034   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00035   *
00036   ****************************************
```

```
**********************************
00037   */
00038
00039 /* Define to prevent recursive inclusion ---
-----------------------------------*/
00040 #ifndef __STM32L475E_IOT01_QSPI_H
00041 #define __STM32L475E_IOT01_QSPI_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------
-----------------------------------*/
00048 #include "stm32l4xx_hal.h"
00049 #include "../Components/mx25r6435f/mx25r6435
f.h"
00050
00051 /** @addtogroup BSP
00052   * @{
00053   */
00054
00055 /** @addtogroup STM32L475E_IOT01
00056   * @{
00057   */
00058
00059 /** @addtogroup STM32L475E_IOT01_QSPI
00060   * @{
00061   */
00062
00063 /* Exported constants --------------------
-----------------------------------*/
00064 /** @defgroup STM32L475E_IOT01_QSPI_Exported
_Constants QSPI Exported Constants
00065   * @{
00066   */
00067 /* QSPI Error codes */
```

```
00068 #define QSPI_OK            ((uint8_t)0x00)
00069 #define QSPI_ERROR         ((uint8_t)0x01)
00070 #define QSPI_BUSY          ((uint8_t)0x02)
00071 #define QSPI_NOT_SUPPORTED ((uint8_t)0x04)
00072 #define QSPI_SUSPENDED     ((uint8_t)0x08)
00073
00074 /**
00075   * @}
00076   */
00077
00078 /* Exported types ---------------------------
-----------------------------------*/
00079 /** @defgroup STM32L475E_IOT01_QSPI_Exported
_Types QSPI Exported Types
00080   * @{
00081   */
00082 /* QSPI Info */
00083 typedef struct {
00084   uint32_t FlashSize;          /*!< Size of
the flash */
00085   uint32_t EraseSectorSize;    /*!< Size of
sectors for the erase operation */
00086   uint32_t EraseSectorsNumber; /*!< Number o
f sectors for the erase operation */
00087   uint32_t ProgPageSize;       /*!< Size of
pages for the program operation */
00088   uint32_t ProgPagesNumber;    /*!< Number o
f pages for the program operation */
00089 } QSPI_Info;
00090
00091 /**
00092   * @}
00093   */
00094
00095 /* Exported functions ----------------------
-----------------------------------*/
00096 /** @defgroup STM32L475E_IOT01_QSPI_Exported
```

```
_Functions QSPI Exported Functions
00097   * @{
00098   */
00099 uint8_t BSP_QSPI_Init                    (void
);
00100 uint8_t BSP_QSPI_DeInit                  (void
);
00101 uint8_t BSP_QSPI_Read                    (uint
8_t* pData, uint32_t ReadAddr, uint32_t Size);
00102 uint8_t BSP_QSPI_Write                   (uint
8_t* pData, uint32_t WriteAddr, uint32_t Size);
00103 uint8_t BSP_QSPI_Erase_Block             (uint
32_t BlockAddress);
00104 uint8_t BSP_QSPI_Erase_Sector            (uint
32_t Sector);
00105 uint8_t BSP_QSPI_Erase_Chip              (void
);
00106 uint8_t BSP_QSPI_GetStatus               (void
);
00107 uint8_t BSP_QSPI_GetInfo                 (QSPI
_Info* pInfo);
00108 uint8_t BSP_QSPI_EnableMemoryMappedMode(void
);
00109 uint8_t BSP_QSPI_SuspendErase            (void
);
00110 uint8_t BSP_QSPI_ResumeErase             (void
);
00111 uint8_t BSP_QSPI_EnterDeepPowerDown      (void
);
00112 uint8_t BSP_QSPI_LeaveDeepPowerDown      (void
);
00113
00114 void BSP_QSPI_MspInit(void);
00115 void BSP_QSPI_MspDeInit(void);
00116 /**
00117   * @}
00118   */
```

```
00119
00120 /**
00121   * @}
00122   */
00123
00124 /**
00125   * @}
00126   */
00127
00128 /**
00129   * @}
00130   */
00131
00132 #ifdef __cplusplus
00133 }
00134 #endif
00135
00136 #endif /* __STM32L475E_IOT01_QSPI_H */
00137
00138 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

B-L475E-IOT01 BSP User Manual

stm32l475e_iot01_qspi.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l475e_iot01_qspi.c
00004   * @author  MCD Application Team
00005   * @version V1.0.0
00006   * @date    17-March-2017
00007   * @brief   This file includes a standard driver for the MX25R6435F QSPI
00008   *          memory mounted on STM32L475E IOT01 board.
00009   @verbatim
00010   ==============================================================================
00011                         ##### How to use this driver #####
00012   ==============================================================================
00013   [..]
00014    (#) This driver is used to drive the MX25R6435F QSPI external
00015        memory mounted on STM32L475E IOT01 board.
00016
```

```
00017    (#) This driver need a specific component
 driver (MX25R6435F) to be included with.
00018
00019    (#) Initialization steps:
00020        (++) Initialize the QPSI external mem
ory using the BSP_QSPI_Init() function. This
00021             function includes the MSP layer
hardware resources initialization and the
00022             QSPI interface with the external
 memory. The BSP_QSPI_DeInit() can be used
00023             to deactivate the QSPI interface.

00024
00025    (#) QSPI memory operations
00026        (++) QSPI memory can be accessed with
 read/write operations once it is
00027             initialized.
00028             Read/write operation can be perf
ormed with AHB access using the functions
00029             BSP_QSPI_Read()/BSP_QSPI_Write().

00030        (++) The function to the QSPI memory
in memory-mapped mode is possible after
00031             the call of the function BSP_QSP
I_EnableMemoryMappedMode().
00032        (++) The function BSP_QSPI_GetInfo()
returns the configuration of the QSPI memory.
00033             (see the QSPI memory data sheet)
00034        (++) Perform erase block operation us
ing the function BSP_QSPI_Erase_Block() and by
00035             specifying the block address. Yo
u can perform an erase operation of the whole
00036             chip by calling the function BSP
_QSPI_Erase_Chip().
00037        (++) The function BSP_QSPI_GetStatus(
) returns the current status of the QSPI memory.
00038             (see the QSPI memory data sheet)
```

```
00039         (++) Perform erase sector operation u
sing the function BSP_QSPI_Erase_Sector()
00040              which is not blocking. So the fu
nction BSP_QSPI_GetStatus() should be used
00041              to check if the memory is busy,
and the functions BSP_QSPI_SuspendErase()/
00042              BSP_QSPI_ResumeErase() can be us
ed to perform other operations during the
00043              sector erase.
00044         (++) Deep power down of the QSPI memo
ry is managed with the call of the functions
00045              BSP_QSPI_EnterDeepPowerDown()/BS
P_QSPI_LeaveDeepPowerDown()
00046   @endverbatim
00047   ******************************************
**********************************
00048   * @attention
00049   *
00050   * <h2><center>&copy; COPYRIGHT(c) 2017 STM
icroelectronics</center></h2>
00051   *
00052   * Redistribution and use in source and bin
ary forms, with or without modification,
00053   * are permitted provided that the followin
g conditions are met:
00054   *   1. Redistributions of source code must
 retain the above copyright notice,
00055   *      this list of conditions and the fol
lowing disclaimer.
00056   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00057   *      this list of conditions and the fol
lowing disclaimer in the documentation
00058   *      and/or other materials provided wit
h the distribution.
00059   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
```

```
00060   *      may be used to endorse or promote p
roducts derived from this software
00061   *      without specific prior written perm
ission.
00062   *
00063   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00064   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00065   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00066   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00067   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00068   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
00069   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00070   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00071   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00072   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00073   *
00074   ****************************************
**********************************
00075   */
00076
00077 /* Includes ------------------------------
-----------------------------------*/
00078 #include "stm32l475e_iot01_qspi.h"
00079
00080 /** @addtogroup BSP
00081   * @{
00082   */
```

```
00083
00084 /** @addtogroup STM32L475E_IOT01
00085   * @{
00086   */
00087
00088 /** @defgroup STM32L475E_IOT01_QSPI QSPI
00089   * @{
00090   */
00091
00092 /* Private constants ---------------------
---------------------------------*/
00093 /** @defgroup STM32L475E_IOT01_QSPI_Private_
Constants QSPI Private Constants
00094   * @{
00095   */
00096 #define QSPI_QUAD_DISABLE       0x0
00097 #define QSPI_QUAD_ENABLE        0x1
00098
00099 #define QSPI_HIGH_PERF_DISABLE  0x0
00100 #define QSPI_HIGH_PERF_ENABLE   0x1
00101 /**
00102   * @}
00103   */
00104 /* Private variables ---------------------
---------------------------------*/
00105
00106 /** @defgroup STM32L475E_IOT01_QSPI_Private_
Variables QSPI Private Variables
00107   * @{
00108   */
00109 QSPI_HandleTypeDef QSPIHandle;
00110
00111 /**
00112   * @}
00113   */
00114
00115
```

```
00116 /* Private functions ---------------------
-----------------------------------*/
00117
00118 /** @defgroup STM32L475E_IOT01_QSPI_Private_
Functions QSPI Private Functions
00119   * @{
00120   */
00121 static uint8_t QSPI_ResetMemory        (QSPI
_HandleTypeDef *hqspi);
00122 static uint8_t QSPI_WriteEnable        (QSPI
_HandleTypeDef *hqspi);
00123 static uint8_t QSPI_AutoPollingMemReady(QSPI
_HandleTypeDef *hqspi, uint32_t Timeout);
00124 static uint8_t QSPI_QuadMode           (QSPI
_HandleTypeDef *hqspi, uint8_t Operation);
00125 static uint8_t QSPI_HighPerfMode       (QSPI
_HandleTypeDef *hqspi, uint8_t Operation);
00126
00127 /**
00128   * @}
00129   */
00130
00131 /* Exported functions ---------------------
-----------------------------------*/
00132
00133 /** @addtogroup STM32L475E_IOT01_QSPI_Export
ed_Functions
00134   * @{
00135   */
00136
00137 /**
00138   * @brief  Initializes the QSPI interface.
00139   * @retval QSPI memory status
00140   */
00141 uint8_t BSP_QSPI_Init(void)
00142 {
00143   QSPIHandle.Instance = QUADSPI;
```

```
00144
00145   /* Call the DeInit function to reset the d
river */
00146   if (HAL_QSPI_DeInit(&QSPIHandle) != HAL_OK
)
00147   {
00148     return QSPI_ERROR;
00149   }
00150
00151   /* System level initialization */
00152   BSP_QSPI_MspInit();
00153
00154   /* QSPI initialization */
00155   QSPIHandle.Init.ClockPrescaler     = 2; /*
 QSPI clock = 80MHz / (ClockPrescaler+1) = 26.67MH
z */
00156   QSPIHandle.Init.FifoThreshold      = 4;
00157   QSPIHandle.Init.SampleShifting     = QSPI_
SAMPLE_SHIFTING_NONE;
00158   QSPIHandle.Init.FlashSize          = POSIT
ION_VAL(MX25R6435F_FLASH_SIZE) - 1;
00159   QSPIHandle.Init.ChipSelectHighTime = QSPI_
CS_HIGH_TIME_1_CYCLE;
00160   QSPIHandle.Init.ClockMode          = QSPI_
CLOCK_MODE_0;
00161
00162   if (HAL_QSPI_Init(&QSPIHandle) != HAL_OK)
00163   {
00164     return QSPI_ERROR;
00165   }
00166
00167   /* QSPI memory reset */
00168   if (QSPI_ResetMemory(&QSPIHandle) != QSPI_
OK)
00169   {
00170     return QSPI_NOT_SUPPORTED;
00171   }
```

```
00172
00173   /* QSPI quad enable */
00174   if (QSPI_QuadMode(&QSPIHandle, QSPI_QUAD_E
NABLE) != QSPI_OK)
00175   {
00176     return QSPI_ERROR;
00177   }
00178
00179   /* High performance mode enable */
00180   if (QSPI_HighPerfMode(&QSPIHandle, QSPI_HI
GH_PERF_ENABLE) != QSPI_OK)
00181   {
00182     return QSPI_ERROR;
00183   }
00184
00185   /* Re-configure the clock for the high per
formance mode */
00186   QSPIHandle.Init.ClockPrescaler = 1; /* QSP
I clock = 80MHz / (ClockPrescaler+1) = 40MHz */
00187
00188   if (HAL_QSPI_Init(&QSPIHandle) != HAL_OK)
00189   {
00190     return QSPI_ERROR;
00191   }
00192
00193   return QSPI_OK;
00194 }
00195
00196 /**
00197   * @brief  De-Initializes the QSPI interfac
e.
00198   * @retval QSPI memory status
00199   */
00200 uint8_t BSP_QSPI_DeInit(void)
00201 {
00202   QSPIHandle.Instance = QUADSPI;
00203
```

```
00204   /* Call the DeInit function to reset the d
river */
00205   if (HAL_QSPI_DeInit(&QSPIHandle) != HAL_OK
)
00206   {
00207     return QSPI_ERROR;
00208   }
00209
00210   /* System level De-initialization */
00211   BSP_QSPI_MspDeInit();
00212
00213   return QSPI_OK;
00214 }
00215
00216 /**
00217   * @brief  Reads an amount of data from the
 QSPI memory.
00218   * @param  pData    : Pointer to data to be
 read
00219   * @param  ReadAddr : Read start address
00220   * @param  Size     : Size of data to read

00221   * @retval QSPI memory status
00222   */
00223 uint8_t BSP_QSPI_Read(uint8_t* pData, uint32
_t ReadAddr, uint32_t Size)
00224 {
00225   QSPI_CommandTypeDef sCommand;
00226
00227   /* Initialize the read command */
00228   sCommand.InstructionMode    = QSPI_INSTRUC
TION_1_LINE;
00229   sCommand.Instruction        = QUAD_INOUT_R
EAD_CMD;
00230   sCommand.AddressMode        = QSPI_ADDRESS
_4_LINES;
00231   sCommand.AddressSize        = QSPI_ADDRESS
```

```
_24_BITS;
00232   sCommand.Address           = ReadAddr;
00233   sCommand.AlternateByteMode  = QSPI_ALTERNA
TE_BYTES_4_LINES;
00234   sCommand.AlternateBytesSize = QSPI_ALTERNA
TE_BYTES_8_BITS;
00235   sCommand.AlternateBytes     = MX25R6435F_A
LT_BYTES_NO_PE_MODE;
00236   sCommand.DataMode           = QSPI_DATA_4_
LINES;
00237   sCommand.DummyCycles        = MX25R6435F_D
UMMY_CYCLES_READ_QUAD;
00238   sCommand.NbData             = Size;
00239   sCommand.DdrMode            = QSPI_DDR_MOD
E_DISABLE;
00240   sCommand.DdrHoldHalfCycle   = QSPI_DDR_HHC
_ANALOG_DELAY;
00241   sCommand.SIOOMode           = QSPI_SIOO_IN
ST_EVERY_CMD;
00242
00243   /* Configure the command */
00244   if (HAL_QSPI_Command(&QSPIHandle, &sComman
d, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00245   {
00246     return QSPI_ERROR;
00247   }
00248
00249   /* Reception of the data */
00250   if (HAL_QSPI_Receive(&QSPIHandle, pData, H
AL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00251   {
00252     return QSPI_ERROR;
00253   }
00254
00255   return QSPI_OK;
00256 }
00257
```

```
00258 /**
00259   * @brief  Writes an amount of data to the
QSPI memory.
00260   * @param  pData     : Pointer to data to b
e written
00261   * @param  WriteAddr : Write start address
00262   * @param  Size      : Size of data to writ
e
00263   * @retval QSPI memory status
00264   */
00265 uint8_t BSP_QSPI_Write(uint8_t* pData, uint3
2_t WriteAddr, uint32_t Size)
00266 {
00267   QSPI_CommandTypeDef sCommand;
00268   uint32_t end_addr, current_size, current_a
ddr;
00269
00270   /* Calculation of the size between the wri
te address and the end of the page */
00271   current_size = MX25R6435F_PAGE_SIZE - (Wri
teAddr % MX25R6435F_PAGE_SIZE);
00272
00273   /* Check if the size of the data is less t
han the remaining place in the page */
00274   if (current_size > Size)
00275   {
00276     current_size = Size;
00277   }
00278
00279   /* Initialize the adress variables */
00280   current_addr = WriteAddr;
00281   end_addr = WriteAddr + Size;
00282
00283   /* Initialize the program command */
00284   sCommand.InstructionMode   = QSPI_INSTRUCT
ION_1_LINE;
00285   sCommand.Instruction       = QUAD_PAGE_PRO
```

```
G_CMD;
00286   sCommand.AddressMode       = QSPI_ADDRESS_
4_LINES;
00287   sCommand.AddressSize       = QSPI_ADDRESS_
24_BITS;
00288   sCommand.AlternateByteMode = QSPI_ALTERNAT
E_BYTES_NONE;
00289   sCommand.DataMode          = QSPI_DATA_4_L
INES;
00290   sCommand.DummyCycles       = 0;
00291   sCommand.DdrMode           = QSPI_DDR_MODE
_DISABLE;
00292   sCommand.DdrHoldHalfCycle  = QSPI_DDR_HHC_
ANALOG_DELAY;
00293   sCommand.SIOOMode          = QSPI_SIOO_INS
T_EVERY_CMD;
00294
00295   /* Perform the write page by page */
00296   do
00297   {
00298     sCommand.Address = current_addr;
00299     sCommand.NbData  = current_size;
00300
00301     /* Enable write operations */
00302     if (QSPI_WriteEnable(&QSPIHandle) != QSP
I_OK)
00303     {
00304       return QSPI_ERROR;
00305     }
00306
00307     /* Configure the command */
00308     if (HAL_QSPI_Command(&QSPIHandle, &sComm
and, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00309     {
00310       return QSPI_ERROR;
00311     }
00312
```

```
00313     /* Transmission of the data */
00314     if (HAL_QSPI_Transmit(&QSPIHandle, pData
, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00315     {
00316       return QSPI_ERROR;
00317     }
00318
00319     /* Configure automatic polling mode to w
ait for end of program */
00320     if (QSPI_AutoPollingMemReady(&QSPIHandle
, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != QSPI_OK)
00321     {
00322       return QSPI_ERROR;
00323     }
00324
00325     /* Update the address and size variables
 for next page programming */
00326     current_addr += current_size;
00327     pData += current_size;
00328     current_size = ((current_addr + MX25R643
5F_PAGE_SIZE) > end_addr) ? (end_addr - current_ad
dr) : MX25R6435F_PAGE_SIZE;
00329   } while (current_addr < end_addr);
00330
00331   return QSPI_OK;
00332 }
00333
00334 /**
00335   * @brief  Erases the specified block of th
e QSPI memory.
00336   * @param  BlockAddress : Block address to
erase
00337   * @retval QSPI memory status
00338   */
00339 uint8_t BSP_QSPI_Erase_Block(uint32_t BlockA
ddress)
00340 {
```

```
00341   QSPI_CommandTypeDef sCommand;
00342
00343   /* Initialize the erase command */
00344   sCommand.InstructionMode   = QSPI_INSTRUCT
ION_1_LINE;
00345   sCommand.Instruction       = BLOCK_ERASE_C
MD;
00346   sCommand.AddressMode       = QSPI_ADDRESS_
1_LINE;
00347   sCommand.AddressSize       = QSPI_ADDRESS_
24_BITS;
00348   sCommand.Address           = BlockAddress;
00349   sCommand.AlternateByteMode = QSPI_ALTERNAT
E_BYTES_NONE;
00350   sCommand.DataMode          = QSPI_DATA_NON
E;
00351   sCommand.DummyCycles       = 0;
00352   sCommand.DdrMode           = QSPI_DDR_MODE
_DISABLE;
00353   sCommand.DdrHoldHalfCycle  = QSPI_DDR_HHC_
ANALOG_DELAY;
00354   sCommand.SIOOMode          = QSPI_SIOO_INS
T_EVERY_CMD;
00355
00356   /* Enable write operations */
00357   if (QSPI_WriteEnable(&QSPIHandle) != QSPI_
OK)
00358   {
00359     return QSPI_ERROR;
00360   }
00361
00362   /* Send the command */
00363   if (HAL_QSPI_Command(&QSPIHandle, &Comman
d, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00364   {
00365     return QSPI_ERROR;
00366   }
```

```
00367
00368   /* Configure automatic polling mode to wai
t for end of erase */
00369   if (QSPI_AutoPollingMemReady(&QSPIHandle,
MX25R6435F_BLOCK_ERASE_MAX_TIME) != QSPI_OK)
00370   {
00371     return QSPI_ERROR;
00372   }
00373
00374   return QSPI_OK;
00375 }
00376
00377 /**
00378   * @brief  Erases the specified sector of t
he QSPI memory.
00379   * @param  Sector : Sector address to erase
 (0 to 255)
00380   * @retval QSPI memory status
00381   * @note This function is non blocking mean
ing that sector erase
00382   *       operation is started but not compl
eted when the function
00383   *       returns. Application has to call B
SP_QSPI_GetStatus()
00384   *       to know when the device is availab
le again (i.e. erase operation
00385   *       completed).
00386   */
00387 uint8_t BSP_QSPI_Erase_Sector(uint32_t Secto
r)
00388 {
00389   QSPI_CommandTypeDef sCommand;
00390
00391   if (Sector >= (uint32_t)(MX25R6435F_FLASH_
SIZE/MX25R6435F_SECTOR_SIZE))
00392   {
00393     return QSPI_ERROR;
```

```
00394   }
00395
00396   /* Initialize the erase command */
00397   sCommand.InstructionMode   = QSPI_INSTRUCT
ION_1_LINE;
00398   sCommand.Instruction       = SECTOR_ERASE_
CMD;
00399   sCommand.AddressMode       = QSPI_ADDRESS_
1_LINE;
00400   sCommand.AddressSize       = QSPI_ADDRESS_
24_BITS;
00401   sCommand.Address           = (Sector * MX2
5R6435F_SECTOR_SIZE);
00402   sCommand.AlternateByteMode = QSPI_ALTERNAT
E_BYTES_NONE;
00403   sCommand.DataMode          = QSPI_DATA_NON
E;
00404   sCommand.DummyCycles       = 0;
00405   sCommand.DdrMode           = QSPI_DDR_MODE
_DISABLE;
00406   sCommand.DdrHoldHalfCycle  = QSPI_DDR_HHC_
ANALOG_DELAY;
00407   sCommand.SIOOMode          = QSPI_SIOO_INS
T_EVERY_CMD;
00408
00409   /* Enable write operations */
00410   if (QSPI_WriteEnable(&QSPIHandle) != QSPI_
OK)
00411   {
00412     return QSPI_ERROR;
00413   }
00414
00415   /* Send the command */
00416   if (HAL_QSPI_Command(&QSPIHandle, &sComman
d, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00417   {
00418     return QSPI_ERROR;
```

```
00419   }
00420
00421   return QSPI_OK;
00422 }
00423
00424 /**
00425   * @brief  Erases the entire QSPI memory.
00426   * @retval QSPI memory status
00427   */
00428 uint8_t BSP_QSPI_Erase_Chip(void)
00429 {
00430   QSPI_CommandTypeDef sCommand;
00431
00432   /* Initialize the erase command */
00433   sCommand.InstructionMode   = QSPI_INSTRUCT
ION_1_LINE;
00434   sCommand.Instruction       = CHIP_ERASE_CM
D;
00435   sCommand.AddressMode       = QSPI_ADDRESS_
NONE;
00436   sCommand.AlternateByteMode = QSPI_ALTERNAT
E_BYTES_NONE;
00437   sCommand.DataMode          = QSPI_DATA_NON
E;
00438   sCommand.DummyCycles       = 0;
00439   sCommand.DdrMode           = QSPI_DDR_MODE
_DISABLE;
00440   sCommand.DdrHoldHalfCycle  = QSPI_DDR_HHC_
ANALOG_DELAY;
00441   sCommand.SIOOMode          = QSPI_SIOO_INS
T_EVERY_CMD;
00442
00443   /* Enable write operations */
00444   if (QSPI_WriteEnable(&QSPIHandle) != QSPI_
OK)
00445   {
00446     return QSPI_ERROR;
```

```
00447   }
00448
00449   /* Send the command */
00450   if (HAL_QSPI_Command(&QSPIHandle, &sComman
d, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00451   {
00452     return QSPI_ERROR;
00453   }
00454
00455   /* Configure automatic polling mode to wai
t for end of erase */
00456   if (QSPI_AutoPollingMemReady(&QSPIHandle,
MX25R6435F_CHIP_ERASE_MAX_TIME) != QSPI_OK)
00457   {
00458     return QSPI_ERROR;
00459   }
00460
00461   return QSPI_OK;
00462 }
00463
00464 /**
00465   * @brief  Reads current status of the QSPI
 memory.
00466   * @retval QSPI memory status
00467   */
00468 uint8_t BSP_QSPI_GetStatus(void)
00469 {
00470   QSPI_CommandTypeDef sCommand;
00471   uint8_t reg;
00472
00473   /* Initialize the read security register c
ommand */
00474   sCommand.InstructionMode   = QSPI_INSTRUCT
ION_1_LINE;
00475   sCommand.Instruction       = READ_SEC_REG_
CMD;
00476   sCommand.AddressMode       = QSPI_ADDRESS_
```

```
NONE;
00477   sCommand.AlternateByteMode = QSPI_ALTERNAT
E_BYTES_NONE;
00478   sCommand.DataMode          = QSPI_DATA_1_L
INE;
00479   sCommand.DummyCycles       = 0;
00480   sCommand.NbData            = 1;
00481   sCommand.DdrMode           = QSPI_DDR_MODE
_DISABLE;
00482   sCommand.DdrHoldHalfCycle  = QSPI_DDR_HHC_
ANALOG_DELAY;
00483   sCommand.SIOOMode          = QSPI_SIOO_INS
T_EVERY_CMD;
00484
00485   /* Configure the command */
00486   if (HAL_QSPI_Command(&QSPIHandle, &sComman
d, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00487   {
00488     return QSPI_ERROR;
00489   }
00490
00491   /* Reception of the data */
00492   if (HAL_QSPI_Receive(&QSPIHandle, &reg, HA
L_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00493   {
00494     return QSPI_ERROR;
00495   }
00496
00497   /* Check the value of the register */
00498   if ((reg & (MX25R6435F_SECR_P_FAIL | MX25R
6435F_SECR_E_FAIL)) != 0)
00499   {
00500     return QSPI_ERROR;
00501   }
00502   else if ((reg & (MX25R6435F_SECR_PSB | MX2
5R6435F_SECR_ESB)) != 0)
00503   {
```

```
00504     return QSPI_SUSPENDED;
00505   }
00506
00507   /* Initialize the read status register com
mand */
00508   sCommand.Instruction       = READ_STATUS_R
EG_CMD;
00509
00510   /* Configure the command */
00511   if (HAL_QSPI_Command(&QSPIHandle, &sComman
d, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00512   {
00513     return QSPI_ERROR;
00514   }
00515
00516   /* Reception of the data */
00517   if (HAL_QSPI_Receive(&QSPIHandle, &reg, HA
L_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00518   {
00519     return QSPI_ERROR;
00520   }
00521
00522   /* Check the value of the register */
00523   if ((reg & MX25R6435F_SR_WIP) != 0)
00524   {
00525     return QSPI_BUSY;
00526   }
00527   else
00528   {
00529     return QSPI_OK;
00530   }
00531 }
00532
00533 /**
00534   * @brief  Return the configuration of the
QSPI memory.
00535   * @param  pInfo : pointer on the configura
```

```
tion structure
00536   * @retval QSPI memory status
00537   */
00538 uint8_t BSP_QSPI_GetInfo(QSPI_Info* pInfo)
00539 {
00540   /* Configure the structure with the memory
 configuration */
00541   pInfo->FlashSize          = MX25R6435F_FLA
SH_SIZE;
00542   pInfo->EraseSectorSize    = MX25R6435F_SEC
TOR_SIZE;
00543   pInfo->EraseSectorsNumber = (MX25R6435F_FL
ASH_SIZE/MX25R6435F_SECTOR_SIZE);
00544   pInfo->ProgPageSize       = MX25R6435F_PAG
E_SIZE;
00545   pInfo->ProgPagesNumber    = (MX25R6435F_FL
ASH_SIZE/MX25R6435F_PAGE_SIZE);
00546
00547   return QSPI_OK;
00548 }
00549
00550 /**
00551   * @brief  Configure the QSPI in memory-map
ped mode
00552   * @retval QSPI memory status
00553   */
00554 uint8_t BSP_QSPI_EnableMemoryMappedMode(void
)
00555 {
00556   QSPI_CommandTypeDef      sCommand;
00557   QSPI_MemoryMappedTypeDef sMemMappedCfg;
00558
00559   /* Configure the command for the read inst
ruction */
00560   sCommand.InstructionMode    = QSPI_INSTRUC
TION_1_LINE;
00561   sCommand.Instruction        = QUAD_INOUT_R
```

```
EAD_CMD;
00562   sCommand.AddressMode        = QSPI_ADDRESS
_4_LINES;
00563   sCommand.AddressSize        = QSPI_ADDRESS
_24_BITS;
00564   sCommand.AlternateByteMode  = QSPI_ALTERNA
TE_BYTES_4_LINES;
00565   sCommand.AlternateBytesSize = QSPI_ALTERNA
TE_BYTES_8_BITS;
00566   sCommand.AlternateBytes     = MX25R6435F_A
LT_BYTES_NO_PE_MODE;
00567   sCommand.DataMode           = QSPI_DATA_4_
LINES;
00568   sCommand.DummyCycles        = MX25R6435F_D
UMMY_CYCLES_READ_QUAD;
00569   sCommand.DdrMode            = QSPI_DDR_MOD
E_DISABLE;
00570   sCommand.DdrHoldHalfCycle   = QSPI_DDR_HHC
_ANALOG_DELAY;
00571   sCommand.SIOOMode           = QSPI_SIOO_IN
ST_EVERY_CMD;
00572
00573   /* Configure the memory mapped mode */
00574   sMemMappedCfg.TimeOutActivation = QSPI_TIM
EOUT_COUNTER_DISABLE;
00575
00576   if (HAL_QSPI_MemoryMapped(&QSPIHandle, &sC
ommand, &sMemMappedCfg) != HAL_OK)
00577   {
00578     return QSPI_ERROR;
00579   }
00580
00581   return QSPI_OK;
00582 }
00583
00584 /**
00585   * @brief  This function suspends an ongoin
```

```
g erase command.
00586   * @retval QSPI memory status
00587   */
00588 uint8_t BSP_QSPI_SuspendErase(void)
00589 {
00590   QSPI_CommandTypeDef sCommand;
00591
00592   /* Check whether the device is busy (erase
 operation is
00593   in progress).
00594   */
00595   if (BSP_QSPI_GetStatus() == QSPI_BUSY)
00596   {
00597     /* Initialize the erase command */
00598     sCommand.InstructionMode   = QSPI_INSTRU
CTION_1_LINE;
00599     sCommand.Instruction       = PROG_ERASE_
SUSPEND_CMD;
00600     sCommand.AddressMode       = QSPI_ADDRES
S_NONE;
00601     sCommand.AlternateByteMode = QSPI_ALTERN
ATE_BYTES_NONE;
00602     sCommand.DataMode          = QSPI_DATA_N
ONE;
00603     sCommand.DummyCycles       = 0;
00604     sCommand.DdrMode           = QSPI_DDR_MO
DE_DISABLE;
00605     sCommand.DdrHoldHalfCycle  = QSPI_DDR_HH
C_ANALOG_DELAY;
00606     sCommand.SIOOMode          = QSPI_SIOO_I
NST_EVERY_CMD;
00607
00608     /* Send the command */
00609     if (HAL_QSPI_Command(&QSPIHandle, &sComm
and, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00610     {
00611       return QSPI_ERROR;
```

```
00612     }
00613
00614     if (BSP_QSPI_GetStatus() == QSPI_SUSPEND
ED)
00615     {
00616       return QSPI_OK;
00617     }
00618
00619     return QSPI_ERROR;
00620   }
00621
00622   return QSPI_OK;
00623 }
00624
00625 /**
00626   * @brief  This function resumes a paused e
rase command.
00627   * @retval QSPI memory status
00628   */
00629 uint8_t BSP_QSPI_ResumeErase(void)
00630 {
00631   QSPI_CommandTypeDef sCommand;
00632
00633   /* Check whether the device is in suspende
d state */
00634   if (BSP_QSPI_GetStatus() == QSPI_SUSPENDED
)
00635   {
00636     /* Initialize the erase command */
00637     sCommand.InstructionMode   = QSPI_INSTRU
CTION_1_LINE;
00638     sCommand.Instruction       = PROG_ERASE_
RESUME_CMD;
00639     sCommand.AddressMode       = QSPI_ADDRES
S_NONE;
00640     sCommand.AlternateByteMode = QSPI_ALTERN
ATE_BYTES_NONE;
```

```
00641     sCommand.DataMode          = QSPI_DATA_N
ONE;
00642     sCommand.DummyCycles       = 0;
00643     sCommand.DdrMode           = QSPI_DDR_MO
DE_DISABLE;
00644     sCommand.DdrHoldHalfCycle  = QSPI_DDR_HH
C_ANALOG_DELAY;
00645     sCommand.SIOOMode          = QSPI_SIOO_I
NST_EVERY_CMD;
00646
00647     /* Send the command */
00648     if (HAL_QSPI_Command(&QSPIHandle, &sComm
and, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00649     {
00650       return QSPI_ERROR;
00651     }
00652
00653     /*
00654     When this command is executed, the statu
s register write in progress bit is set to 1, and
00655     the flag status register program erase c
ontroller bit is set to 0. This command is ignored
00656     if the device is not in a suspended stat
e.
00657     */
00658
00659     if (BSP_QSPI_GetStatus() == QSPI_BUSY)
00660     {
00661       return QSPI_OK;
00662     }
00663
00664     return QSPI_ERROR;
00665   }
00666
00667   return QSPI_OK;
00668 }
00669
```

```
00670 /**
00671   * @brief  This function enter the QSPI mem
ory in deep power down mode.
00672   * @retval QSPI memory status
00673   */
00674 uint8_t BSP_QSPI_EnterDeepPowerDown(void)
00675 {
00676   QSPI_CommandTypeDef sCommand;
00677
00678   /* Initialize the deep power down command
*/
00679   sCommand.InstructionMode   = QSPI_INSTRUCT
ION_1_LINE;
00680   sCommand.Instruction       = DEEP_POWER_DO
WN_CMD;
00681   sCommand.AddressMode       = QSPI_ADDRESS_
NONE;
00682   sCommand.AlternateByteMode = QSPI_ALTERNAT
E_BYTES_NONE;
00683   sCommand.DataMode          = QSPI_DATA_NON
E;
00684   sCommand.DummyCycles       = 0;
00685   sCommand.DdrMode           = QSPI_DDR_MODE
_DISABLE;
00686   sCommand.DdrHoldHalfCycle  = QSPI_DDR_HHC_
ANALOG_DELAY;
00687   sCommand.SIOOMode          = QSPI_SIOO_INS
T_EVERY_CMD;
00688
00689   /* Send the command */
00690   if (HAL_QSPI_Command(&QSPIHandle, &Comman
d, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00691   {
00692     return QSPI_ERROR;
00693   }
00694
00695   /* ---          Memory takes 10us max to e
```

```
nter deep power down           --- */
00696   /* --- At least 30us should be respected b
efore leaving deep power down --- */
00697
00698   return QSPI_OK;
00699 }
00700
00701 /**
00702   * @brief  This function leave the QSPI mem
ory from deep power down mode.
00703   * @retval QSPI memory status
00704   */
00705 uint8_t BSP_QSPI_LeaveDeepPowerDown(void)
00706 {
00707   QSPI_CommandTypeDef sCommand;
00708
00709   /* Initialize the erase command */
00710   sCommand.InstructionMode   = QSPI_INSTRUCT
ION_1_LINE;
00711   sCommand.Instruction       = NO_OPERATION_
CMD;
00712   sCommand.AddressMode       = QSPI_ADDRESS_
NONE;
00713   sCommand.AlternateByteMode = QSPI_ALTERNAT
E_BYTES_NONE;
00714   sCommand.DataMode          = QSPI_DATA_NON
E;
00715   sCommand.DummyCycles       = 0;
00716   sCommand.DdrMode           = QSPI_DDR_MODE
_DISABLE;
00717   sCommand.DdrHoldHalfCycle  = QSPI_DDR_HHC_
ANALOG_DELAY;
00718   sCommand.SIOOMode          = QSPI_SIOO_INS
T_EVERY_CMD;
00719
00720   /* Send the command */
00721   if (HAL_QSPI_Command(&QSPIHandle, &Comman
```

```
d, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00722   {
00723     return QSPI_ERROR;
00724   }
00725
00726   /* --- A NOP command is sent to the memory
, as the nCS should be low for at least 20 ns ---
*/
00727   /* ---                    Memory takes 35us
min to leave deep power down                    ---
*/
00728
00729   return QSPI_OK;
00730 }
00731
00732 /**
00733   * @brief  Initializes the QSPI MSP.
00734   * @retval None
00735   */
00736 __weak void BSP_QSPI_MspInit(void)
00737 {
00738   GPIO_InitTypeDef GPIO_InitStruct;
00739
00740   /* Enable the QuadSPI memory interface clo
ck */
00741   __HAL_RCC_QSPI_CLK_ENABLE();
00742
00743   /* Reset the QuadSPI memory interface */
00744   __HAL_RCC_QSPI_FORCE_RESET();
00745   __HAL_RCC_QSPI_RELEASE_RESET();
00746
00747   /* Enable GPIO clocks */
00748   __HAL_RCC_GPIOE_CLK_ENABLE();
00749
00750   /* QSPI CLK, CS, D0, D1, D2 and D3 GPIO pi
ns configuration  */
00751   GPIO_InitStruct.Pin       = GPIO_PIN_10 |
```

```
GPIO_PIN_11 | GPIO_PIN_12 |\
00752                                 GPIO_PIN_13 |
GPIO_PIN_14 | GPIO_PIN_15;
00753   GPIO_InitStruct.Mode      = GPIO_MODE_AF_P
P;
00754   GPIO_InitStruct.Pull      = GPIO_NOPULL;
00755   GPIO_InitStruct.Speed     = GPIO_SPEED_FRE
Q_VERY_HIGH;
00756   GPIO_InitStruct.Alternate = GPIO_AF10_QUAD
SPI;
00757   HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);
00758 }
00759
00760 /**
00761   * @brief  De-Initializes the QSPI MSP.
00762   * @retval None
00763   */
00764 __weak void BSP_QSPI_MspDeInit(void)
00765 {
00766   GPIO_InitTypeDef GPIO_InitStruct;
00767
00768   /* QSPI CLK, CS, D0-D3 GPIO pins de-config
uration  */
00769   __HAL_RCC_GPIOE_CLK_ENABLE();
00770   GPIO_InitStruct.Pin       = GPIO_PIN_10 |
GPIO_PIN_11 | GPIO_PIN_12 |\
00771                               GPIO_PIN_13 |
GPIO_PIN_14 | GPIO_PIN_15;
00772
00773   HAL_GPIO_DeInit(GPIOE, GPIO_InitStruct.Pin
);
00774
00775   /* Reset the QuadSPI memory interface */
00776   __HAL_RCC_QSPI_FORCE_RESET();
00777   __HAL_RCC_QSPI_RELEASE_RESET();
00778
00779   /* Disable the QuadSPI memory interface cl
```

```
ock */
00780   __HAL_RCC_QSPI_CLK_DISABLE();
00781 }
00782
00783 /**
00784   * @}
00785   */
00786
00787 /** @addtogroup STM32L475E_IOT01_QSPI_Privat
e_Functions
00788   * @{
00789   */
00790
00791 /**
00792   * @brief  This function reset the QSPI mem
ory.
00793   * @param  hqspi : QSPI handle
00794   * @retval None
00795   */
00796 static uint8_t QSPI_ResetMemory(QSPI_HandleT
ypeDef *hqspi)
00797 {
00798   QSPI_CommandTypeDef sCommand;
00799
00800   /* Initialize the reset enable command */
00801   sCommand.InstructionMode   = QSPI_INSTRUCT
ION_1_LINE;
00802   sCommand.Instruction       = RESET_ENABLE_
CMD;
00803   sCommand.AddressMode       = QSPI_ADDRESS_
NONE;
00804   sCommand.AlternateByteMode = QSPI_ALTERNAT
E_BYTES_NONE;
00805   sCommand.DataMode          = QSPI_DATA_NON
E;
00806   sCommand.DummyCycles       = 0;
00807   sCommand.DdrMode           = QSPI_DDR_MODE
```

```
_DISABLE;
00808   sCommand.DdrHoldHalfCycle  = QSPI_DDR_HHC_
ANALOG_DELAY;
00809   sCommand.SIOOMode          = QSPI_SIOO_INS
T_EVERY_CMD;
00810
00811   /* Send the command */
00812   if (HAL_QSPI_Command(&QSPIHandle, &sComman
d, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00813   {
00814     return QSPI_ERROR;
00815   }
00816
00817   /* Send the reset memory command */
00818   sCommand.Instruction = RESET_MEMORY_CMD;
00819   if (HAL_QSPI_Command(&QSPIHandle, &sComman
d, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00820   {
00821     return QSPI_ERROR;
00822   }
00823
00824   /* Configure automatic polling mode to wai
t the memory is ready */
00825   if (QSPI_AutoPollingMemReady(&QSPIHandle,
HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != QSPI_OK)
00826   {
00827     return QSPI_ERROR;
00828   }
00829
00830   return QSPI_OK;
00831 }
00832
00833 /**
00834   * @brief  This function send a Write Enabl
e and wait it is effective.
00835   * @param  hqspi : QSPI handle
00836   * @retval None
```

```
00837   */
00838 static uint8_t QSPI_WriteEnable(QSPI_HandleT
ypeDef *hqspi)
00839 {
00840   QSPI_CommandTypeDef     sCommand;
00841   QSPI_AutoPollingTypeDef sConfig;
00842
00843   /* Enable write operations */
00844   sCommand.InstructionMode   = QSPI_INSTRUCT
ION_1_LINE;
00845   sCommand.Instruction       = WRITE_ENABLE_
CMD;
00846   sCommand.AddressMode       = QSPI_ADDRESS_
NONE;
00847   sCommand.AlternateByteMode = QSPI_ALTERNAT
E_BYTES_NONE;
00848   sCommand.DataMode          = QSPI_DATA_NON
E;
00849   sCommand.DummyCycles       = 0;
00850   sCommand.DdrMode           = QSPI_DDR_MODE
_DISABLE;
00851   sCommand.DdrHoldHalfCycle  = QSPI_DDR_HHC_
ANALOG_DELAY;
00852   sCommand.SIOOMode          = QSPI_SIOO_INS
T_EVERY_CMD;
00853
00854   if (HAL_QSPI_Command(&QSPIHandle, &sComman
d, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00855   {
00856     return QSPI_ERROR;
00857   }
00858
00859   /* Configure automatic polling mode to wai
t for write enabling */
00860   sConfig.Match           = MX25R6435F_SR_WE
L;
00861   sConfig.Mask            = MX25R6435F_SR_WE
```

```
L;
00862   sConfig.MatchMode       = QSPI_MATCH_MODE_
AND;
00863   sConfig.StatusBytesSize = 1;
00864   sConfig.Interval        = 0x10;
00865   sConfig.AutomaticStop   = QSPI_AUTOMATIC_S
TOP_ENABLE;
00866
00867   sCommand.Instruction    = READ_STATUS_REG_
CMD;
00868   sCommand.DataMode       = QSPI_DATA_1_LINE
;
00869
00870   if (HAL_QSPI_AutoPolling(&QSPIHandle, &sCo
mmand, &sConfig, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) !
= HAL_OK)
00871   {
00872     return QSPI_ERROR;
00873   }
00874
00875   return QSPI_OK;
00876 }
00877
00878 /**
00879   * @brief  This function read the SR of the
 memory and wait the EOP.
00880   * @param  hqspi   : QSPI handle
00881   * @param  Timeout : Timeout for auto-polli
ng
00882   * @retval None
00883   */
00884 static uint8_t QSPI_AutoPollingMemReady(QSPI
_HandleTypeDef *hqspi, uint32_t Timeout)
00885 {
00886   QSPI_CommandTypeDef     sCommand;
00887   QSPI_AutoPollingTypeDef sConfig;
00888
```

```
00889   /* Configure automatic polling mode to wai
t for memory ready */
00890   sCommand.InstructionMode   = QSPI_INSTRUCT
ION_1_LINE;
00891   sCommand.Instruction       = READ_STATUS_R
EG_CMD;
00892   sCommand.AddressMode       = QSPI_ADDRESS_
NONE;
00893   sCommand.AlternateByteMode = QSPI_ALTERNAT
E_BYTES_NONE;
00894   sCommand.DataMode          = QSPI_DATA_1_L
INE;
00895   sCommand.DummyCycles       = 0;
00896   sCommand.DdrMode           = QSPI_DDR_MODE
_DISABLE;
00897   sCommand.DdrHoldHalfCycle  = QSPI_DDR_HHC_
ANALOG_DELAY;
00898   sCommand.SIOOMode          = QSPI_SIOO_INS
T_EVERY_CMD;
00899
00900   sConfig.Match           = 0;
00901   sConfig.Mask            = MX25R6435F_SR_WI
P;
00902   sConfig.MatchMode       = QSPI_MATCH_MODE_
AND;
00903   sConfig.StatusBytesSize = 1;
00904   sConfig.Interval        = 0x10;
00905   sConfig.AutomaticStop   = QSPI_AUTOMATIC_S
TOP_ENABLE;
00906
00907   if (HAL_QSPI_AutoPolling(&QSPIHandle, &sCo
mmand, &sConfig, Timeout) != HAL_OK)
00908   {
00909     return QSPI_ERROR;
00910   }
00911
00912   return QSPI_OK;
```

```
00913 }
00914
00915 /**
00916   * @brief  This function enables/disables t
he Quad mode of the memory.
00917   * @param  hqspi     : QSPI handle
00918   * @param  Operation : QSPI_QUAD_ENABLE or
QSPI_QUAD_DISABLE mode
00919   * @retval None
00920   */
00921 static uint8_t QSPI_QuadMode(QSPI_HandleType
Def *hqspi, uint8_t Operation)
00922 {
00923   QSPI_CommandTypeDef sCommand;
00924   uint8_t reg;
00925
00926   /* Read status register */
00927   sCommand.InstructionMode   = QSPI_INSTRUCT
ION_1_LINE;
00928   sCommand.Instruction       = READ_STATUS_R
EG_CMD;
00929   sCommand.AddressMode       = QSPI_ADDRESS_
NONE;
00930   sCommand.AlternateByteMode = QSPI_ALTERNAT
E_BYTES_NONE;
00931   sCommand.DataMode          = QSPI_DATA_1_L
INE;
00932   sCommand.DummyCycles       = 0;
00933   sCommand.NbData            = 1;
00934   sCommand.DdrMode           = QSPI_DDR_MODE
_DISABLE;
00935   sCommand.DdrHoldHalfCycle  = QSPI_DDR_HHC_
ANALOG_DELAY;
00936   sCommand.SIOOMode          = QSPI_SIOO_INS
T_EVERY_CMD;
00937
00938   if (HAL_QSPI_Command(&QSPIHandle, &sComman
```

```
d, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00939   {
00940     return QSPI_ERROR;
00941   }
00942
00943   if (HAL_QSPI_Receive(&QSPIHandle, &reg, HA
L_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00944   {
00945     return QSPI_ERROR;
00946   }
00947
00948   /* Enable write operations */
00949   if (QSPI_WriteEnable(&QSPIHandle) != QSPI_
OK)
00950   {
00951     return QSPI_ERROR;
00952   }
00953
00954   /* Activate/deactivate the Quad mode */
00955   if (Operation == QSPI_QUAD_ENABLE)
00956   {
00957     SET_BIT(reg, MX25R6435F_SR_QE);
00958   }
00959   else
00960   {
00961     CLEAR_BIT(reg, MX25R6435F_SR_QE);
00962   }
00963
00964   sCommand.Instruction = WRITE_STATUS_CFG_RE
G_CMD;
00965
00966   if (HAL_QSPI_Command(&QSPIHandle, &sComman
d, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00967   {
00968     return QSPI_ERROR;
00969   }
00970
```

```
00971   if (HAL_QSPI_Transmit(&QSPIHandle, &reg, H
AL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00972   {
00973     return QSPI_ERROR;
00974   }
00975
00976   /* Wait that memory is ready */
00977   if (QSPI_AutoPollingMemReady(&QSPIHandle,
HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != QSPI_OK)
00978   {
00979     return QSPI_ERROR;
00980   }
00981
00982   /* Check the configuration has been correc
tly done */
00983   sCommand.Instruction = READ_STATUS_REG_CMD
;
00984
00985   if (HAL_QSPI_Command(&QSPIHandle, &sComman
d, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00986   {
00987     return QSPI_ERROR;
00988   }
00989
00990   if (HAL_QSPI_Receive(&QSPIHandle, &reg, HA
L_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00991   {
00992     return QSPI_ERROR;
00993   }
00994
00995   if ((((reg & MX25R6435F_SR_QE) == 0) && (O
peration == QSPI_QUAD_ENABLE)) ||
00996       (((reg & MX25R6435F_SR_QE) != 0) && (O
peration == QSPI_QUAD_DISABLE)))
00997   {
00998     return QSPI_ERROR;
00999   }
```

```
01000
01001   return QSPI_OK;
01002 }
01003
01004 /**
01005   * @brief  This function enables/disables t
he high performance mode of the memory.
01006   * @param  hqspi     : QSPI handle
01007   * @param  Operation : QSPI_HIGH_PERF_ENABL
E or QSPI_HIGH_PERF_DISABLE high performance mode

01008   * @retval None
01009   */
01010 static uint8_t QSPI_HighPerfMode(QSPI_Handle
TypeDef *hqspi, uint8_t Operation)
01011 {
01012   QSPI_CommandTypeDef sCommand;
01013   uint8_t reg[3];
01014
01015   /* Read status register */
01016   sCommand.InstructionMode   = QSPI_INSTRUCT
ION_1_LINE;
01017   sCommand.Instruction       = READ_STATUS_R
EG_CMD;
01018   sCommand.AddressMode       = QSPI_ADDRESS_
NONE;
01019   sCommand.AlternateByteMode = QSPI_ALTERNAT
E_BYTES_NONE;
01020   sCommand.DataMode          = QSPI_DATA_1_L
INE;
01021   sCommand.DummyCycles       = 0;
01022   sCommand.NbData            = 1;
01023   sCommand.DdrMode           = QSPI_DDR_MODE
_DISABLE;
01024   sCommand.DdrHoldHalfCycle  = QSPI_DDR_HHC_
ANALOG_DELAY;
01025   sCommand.SIOOMode          = QSPI_SIOO_INS
```

```
T_EVERY_CMD;
01026
01027   if (HAL_QSPI_Command(&QSPIHandle, &sComman
d, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
01028   {
01029     return QSPI_ERROR;
01030   }
01031
01032   if (HAL_QSPI_Receive(&QSPIHandle, &(reg[0]
), HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
01033   {
01034     return QSPI_ERROR;
01035   }
01036
01037   /* Read configuration registers */
01038   sCommand.Instruction = READ_CFG_REG_CMD;
01039   sCommand.NbData      = 2;
01040
01041   if (HAL_QSPI_Command(&QSPIHandle, &sComman
d, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
01042   {
01043     return QSPI_ERROR;
01044   }
01045
01046   if (HAL_QSPI_Receive(&QSPIHandle, &(reg[1]
), HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
01047   {
01048     return QSPI_ERROR;
01049   }
01050
01051   /* Enable write operations */
01052   if (QSPI_WriteEnable(&QSPIHandle) != QSPI_
OK)
01053   {
01054     return QSPI_ERROR;
01055   }
01056
```

```
01057   /* Activate/deactivate the Quad mode */
01058   if (Operation == QSPI_HIGH_PERF_ENABLE)
01059   {
01060     SET_BIT(reg[2], MX25R6435F_CR2_LH_SWITCH
);
01061   }
01062   else
01063   {
01064     CLEAR_BIT(reg[2], MX25R6435F_CR2_LH_SWIT
CH);
01065   }
01066
01067   sCommand.Instruction = WRITE_STATUS_CFG_RE
G_CMD;
01068   sCommand.NbData      = 3;
01069
01070   if (HAL_QSPI_Command(&QSPIHandle, &sComman
d, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
01071   {
01072     return QSPI_ERROR;
01073   }
01074
01075   if (HAL_QSPI_Transmit(&QSPIHandle, &(reg[0
]), HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
01076   {
01077     return QSPI_ERROR;
01078   }
01079
01080   /* Wait that memory is ready */
01081   if (QSPI_AutoPollingMemReady(&QSPIHandle,
HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != QSPI_OK)
01082   {
01083     return QSPI_ERROR;
01084   }
01085
01086   /* Check the configuration has been correc
tly done */
```

```
01087   sCommand.Instruction = READ_CFG_REG_CMD;
01088   sCommand.NbData      = 2;
01089
01090   if (HAL_QSPI_Command(&QSPIHandle, &sComman
d, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
01091   {
01092     return QSPI_ERROR;
01093   }
01094
01095   if (HAL_QSPI_Receive(&QSPIHandle, &(reg[0]
), HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
01096   {
01097     return QSPI_ERROR;
01098   }
01099
01100   if ((((reg[1] & MX25R6435F_CR2_LH_SWITCH)
== 0) && (Operation == QSPI_HIGH_PERF_ENABLE)) ||
01101       (((reg[1] & MX25R6435F_CR2_LH_SWITCH)
!= 0) && (Operation == QSPI_HIGH_PERF_DISABLE)))
01102   {
01103     return QSPI_ERROR;
01104   }
01105
01106   return QSPI_OK;
01107 }
01108
01109 /**
01110   * @}
01111   */
01112
01113 /**
01114   * @}
01115   */
01116
01117 /**
01118   * @}
01119   */
```

```
01120
01121 /**
01122   * @}
01123   */
01124
01125 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
01126
```

B-L475E-IOT01 BSP User Manual

LOW LEVEL Private Def

LOW LEVEL

Defines

#define	**__STM32L475E_IOT01_BSP_VERSION_MAIN** (0x01) STM32L475E IOT01 BSP Driver version number V1.0.0.
#define	**__STM32L475E_IOT01_BSP_VERSION_SUB1** (0x00)
#define	**__STM32L475E_IOT01_BSP_VERSION_SUB2** (0x00)
#define	**__STM32L475E_IOT01_BSP_VERSION_RC** (0x00)
#define	**__STM32L475E_IOT01_BSP_VERSION**

Define Documentation

#define __STM32L475E_IOT01_BSP_VERSION

Value:

```
((__STM32L475E_IOT01_BSP_VERSION_MAIN << 24)\
|(__STM32L475E_IOT01_BSP_VERSION_SUB1 << 16)\
|(__STM32L475E_IOT01_BSP_VERSION_SUB2 << 8 )\
|(__STM32L475E_IOT01_BSP_VERSION_RC))
```

Definition at line **72** of file **stm32l475e_iot01.c**.

Referenced by **BSP_GetVersion()**.

#define __STM32L475E_IOT01_BSP_VERSION_MAIN (0x01)

STM32L475E IOT01 BSP Driver version number V1.0.0.

[31:24] main version

Definition at line **68** of file **stm32l475e_iot01.c**.

#define __STM32L475E_IOT01_BSP_VERSION_RC (0x00)

[7:0] release candidate

Definition at line **71** of file **stm32l475e_iot01.c**.

#define __STM32L475E_IOT01_BSP_VERSION_SUB1 (0x00)

[23:16] sub1 version

Definition at line **69** of file **stm32l475e_iot01.c**.

#define __STM32L475E_IOT01_BSP_VERSION_SUB2 (0x00)

[15:8] sub2 version

Definition at line **70** of file **stm32l475e_iot01.c**.

B-L475E-IOT01 BSP User Manual

Enumerations

ACCELERO Exported Types

ACCELERO

Enumerations

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

Enumeration Type Documentation

enum ACCELERO_StatusTypeDef

Enumerator:

ACCELERO_OK

ACCELERO_ERROR

ACCELERO_TIMEOUT

Definition at line **76** of file **stm32l475e_iot01_accelero.h**.

B-L475E-IOT01 BSP User Manual

Variables

ACCELERO Private Variables

ACCELERO

Variables

static ACCELERO_DrvTypeDef *	**AccelerometerDrv**

Variable Documentation

ACCELERO_DrvTypeDef* AccelerometerDrv [static]

Definition at line **65** of file **stm32l475e_iot01_accelero.c**.

Referenced by **BSP_ACCELERO_AccGetXYZ()**, **BSP_ACCELERO_DeInit()**, **BSP_ACCELERO_Init()**, and **BSP_ACCELERO_LowPower()**.

B-L475E-IOT01 BSP User Manual

ACCELERO Private Functions

ACCELERO

Functions

ACCELERO_StatusTypeDef	**BSP_ACCELERO_Init** (void) Initialize the ACCELERO.
void	**BSP_ACCELERO_DeInit** (void) DeInitialize the ACCELERO.
void	**BSP_ACCELERO_LowPower** (uint16_t status) Set/Unset the ACCELERO in low power mode.
void	**BSP_ACCELERO_AccGetXYZ** (int16_t *pDataXYZ) Get XYZ acceleration values.

Function Documentation

void BSP_ACCELERO_AccGetXYZ (int16_t * pDataXYZ)

Get XYZ acceleration values.

Parameters:

pDataXYZ	Pointer on 3 angular accelerations table with pDataXYZ[0] = X axis, pDataXYZ[1] = Y axis, pDataXYZ[2] = Z axis

Return values:

None

Definition at line **153** of file **stm32l475e_iot01_accelero.c**.

References **AccelerometerDrv**.

void BSP_ACCELERO_DeInit (void)

DeInitialize the ACCELERO.

Return values:

None.

Definition at line **118** of file **stm32l475e_iot01_accelero.c**.

References **AccelerometerDrv**.

ACCELERO_StatusTypeDef BSP_ACCELERO_Init (void)

Initialize the ACCELERO.

Return values:

ACCELERO_OK or ACCELERO_ERROR

Definition at line **77** of file **stm32l475e_iot01_accelero.c**.

References **ACCELERO_ERROR**, **ACCELERO_OK**, and **AccelerometerDrv**.

void BSP_ACCELERO_LowPower (uint16_t status)

Set/Unset the ACCELERO in low power mode.

Parameters:

status 0 means disable Low Power Mode, otherwise Low Power Mode is enabled

Return values:

None

Definition at line **135** of file **stm32l475e_iot01_accelero.c**.

References **AccelerometerDrv**.

ACCELERO Exported Functions

ACCELERO

Functions

ACCELERO_StatusTypeDef	**BSP_ACCELERO_Init** (void) Initialize the ACCELERO.
void	**BSP_ACCELERO_DeInit** (void) DeInitialize the ACCELERO.
void	**BSP_ACCELERO_LowPower** (uint16_t status) Set/Unset the ACCELERO in low power mode.
void	**BSP_ACCELERO_AccGetXYZ** (int16_t *pDataXYZ) Get XYZ acceleration values.

Function Documentation

void BSP_ACCELERO_AccGetXYZ (int16_t * pDataXYZ)

Get XYZ acceleration values.

Parameters:

pDataXYZ	Pointer on 3 angular accelerations table with pDataXYZ[0] = X axis, pDataXYZ[1] = Y axis, pDataXYZ[2] = Z axis

Return values:

None

Definition at line **153** of file **stm32l475e_iot01_accelero.c**.

References **AccelerometerDrv**.

void BSP_ACCELERO_DeInit (void)

DeInitialize the ACCELERO.

Return values:

None.

Definition at line **118** of file **stm32l475e_iot01_accelero.c**.

References **AccelerometerDrv**.

ACCELERO_StatusTypeDef BSP_ACCELERO_Init (void)

Initialize the ACCELERO.

Return values:

ACCELERO_OK or ACCELERO_ERROR

Definition at line **77** of file **stm32l475e_iot01_accelero.c**.

References **ACCELERO_ERROR**, **ACCELERO_OK**, and **AccelerometerDrv**.

void BSP_ACCELERO_LowPower (uint16_t status)

Set/Unset the ACCELERO in low power mode.

Parameters:

status 0 means disable Low Power Mode, otherwise Low Power Mode is enabled

Return values:

None

Definition at line **135** of file **stm32l475e_iot01_accelero.c**.

References **AccelerometerDrv**.

B-L475E-IOT01 BSP User Manual

LOW LEVEL Private Functions

LOW LEVEL

Functions

uint32_t	**BSP_GetVersion** (void) This method returns the STM32L475E IOT01 BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LEDs.
void	**BSP_LED_DeInit** (**Led_TypeDef** Led) DeInit LEDs.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** ButtonMode) Configures button GPIO and EXTI Line.
void	**BSP_PB_DeInit** (**Button_TypeDef** Button) Push Button DeInit.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) Configures COM port.
void	**BSP_COM_DeInit** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) DeInit COM port.
static void	**I2Cx_MspInit** (I2C_HandleTypeDef *i2c_handler) Initializes I2C MSP.

static void	**I2Cx_MspDeInit** (I2C_HandleTypeDef *i2c_handler) DeInitializes I2C MSP.
static void	**I2Cx_Init** (I2C_HandleTypeDef *i2c_handler) Initializes I2C HAL.
static void	**I2Cx_DeInit** (I2C_HandleTypeDef *i2c_handler) DeInitializes I2C HAL.
static HAL_StatusTypeDef	**I2Cx_ReadMultiple** (I2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t *Buffer, uint16_t Length) Reads multiple data.
static HAL_StatusTypeDef	**I2Cx_WriteMultiple** (I2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t *Buffer, uint16_t Length) Writes a value in a register of the device through BUS in using DMA mode.
static HAL_StatusTypeDef	**I2Cx_IsDeviceReady** (I2C_HandleTypeDef *i2c_handler, uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.
static void	**I2Cx_Error** (I2C_HandleTypeDef *i2c_handler, uint8_t Addr) Manages error callback by re-initializing I2C.

Function Documentation

void BSP_COM_DeInit (COM_TypeDef COM,
UART_HandleTypeDef * huart
)

DeInit COM port.

Parameters:

COM,: COM port to be configured. This parameter can be one of the following values:
- COM1

huart,: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified USART peripheral.

Definition at line **345** of file **stm32l475e_iot01.c**.

References **COM_USART**, and **DISCOVERY_COMx_CLK_DISABLE**.

void BSP_COM_Init (COM_TypeDef COM,
UART_HandleTypeDef * huart
)

Configures COM port.

Parameters:

COM,: COM port to be configured. This parameter can be one of the following values:
- COM1

huart,: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified USART peripheral.

Definition at line 307 of file stm32l475e_iot01.c.

References COM_RX_AF, COM_RX_PIN, COM_RX_PORT, COM_TX_AF, COM_TX_PIN, COM_TX_PORT, COM_USART, DISCOVERY_COMx_CLK_ENABLE, DISCOVERY_COMx_RX_GPIO_CLK_ENABLE, and DISCOVERY_COMx_TX_GPIO_CLK_ENABLE.

uint32_t BSP_GetVersion (void)

This method returns the STM32L475E IOT01 BSP Driver revision.

Return values:

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

Definition at line 149 of file stm32l475e_iot01.c.

References __STM32L475E_IOT01_BSP_VERSION.

void BSP_LED_DeInit (Led_TypeDef Led)

DeInit LEDs.

Parameters:

Led,: LED to be configured. This parameter can be one of the following values:

- LED2

Definition at line 180 of file stm32l475e_iot01.c.

References GPIO_PIN, and GPIO_PORT.

void BSP_LED_Init (Led_TypeDef Led)

Configures LEDs.

Parameters:

Led,: LED to be configured. This parameter can be one of the following values:

- LED2

Definition at line **160** of file **stm32l475e_iot01.c**.

References **GPIO_PIN**, **GPIO_PORT**, and **LEDx_GPIO_CLK_ENABLE**.

void BSP_LED_Off (Led_TypeDef Led)

Turns selected LED Off.

Parameters:

Led,: LED to be set off This parameter can be one of the following values:

- LED2

Definition at line **209** of file **stm32l475e_iot01.c**.

References **GPIO_PIN**, and **GPIO_PORT**.

void BSP_LED_On (Led_TypeDef Led)

Turns selected LED On.

Parameters:

Led,: LED to be set on This parameter can be one of the following values:

- LED2

Definition at line **198** of file **stm32l475e_iot01.c**.

References **GPIO_PIN**, and **GPIO_PORT**.

void BSP_LED_Toggle (Led_TypeDef Led)

Toggles the selected LED.

Parameters:

Led,: LED to be toggled This parameter can be one of the following values:

- LED2

Definition at line **220** of file **stm32l475e_iot01.c**.

References **GPIO_PIN**, and **GPIO_PORT**.

void BSP_PB_DeInit (Button_TypeDef Button)

Push Button DeInit.

Parameters:

Button,: Button to be configured This parameter can be one of the following values:

- BUTTON_WAKEUP: Wakeup Push Button

Note:

PB DeInit does not disable the GPIO clock

Definition at line **277** of file **stm32l475e_iot01.c**.

References **BUTTON_IRQn**, **BUTTON_PIN**, and **BUTTON_PORT**.

uint32_t BSP_PB_GetState (Button_TypeDef Button)

Returns the selected button state.

Parameters:

Button,: Button to be checked This parameter can be one of the following values:

- BUTTON_WAKEUP: Wakeup Push Button

Return values:

The Button GPIO pin value (GPIO_PIN_RESET = button pressed)

Definition at line **294** of file **stm32l475e_iot01.c**.

References **BUTTON_PIN**, and **BUTTON_PORT**.

void BSP_PB_Init (Button_TypeDef Button,
ButtonMode_TypeDef ButtonMode
)

Configures button GPIO and EXTI Line.

Parameters:

Button,: Button to be configured This parameter can be one of the following values:

- BUTTON_WAKEUP: Wakeup Push Button

ButtonMode,: Button mode This parameter can be one of the following values:

- BUTTON_MODE_GPIO: Button will be used as simple IO
- BUTTON_MODE_EXTI: Button will be connected to EXTI line with interrupt generation capability

Definition at line **236** of file **stm32l475e_iot01.c**.

References **BUTTON_IRQn**, **BUTTON_MODE_EXTI**, **BUTTON_MODE_GPIO**, **BUTTON_PIN**, **BUTTON_PORT**, and

USER_BUTTON_GPIO_CLK_ENABLE.

static void I2Cx_DeInit (I2C_HandleTypeDef * i2c_handler) [static

DeInitializes I2C HAL.

Parameters:

i2c_handler : I2C handler

Return values:

None

Definition at line **458** of file **stm32l475e_iot01.c**.

References **I2Cx_MspDeInit()**.

Referenced by **SENSOR_IO_DeInit()**.

static void I2Cx_Error (I2C_HandleTypeDef * i2c_handler, uint8_t Addr) [static]

Manages error callback by re-initializing I2C.

Parameters:

i2c_handler : I2C handler

Addr,: I2C Address

Return values:

None

Definition at line **534** of file **stm32l475e_iot01.c**.

References **I2Cx_Init()**.

Referenced by **I2Cx_ReadMultiple()**, and **I2Cx_WriteMultiple()**.

static void I2Cx_Init (I2C_HandleTypeDef * i2c_handler) [static]

Initializes I2C HAL.

Parameters:

i2c_handler : I2C handler

Return values:

None

Configure Analogue filter

Definition at line **433** of file **stm32l475e_iot01.c**.

References **DISCOVERY_I2Cx**, and **I2Cx_MspInit()**.

Referenced by **I2Cx_Error()**, and **SENSOR_IO_Init()**.

static HAL_StatusTypeDef I2Cx_IsDeviceReady (I2C_HandleTypeD
uint16_t
uint32_t
)

Checks if target device is ready for communication.

Note:

This function is used with Memory devices

Parameters:

i2c_handler : I2C handler
DevAddress,: Target device address
Trials,: Number of trials

Return values:

HAL status

Definition at line **523** of file **stm32l475e_iot01.c**.

Referenced by **SENSOR_IO_IsDeviceReady()**.

static void I2Cx_MspDeInit (I2C_HandleTypeDef * i2c_handler) [st

DeInitializes I2C MSP.

Parameters:

i2c_handler : I2C handler

Return values:

None

Definition at line **414** of file **stm32l475e_iot01.c**.

References **DISCOVERY_I2Cx_CLK_DISABLE**, **DISCOVERY_I2Cx_SCL_PIN**, **DISCOVERY_I2Cx_SCL_SDA_GPIO_CLK_DISABLE**, **DISCOVERY_I2Cx_SCL_SDA_GPIO_PORT**, and **DISCOVERY_I2Cx_SDA_PIN**.

Referenced by **I2Cx_DeInit()**.

static void I2Cx_MspInit (I2C_HandleTypeDef * i2c_handler) [stat

Initializes I2C MSP.

Parameters:

i2c_handler : I2C handler

Return values:

None

Definition at line **372** of file **stm32l475e_iot01.c**.

References **DISCOVERY_I2Cx_CLK_ENABLE**, **DISCOVERY_I2Cx_ER_IRQn**, **DISCOVERY_I2Cx_EV_IRQn**, **DISCOVERY_I2Cx_FORCE_RESET**, **DISCOVERY_I2Cx_RELEASE_RESET**, **DISCOVERY_I2Cx_SCL_PIN**, **DISCOVERY_I2Cx_SCL_SDA_AF**, **DISCOVERY_I2Cx_SCL_SDA_GPIO_CLK_ENABLE**, **DISCOVERY_I2Cx_SCL_SDA_GPIO_PORT**, and **DISCOVERY_I2Cx_SDA_PIN**.

Referenced by **I2Cx_Init()**.

static HAL_StatusTypeDef I2Cx_ReadMultiple (I2C_HandleTypeDef
uint8_t
uint16_t
uint16_t
uint8_t *
uint16_t
)

Reads multiple data.

Parameters:

i2c_handler	: I2C handler
Addr,:	I2C address
Reg,:	Reg address
MemAddress,:	memory address
Buffer,:	Pointer to data buffer
Length,:	Length of the data

Return values:

HAL status

Definition at line **474** of file **stm32l475e_iot01.c**.

References **I2Cx_Error()**.

Referenced by **SENSOR_IO_Read()**, and **SENSOR_IO_ReadMultiple()**.

static HAL_StatusTypeDef I2Cx_WriteMultiple (I2C_HandleTypeDef *
uint8_t
uint16_t
uint16_t
uint8_t *
uint16_t
)

Writes a value in a register of the device through BUS in using DMA mode.

Parameters:

i2c_handler	: I2C handler
Addr,:	Device address on BUS Bus.
Reg,:	The target register address to write
MemAddress,:	memory address
Buffer,:	The target register value to be written
Length,:	buffer size to be written

Return values:

HAL status

Definition at line **500** of file **stm32l475e_iot01.c**.

References **I2Cx_Error()**.

Referenced by **SENSOR_IO_Write()**, and **SENSOR_IO_WriteMultiple()**.

B-L475E-IOT01 BSP User Manual

Functions

LOW LEVEL Exported Functions

LOW LEVEL

Functions

uint32_t	**BSP_GetVersion** (void) This method returns the STM32L475E IOT01 BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LEDs.
void	**BSP_LED_DeInit** (**Led_TypeDef** Led) DeInit LEDs.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** ButtonMode) Configures button GPIO and EXTI Line.
void	**BSP_PB_DeInit** (**Button_TypeDef** Button) Push Button DeInit.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *husart) Configures COM port.
void	**BSP_COM_DeInit** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) DeInit COM port.

Function Documentation

void BSP_COM_DeInit (COM_TypeDef COM,
UART_HandleTypeDef * huart
)

DeInit COM port.

Parameters:

COM,: COM port to be configured. This parameter can be one of the following values:
- COM1

huart,: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified USART peripheral.

Definition at line **345** of file **stm32l475e_iot01.c**.

References **COM_USART**, and **DISCOVERY_COMx_CLK_DISABLE**.

void BSP_COM_Init (COM_TypeDef COM,
UART_HandleTypeDef * huart
)

Configures COM port.

Parameters:

COM,: COM port to be configured. This parameter can be one of the following values:
- COM1

huart,: Pointer to a UART_HandleTypeDef structure that contains the configuration information for the specified USART peripheral.

Definition at line 307 of file stm32l475e_iot01.c.

References COM_RX_AF, COM_RX_PIN, COM_RX_PORT, COM_TX_AF, COM_TX_PIN, COM_TX_PORT, COM_USART, DISCOVERY_COMx_CLK_ENABLE, DISCOVERY_COMx_RX_GPIO_CLK_ENABLE, and DISCOVERY_COMx_TX_GPIO_CLK_ENABLE.

uint32_t BSP_GetVersion (void)

This method returns the STM32L475E IOT01 BSP Driver revision.

Return values:

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

Definition at line 149 of file stm32l475e_iot01.c.

References __STM32L475E_IOT01_BSP_VERSION.

void BSP_LED_DeInit (Led_TypeDef Led)

DeInit LEDs.

Parameters:

Led,: LED to be configured. This parameter can be one of the following values:

- LED2

Definition at line 180 of file stm32l475e_iot01.c.

References GPIO_PIN, and GPIO_PORT.

void BSP_LED_Init (Led_TypeDef Led)

Configures LEDs.

Parameters:

Led,: LED to be configured. This parameter can be one of the following values:

- LED2

Definition at line **160** of file **stm32l475e_iot01.c**.

References **GPIO_PIN**, **GPIO_PORT**, and **LEDx_GPIO_CLK_ENABLE**.

void BSP_LED_Off (Led_TypeDef Led)

Turns selected LED Off.

Parameters:

Led,: LED to be set off This parameter can be one of the following values:

- LED2

Definition at line **209** of file **stm32l475e_iot01.c**.

References **GPIO_PIN**, and **GPIO_PORT**.

void BSP_LED_On (Led_TypeDef Led)

Turns selected LED On.

Parameters:

Led,: LED to be set on This parameter can be one of the following values:

- LED2

Definition at line **198** of file **stm32l475e_iot01.c**.

References **GPIO_PIN**, and **GPIO_PORT**.

void BSP_LED_Toggle (Led_TypeDef Led)

Toggles the selected LED.

Parameters:

Led,: LED to be toggled This parameter can be one of the following values:

- LED2

Definition at line **220** of file **stm32l475e_iot01.c**.

References **GPIO_PIN**, and **GPIO_PORT**.

void BSP_PB_DeInit (Button_TypeDef Button)

Push Button DeInit.

Parameters:

Button,: Button to be configured This parameter can be one of the following values:

- BUTTON_WAKEUP: Wakeup Push Button

Note:

PB DeInit does not disable the GPIO clock

Definition at line **277** of file **stm32l475e_iot01.c**.

References **BUTTON_IRQn**, **BUTTON_PIN**, and **BUTTON_PORT**.

uint32_t BSP_PB_GetState (Button_TypeDef Button)

Returns the selected button state.

Parameters:

Button,: Button to be checked This parameter can be one of the following values:

- BUTTON_WAKEUP: Wakeup Push Button

Return values:

The Button GPIO pin value (GPIO_PIN_RESET = button pressed)

Definition at line **294** of file **stm32l475e_iot01.c**.

References **BUTTON_PIN**, and **BUTTON_PORT**.

void BSP_PB_Init (Button_TypeDef Button,
ButtonMode_TypeDef ButtonMode
)

Configures button GPIO and EXTI Line.

Parameters:

Button,: Button to be configured This parameter can be one of the following values:

- BUTTON_WAKEUP: Wakeup Push Button

ButtonMode,: Button mode This parameter can be one of the following values:

- BUTTON_MODE_GPIO: Button will be used as simple IO
- BUTTON_MODE_EXTI: Button will be connected to EXTI line with interrupt generation capability

Definition at line **236** of file **stm32l475e_iot01.c**.

References **BUTTON_IRQn**, **BUTTON_MODE_EXTI**, **BUTTON_MODE_GPIO**, **BUTTON_PIN**, **BUTTON_PORT**, and

USER_BUTTON_GPIO_CLK_ENABLE.

B-L475E-IOT01 BSP User Manual

Functions

GYROSCOPE Exported Functions

GYROSCOPE

Functions

uint8_t	**BSP_GYRO_Init** (void) Initialize Gyroscope.
void	**BSP_GYRO_DeInit** (void) DeInitialize Gyroscope.
void	**BSP_GYRO_LowPower** (uint16_t status) Set/Unset Gyroscope in low power mode.
void	**BSP_GYRO_GetXYZ** (float *pfData) Get XYZ angular acceleration from the Gyroscope.

Function Documentation

void BSP_GYRO_DeInit (void)

DeInitialize Gyroscope.

Definition at line 123 of file stm32l475e_iot01_gyro.c.

References GyroscopeDrv.

void BSP_GYRO_GetXYZ (float * pfData)

Get XYZ angular acceleration from the Gyroscope.

Parameters:

pfData,: pointer on floating array

Definition at line 156 of file stm32l475e_iot01_gyro.c.

References GyroscopeDrv.

uint8_t BSP_GYRO_Init (void)

Initialize Gyroscope.

Return values:

GYRO_OK or GYRO_ERROR

Definition at line 80 of file stm32l475e_iot01_gyro.c.

References GYRO_ERROR, GYRO_OK, and GyroscopeDrv.

void BSP_GYRO_LowPower (uint16_t status)

Set/Unset Gyroscope in low power mode.

Parameters:

status 0 means disable Low Power Mode, otherwise Low Power Mode is enabled

Definition at line **140** of file **stm32l475e_iot01_gyro.c**.

References **GyroscopeDrv**.

B-L475E-IOT01 BSP User Manual

Functions

GYROSCOPE Private Functions

GYROSCOPE

Functions

uint8_t	**BSP_GYRO_Init** (void) Initialize Gyroscope.
void	**BSP_GYRO_DeInit** (void) DeInitialize Gyroscope.
void	**BSP_GYRO_LowPower** (uint16_t status) Set/Unset Gyroscope in low power mode.
void	**BSP_GYRO_GetXYZ** (float *pfData) Get XYZ angular acceleration from the Gyroscope.

Function Documentation

void BSP_GYRO_DeInit (void)

DeInitialize Gyroscope.

Definition at line 123 of file stm32l475e_iot01_gyro.c.

References GyroscopeDrv.

void BSP_GYRO_GetXYZ (float * pfData)

Get XYZ angular acceleration from the Gyroscope.

Parameters:

pfData,: pointer on floating array

Definition at line 156 of file stm32l475e_iot01_gyro.c.

References GyroscopeDrv.

uint8_t BSP_GYRO_Init (void)

Initialize Gyroscope.

Return values:

GYRO_OK or GYRO_ERROR

Definition at line 80 of file stm32l475e_iot01_gyro.c.

References GYRO_ERROR, GYRO_OK, and GyroscopeDrv.

void BSP_GYRO_LowPower (uint16_t status)

Set/Unset Gyroscope in low power mode.

Parameters:

status 0 means disable Low Power Mode, otherwise Low Power Mode is enabled

Definition at line **140** of file **stm32l475e_iot01_gyro.c**.

References **GyroscopeDrv**.

B-L475E-IOT01 BSP User Manual

Functions

HUMIDITY Private Functions

HUMIDITY

Functions

uint32_t	**BSP_HSENSOR_Init** (void) Initializes peripherals used by the I2C Humidity Sensor driver.
uint8_t	**BSP_HSENSOR_ReadID** (void) Read ID of HTS221.
float	**BSP_HSENSOR_ReadHumidity** (void) Read Humidity register of HTS221.

Function Documentation

uint32_t BSP_HSENSOR_Init (void)

Initializes peripherals used by the I2C Humidity Sensor driver.

Return values:

HSENSOR status

Definition at line 79 of file stm32l475e_iot01_hsensor.c.

References Hsensor_drv, HSENSOR_ERROR, HSENSOR_OK, and HTS221_I2C_ADDRESS.

float BSP_HSENSOR_ReadHumidity (void)

Read Humidity register of HTS221.

Return values:

HTS221 measured humidity value.

Definition at line 111 of file stm32l475e_iot01_hsensor.c.

References Hsensor_drv, and HTS221_I2C_ADDRESS.

uint8_t BSP_HSENSOR_ReadID (void)

Read ID of HTS221.

Return values:

HTS221 ID value.

Definition at line 102 of file stm32l475e_iot01_hsensor.c.

References **Hsensor_drv**, and **HTS221_I2C_ADDRESS**.

B-L475E-IOT01 BSP User Manual

Functions

HUMIDITY Exported Functions

HUMIDITY

Functions

uint32_t	**BSP_HSENSOR_Init** (void) Initializes peripherals used by the I2C Humidity Sensor driver.
uint8_t	**BSP_HSENSOR_ReadID** (void) Read ID of HTS221.
float	**BSP_HSENSOR_ReadHumidity** (void) Read Humidity register of HTS221.

Function Documentation

uint32_t BSP_HSENSOR_Init (void)

Initializes peripherals used by the I2C Humidity Sensor driver.

Return values:

HSENSOR status

Definition at line 79 of file stm32l475e_iot01_hsensor.c.

References Hsensor_drv, HSENSOR_ERROR, HSENSOR_OK, and HTS221_I2C_ADDRESS.

float BSP_HSENSOR_ReadHumidity (void)

Read Humidity register of HTS221.

Return values:

HTS221 measured humidity value.

Definition at line 111 of file stm32l475e_iot01_hsensor.c.

References Hsensor_drv, and HTS221_I2C_ADDRESS.

uint8_t BSP_HSENSOR_ReadID (void)

Read ID of HTS221.

Return values:

HTS221 ID value.

Definition at line 102 of file stm32l475e_iot01_hsensor.c.

References **Hsensor_drv**, and **HTS221_I2C_ADDRESS**.

MAGNETO Private Functions

MAGNETO

Functions

MAGNETO_StatusTypeDef	**BSP_MAGNETO_Init** (void) Initialize a magnetometer sensor.
void	**BSP_MAGNETO_DeInit** (void) DeInitialize the MAGNETO.
void	**BSP_MAGNETO_LowPower** (uint16_t status) Set/Unset the MAGNETO in low power mode.
void	**BSP_MAGNETO_GetXYZ** (int16_t *pDataXYZ) Get XYZ magnetometer values.

Function Documentation

void BSP_MAGNETO_DeInit (void)

DeInitialize the MAGNETO.

Definition at line 111 of file stm32l475e_iot01_magneto.c.

References MagnetoDrv.

void BSP_MAGNETO_GetXYZ (int16_t * pDataXYZ)

Get XYZ magnetometer values.

Parameters:

pDataXYZ	Pointer on 3 magnetometer values table with pDataXYZ[0] = X axis, pDataXYZ[1] = Y axis, pDataXYZ[2] = Z axis

Definition at line 143 of file stm32l475e_iot01_magneto.c.

References MagnetoDrv.

MAGNETO_StatusTypeDef BSP_MAGNETO_Init (void)

Initialize a magnetometer sensor.

Return values:

COMPONENT_ERROR	in case of failure

Definition at line 80 of file stm32l475e_iot01_magneto.c.

References MAGNETO_ERROR, MAGNETO_OK, and MagnetoDrv.

void BSP_MAGNETO_LowPower (uint16_t status)

Set/Unset the MAGNETO in low power mode.

Definition at line **126** of file **stm32l475e_iot01_magneto.c**.

References **MagnetoDrv**.

MAGNETO Exported Functions

MAGNETO

Functions

MAGNETO_StatusTypeDef	**BSP_MAGNETO_Init** (void) Initialize a magnetometer sensor.
void	**BSP_MAGNETO_DeInit** (void) DeInitialize the MAGNETO.
void	**BSP_MAGNETO_LowPower** (uint16_t status) Set/Unset the MAGNETO in low power mode.
void	**BSP_MAGNETO_GetXYZ** (int16_t *pDataXYZ) Get XYZ magnetometer values.

Function Documentation

void BSP_MAGNETO_DeInit (void)

DeInitialize the MAGNETO.

Definition at line 111 of file stm32l475e_iot01_magneto.c.

References MagnetoDrv.

void BSP_MAGNETO_GetXYZ (int16_t * pDataXYZ)

Get XYZ magnetometer values.

Parameters:

pDataXYZ	Pointer on 3 magnetometer values table with pDataXYZ[0] = X axis, pDataXYZ[1] = Y axis, pDataXYZ[2] = Z axis

Definition at line 143 of file stm32l475e_iot01_magneto.c.

References MagnetoDrv.

MAGNETO_StatusTypeDef BSP_MAGNETO_Init (void)

Initialize a magnetometer sensor.

Return values:

COMPONENT_ERROR	in case of failure

Definition at line 80 of file stm32l475e_iot01_magneto.c.

References MAGNETO_ERROR, MAGNETO_OK, and MagnetoDrv.

void BSP_MAGNETO_LowPower (uint16_t status)

Set/Unset the MAGNETO in low power mode.

Definition at line **126** of file **stm32l475e_iot01_magneto.c**.

References **MagnetoDrv**.

B-L475E-IOT01 BSP User Manual

Functions

PRESSURE Private Functions

PRESSURE

Functions

uint32_t	**BSP_PSENSOR_Init** (void) Initializes peripherals used by the I2C Pressure Sensor driver.
uint8_t	**BSP_PSENSOR_ReadID** (void) Read ID of LPS22HB.
float	**BSP_PSENSOR_ReadPressure** (void) Read Pressure register of LPS22HB.

Function Documentation

uint32_t BSP_PSENSOR_Init (void)

Initializes peripherals used by the I2C Pressure Sensor driver.

Return values:

PSENSOR status

Definition at line 79 of file stm32l475e_iot01_psensor.c.

References LPS22HB_I2C_ADDRESS, Psensor_drv, PSENSOR_ERROR, and PSENSOR_OK.

uint8_t BSP_PSENSOR_ReadID (void)

Read ID of LPS22HB.

Return values:

LPS22HB ID value.

Definition at line 103 of file stm32l475e_iot01_psensor.c.

References LPS22HB_I2C_ADDRESS, and Psensor_drv.

float BSP_PSENSOR_ReadPressure (void)

Read Pressure register of LPS22HB.

Return values:

LPS22HB measured pressure value.

Definition at line 112 of file stm32l475e_iot01_psensor.c.

References **LPS22HB_I2C_ADDRESS**, and **Psensor_drv**.

PRESSURE Exported Functions

PRESSURE

Functions

uint32_t	**BSP_PSENSOR_Init** (void) Initializes peripherals used by the I2C Pressure Sensor driver.
uint8_t	**BSP_PSENSOR_ReadID** (void) Read ID of LPS22HB.
float	**BSP_PSENSOR_ReadPressure** (void) Read Pressure register of LPS22HB.

Function Documentation

uint32_t BSP_PSENSOR_Init (void)

Initializes peripherals used by the I2C Pressure Sensor driver.

Return values:

PSENSOR status

Definition at line 79 of file stm32l475e_iot01_psensor.c.

References LPS22HB_I2C_ADDRESS, Psensor_drv, PSENSOR_ERROR, and PSENSOR_OK.

uint8_t BSP_PSENSOR_ReadID (void)

Read ID of LPS22HB.

Return values:

LPS22HB ID value.

Definition at line 103 of file stm32l475e_iot01_psensor.c.

References LPS22HB_I2C_ADDRESS, and Psensor_drv.

float BSP_PSENSOR_ReadPressure (void)

Read Pressure register of LPS22HB.

Return values:

LPS22HB measured pressure value.

Definition at line 112 of file stm32l475e_iot01_psensor.c.

References **LPS22HB_I2C_ADDRESS**, and **Psensor_drv**.

B-L475E-IOT01 BSP User Manual

Functions

QSPI Exported Functions

QSPI

Functions

uint8_t	**BSP_QSPI_Init** (void) Initializes the QSPI interface.
uint8_t	**BSP_QSPI_DeInit** (void) De-Initializes the QSPI interface.
uint8_t	**BSP_QSPI_Read** (uint8_t *pData, uint32_t ReadAddr, uint32_t Size) Reads an amount of data from the QSPI memory.
uint8_t	**BSP_QSPI_Write** (uint8_t *pData, uint32_t WriteAddr, uint32_t Size) Writes an amount of data to the QSPI memory.
uint8_t	**BSP_QSPI_Erase_Block** (uint32_t BlockAddress) Erases the specified block of the QSPI memory.
uint8_t	**BSP_QSPI_Erase_Sector** (uint32_t Sector) Erases the specified sector of the QSPI memory.
uint8_t	**BSP_QSPI_Erase_Chip** (void) Erases the entire QSPI memory.
uint8_t	**BSP_QSPI_GetStatus** (void) Reads current status of the QSPI memory.
uint8_t	**BSP_QSPI_GetInfo** (**QSPI_Info** *pInfo) Return the configuration of the QSPI memory.
uint8_t	**BSP_QSPI_EnableMemoryMappedMode** (void) Configure the QSPI in memory-mapped mode.
uint8_t	**BSP_QSPI_SuspendErase** (void) This function suspends an ongoing erase command.
uint8_t	**BSP_QSPI_ResumeErase** (void) This function resumes a paused erase command.
uint8_t	**BSP_QSPI_EnterDeepPowerDown** (void) This function enter the QSPI memory in deep power down mode.
uint8_t	**BSP_QSPI_LeaveDeepPowerDown** (void) This function leave the QSPI memory from deep power

	down mode.
__weak void	**BSP_QSPI_MspInit** (void) Initializes the QSPI MSP.
__weak void	**BSP_QSPI_MspDeInit** (void) De-Initializes the QSPI MSP.

Function Documentation

uint8_t BSP_QSPI_DeInit (void)

De-Initializes the QSPI interface.

Return values:
QSPI memory status

Definition at line **200** of file **stm32l475e_iot01_qspi.c**.

References **BSP_QSPI_MspDeInit()**, **QSPI_ERROR**, **QSPI_OK**, and **QSPIHandle**.

uint8_t BSP_QSPI_EnableMemoryMappedMode (void)

Configure the QSPI in memory-mapped mode.

Return values:
QSPI memory status

Definition at line **554** of file **stm32l475e_iot01_qspi.c**.

References **QSPI_ERROR**, **QSPI_OK**, and **QSPIHandle**.

uint8_t BSP_QSPI_EnterDeepPowerDown (void)

This function enter the QSPI memory in deep power down mode.

Return values:
QSPI memory status

Definition at line **674** of file **stm32l475e_iot01_qspi.c**.

References QSPI_ERROR, QSPI_OK, and QSPIHandle.

uint8_t BSP_QSPI_Erase_Block (uint32_t BlockAddress)

Erases the specified block of the QSPI memory.

Parameters:

BlockAddress : Block address to erase

Return values:

QSPI memory status

Definition at line 339 of file stm32l475e_iot01_qspi.c.

References QSPI_AutoPollingMemReady(), QSPI_ERROR, QSPI_OK, QSPI_WriteEnable(), and QSPIHandle.

uint8_t BSP_QSPI_Erase_Chip (void)

Erases the entire QSPI memory.

Return values:

QSPI memory status

Definition at line 428 of file stm32l475e_iot01_qspi.c.

References QSPI_AutoPollingMemReady(), QSPI_ERROR, QSPI_OK, QSPI_WriteEnable(), and QSPIHandle.

uint8_t BSP_QSPI_Erase_Sector (uint32_t Sector)

Erases the specified sector of the QSPI memory.

Parameters:

Sector : Sector address to erase (0 to 255)

Return values:

QSPI memory status

Note:

This function is non blocking meaning that sector erase operation is started but not completed when the function returns. Application has to call **BSP_QSPI_GetStatus()** to know when the device is available again (i.e. erase operation completed).

Definition at line **387** of file **stm32l475e_iot01_qspi.c**.

References **QSPI_ERROR**, **QSPI_OK**, **QSPI_WriteEnable()**, and **QSPIHandle**.

uint8_t BSP_QSPI_GetInfo (QSPI_Info * pInfo)

Return the configuration of the QSPI memory.

Parameters:

pInfo : pointer on the configuration structure

Return values:

QSPI memory status

Definition at line **538** of file **stm32l475e_iot01_qspi.c**.

References **QSPI_Info::EraseSectorSize**, **QSPI_Info::EraseSectorsNumber**, **QSPI_Info::FlashSize**, **QSPI_Info::ProgPageSize**, **QSPI_Info::ProgPagesNumber**, and **QSPI_OK**.

uint8_t BSP_QSPI_GetStatus (void)

Reads current status of the QSPI memory.

Return values:

QSPI memory status

Definition at line **468** of file **stm32l475e_iot01_qspi.c**.

References **QSPI_BUSY**, **QSPI_ERROR**, **QSPI_OK**, **QSPI_SUSPENDED**, and **QSPIHandle**.

Referenced by **BSP_QSPI_ResumeErase()**, and **BSP_QSPI_SuspendErase()**.

uint8_t BSP_QSPI_Init (void)

Initializes the QSPI interface.

Return values:

QSPI memory status

Definition at line **141** of file **stm32l475e_iot01_qspi.c**.

References **BSP_QSPI_MspInit()**, **QSPI_ERROR**, **QSPI_HIGH_PERF_ENABLE**, **QSPI_HighPerfMode()**, **QSPI_NOT_SUPPORTED**, **QSPI_OK**, **QSPI_QUAD_ENABLE**, **QSPI_QuadMode()**, **QSPI_ResetMemory()**, and **QSPIHandle**.

uint8_t BSP_QSPI_LeaveDeepPowerDown (void)

This function leave the QSPI memory from deep power down mode.

Return values:

QSPI memory status

Definition at line **705** of file **stm32l475e_iot01_qspi.c**.

References **QSPI_ERROR**, **QSPI_OK**, and **QSPIHandle**.

void BSP_QSPI_MspDeInit (void)

De-Initializes the QSPI MSP.

Return values:
None

Definition at line **764** of file **stm32l475e_iot01_qspi.c**.

Referenced by **BSP_QSPI_DeInit()**.

void BSP_QSPI_MspInit (void)

Initializes the QSPI MSP.

Return values:
None

Definition at line **736** of file **stm32l475e_iot01_qspi.c**.

Referenced by **BSP_QSPI_Init()**.

uint8_t BSP_QSPI_Read (uint8_t * pData, uint32_t ReadAddr, uint32_t Size)

Reads an amount of data from the QSPI memory.

Parameters:
pData : Pointer to data to be read
ReadAddr : Read start address

Size : Size of data to read

Return values:

QSPI memory status

Definition at line **223** of file **stm32l475e_iot01_qspi.c**.

References **QSPI_ERROR**, **QSPI_OK**, and **QSPIHandle**.

uint8_t BSP_QSPI_ResumeErase (void)

This function resumes a paused erase command.

Return values:

QSPI memory status

Definition at line **629** of file **stm32l475e_iot01_qspi.c**.

References **BSP_QSPI_GetStatus()**, **QSPI_BUSY**, **QSPI_ERROR**, **QSPI_OK**, **QSPI_SUSPENDED**, and **QSPIHandle**.

uint8_t BSP_QSPI_SuspendErase (void)

This function suspends an ongoing erase command.

Return values:

QSPI memory status

Definition at line **588** of file **stm32l475e_iot01_qspi.c**.

References **BSP_QSPI_GetStatus()**, **QSPI_BUSY**, **QSPI_ERROR**, **QSPI_OK**, **QSPI_SUSPENDED**, and **QSPIHandle**.

uint8_t BSP_QSPI_Write (uint8_t * pData,

```
                                    uint32_t  WriteAddr,
                                    uint32_t  Size
                                  )
```

Writes an amount of data to the QSPI memory.

Parameters:

pData	: Pointer to data to be written
WriteAddr	: Write start address
Size	: Size of data to write

Return values:

QSPI memory status

Definition at line **265** of file **stm32l475e_iot01_qspi.c**.

References **QSPI_AutoPollingMemReady()**, **QSPI_ERROR**, **QSPI_OK**, **QSPI_WriteEnable()**, and **QSPIHandle**.

B-L475E-IOT01 BSP User Manual

TEMPERATURE
Private Functions

TEMPERATURE

Functions

uint32_t	**BSP_TSENSOR_Init** (void) Initializes peripherals used by the I2C Temperature Sensor driver.
float	**BSP_TSENSOR_ReadTemp** (void) Read Temperature register of TS751.

Function Documentation

uint32_t BSP_TSENSOR_Init (void)

Initializes peripherals used by the I2C Temperature Sensor driver.

Return values:

TSENSOR status

Definition at line **79** of file **stm32l475e_iot01_tsensor.c**.

References **SENSOR_IO_Init()**, **tsensor_drv**, **TSENSOR_ERROR**, **TSENSOR_I2C_ADDRESS**, and **TSENSOR_OK**.

float BSP_TSENSOR_ReadTemp (void)

Read Temperature register of TS751.

Return values:

STTS751 measured temperature value.

Definition at line **104** of file **stm32l475e_iot01_tsensor.c**.

References **tsensor_drv**, and **TSENSOR_I2C_ADDRESS**.

B-L475E-IOT01 BSP User Manual

TEMPERATURE
Exported Constants

TEMPERATURE

Functions

uint32_t	**BSP_TSENSOR_Init** (void) Initializes peripherals used by the I2C Temperature Sensor driver.
float	**BSP_TSENSOR_ReadTemp** (void) Read Temperature register of TS751.

Function Documentation

uint32_t BSP_TSENSOR_Init (void)

Initializes peripherals used by the I2C Temperature Sensor driver.

Return values:

TSENSOR status

Definition at line **79** of file **stm32l475e_iot01_tsensor.c**.

References **SENSOR_IO_Init()**, **tsensor_drv**, **TSENSOR_ERROR**, **TSENSOR_I2C_ADDRESS**, and **TSENSOR_OK**.

float BSP_TSENSOR_ReadTemp (void)

Read Temperature register of TS751.

Return values:

STTS751 measured temperature value.

Definition at line **104** of file **stm32l475e_iot01_tsensor.c**.

References **tsensor_drv**, and **TSENSOR_I2C_ADDRESS**.

B-L475E-IOT01 BSP User Manual

LOW LEVEL Variables

LOW LEVEL

Variables

const uint32_t	**GPIO_PIN** [**LEDn**] = {**LED2_PIN**}
GPIO_TypeDef *	**GPIO_PORT** [**LEDn**] = {**LED2_GPIO_PORT**}
GPIO_TypeDef *	**BUTTON_PORT** [**BUTTONn**] = {**USER_BUTTON_GPIO_PORT**}
const uint16_t	**BUTTON_PIN** [**BUTTONn**] = {**USER_BUTTON_PIN**}
const uint16_t	**BUTTON_IRQn** [**BUTTONn**] = {**USER_BUTTON_EXTI_IRQn**}
USART_TypeDef *	**COM_USART** [**COMn**] = {**DISCOVERY_COM1**}
GPIO_TypeDef *	**COM_TX_PORT** [**COMn**] = {**DISCOVERY_COM1_TX_GPIO_PORT**}
GPIO_TypeDef *	**COM_RX_PORT** [**COMn**] = {**DISCOVERY_COM1_RX_GPIO_PORT**}
const uint16_t	**COM_TX_PIN** [**COMn**] = {**DISCOVERY_COM1_TX_PIN**}
const uint16_t	**COM_RX_PIN** [**COMn**] = {**DISCOVERY_COM1_RX_PIN**}
const uint16_t	**COM_TX_AF** [**COMn**] = {**DISCOVERY_COM1_TX_AF**}
const uint16_t	**COM_RX_AF** [**COMn**] = {**DISCOVERY_COM1_RX_AF**}
I2C_HandleTypeDef	**hI2cHandler**
UART_HandleTypeDef	**hDiscoUart**

Variable Documentation

const uint16_t BUTTON_IRQn[BUTTONn] = {USER_BUTTON_EXTI_

Definition at line 94 of file stm32l475e_iot01.c.

Referenced by BSP_PB_DeInit(), and BSP_PB_Init().

const uint16_t BUTTON_PIN[BUTTONn] = {USER_BUTTON_PIN}

Definition at line 92 of file stm32l475e_iot01.c.

Referenced by BSP_PB_DeInit(), BSP_PB_GetState(), and BSP_PB_Init().

GPIO_TypeDef* BUTTON_PORT[BUTTONn] = {USER_BUTTON_GPI

Definition at line 90 of file stm32l475e_iot01.c.

Referenced by BSP_PB_DeInit(), BSP_PB_GetState(), and BSP_PB_Init().

const uint16_t COM_RX_AF[COMn] = {DISCOVERY_COM1_RX_AF

Definition at line 108 of file stm32l475e_iot01.c.

Referenced by BSP_COM_Init().

const uint16_t COM_RX_PIN[COMn] = {DISCOVERY_COM1_RX_PI

Definition at line 104 of file stm32l475e_iot01.c.

Referenced by **BSP_COM_Init()**.

GPIO_TypeDef* COM_RX_PORT[COMn] = {DISCOVERY_COM1_RX

Definition at line **100** of file **stm32l475e_iot01.c**.

Referenced by **BSP_COM_Init()**.

const uint16_t COM_TX_AF[COMn] = {DISCOVERY_COM1_TX_AF}

Definition at line **106** of file **stm32l475e_iot01.c**.

Referenced by **BSP_COM_Init()**.

const uint16_t COM_TX_PIN[COMn] = {DISCOVERY_COM1_TX_PIN

Definition at line **102** of file **stm32l475e_iot01.c**.

Referenced by **BSP_COM_Init()**.

GPIO_TypeDef* COM_TX_PORT[COMn] = {DISCOVERY_COM1_TX_

Definition at line **98** of file **stm32l475e_iot01.c**.

Referenced by **BSP_COM_Init()**.

USART_TypeDef* COM_USART[COMn] = {DISCOVERY_COM1}

Definition at line **96** of file **stm32l475e_iot01.c**.

Referenced by **BSP_COM_DeInit()**, and **BSP_COM_Init()**.

const uint32_t GPIO_PIN[LEDn] = {LED2_PIN}

Definition at line **84** of file **stm32l475e_iot01.c**.

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

GPIO_TypeDef* GPIO_PORT[LEDn] = {LED2_GPIO_PORT}

Definition at line **87** of file **stm32l475e_iot01.c**.

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

UART_HandleTypeDef hDiscoUart

Definition at line **111** of file **stm32l475e_iot01.c**.

I2C_HandleTypeDef hI2cHandler

Definition at line **110** of file **stm32l475e_iot01.c**.

Referenced by **SENSOR_IO_DeInit()**, **SENSOR_IO_Init()**, **SENSOR_IO_IsDeviceReady()**, **SENSOR_IO_Read()**, **SENSOR_IO_ReadMultiple()**, **SENSOR_IO_Write()**, and **SENSOR_IO_WriteMultiple()**.

B-L475E-IOT01 BSP User Manual

LOW LEVEL Exported Types

LOW LEVEL

Enumerations

enum	**Led_TypeDef** { **LED2** = 0, **LED_GREEN** = LED2 }
enum	**Button_TypeDef** { **BUTTON_USER** = 0 }
enum	**ButtonMode_TypeDef** { **BUTTON_MODE_GPIO** = 0, **BUTTON_MODE_EXTI** = 1 }
enum	**COM_TypeDef** { **COM1** = 0, **COM2** = 0 }

Enumeration Type Documentation

enum Button_TypeDef

Enumerator:

BUTTON_USER

Definition at line **81** of file **stm32l475e_iot01.h**.

enum ButtonMode_TypeDef

Enumerator:

BUTTON_MODE_GPIO

BUTTON_MODE_EXTI

Definition at line **86** of file **stm32l475e_iot01.h**.

enum COM_TypeDef

Enumerator:

COM1

COM2

Definition at line **92** of file **stm32l475e_iot01.h**.

enum Led_TypeDef

Enumerator:

LED2

LED_GREEN

Definition at line **74** of file **stm32l475e_iot01.h**.

B-L475E-IOT01 BSP User Manual

Defines

LOW LEVEL Exported Constants

LOW LEVEL

Defines

#define **LEDn** ((uint8_t)1)
Define for STM32L475E_IOT01 board.

#define **LED2_PIN** GPIO_PIN_14

#define **LED2_GPIO_PORT** GPIOB

#define **LED2_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_CLK_E

#define **LED2_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_CLK_D

#define **LEDx_GPIO_CLK_ENABLE**(__INDEX__) do{if((__INDEX__)
LED2_GPIO_CLK_ENABLE();}while(0)

#define **LEDx_GPIO_CLK_DISABLE**(__INDEX__) do{if((__INDEX__
LED2_GPIO_CLK_DISABLE();}while(0)

#define **BUTTONn** ((uint8_t)1)

#define **USER_BUTTON_PIN** GPIO_PIN_13
Wakeup push-button.

#define **USER_BUTTON_GPIO_PORT** GPIOC

#define **USER_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_GPIC

#define **USER_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_GPI

#define **USER_BUTTON_EXTI_IRQn** EXTI15_10_IRQn

#define **COMn** ((uint8_t)1)

#define **DISCOVERY_COM1** USART1
Definition for COM port1, connected to USART1.

#define **DISCOVERY_COM1_CLK_ENABLE**() __HAL_RCC_USART

#define **DISCOVERY_COM1_CLK_DISABLE**() __HAL_RCC_USART

#define **DISCOVERY_COM1_TX_PIN** GPIO_PIN_6

#define **DISCOVERY_COM1_TX_GPIO_PORT** GPIOB

#define **DISCOVERY_COM1_TX_GPIO_CLK_ENABLE**() __HAL_RC

#define **DISCOVERY_COM1_TX_GPIO_CLK_DISABLE**() __HAL_R

#define **DISCOVERY_COM1_TX_AF** GPIO_AF7_USART1

#define **DISCOVERY_COM1_RX_PIN** GPIO_PIN_7

#define **DISCOVERY_COM1_RX_GPIO_PORT** GPIOB

#define **DISCOVERY_COM1_RX_GPIO_CLK_ENABLE**() __HAL_RC

#define **DISCOVERY_COM1_RX_GPIO_CLK_DISABLE**() __HAL_R

```
#define DISCOVERY_COM1_RX_AF   GPIO_AF7_USART1
#define DISCOVERY_COM1_IRQn   USART1_IRQn
#define DISCOVERY_COMx_CLK_ENABLE(__INDEX__)   do { if((__
                {DISCOVERY_COM1_CLK_ENABLE();}} while(0)
#define DISCOVERY_COMx_CLK_DISABLE(__INDEX__)   do { if((__
                {DISCOVERY_COM1_CLK_DISABLE();}} while(0)
#define DISCOVERY_COMx_TX_GPIO_CLK_ENABLE(__INDEX__)
                {DISCOVERY_COM1_TX_GPIO_CLK_ENABLE();}} while(0)
#define DISCOVERY_COMx_TX_GPIO_CLK_DISABLE(__INDEX__)
                {DISCOVERY_COM1_TX_GPIO_CLK_DISABLE();}} while(0)
#define DISCOVERY_COMx_RX_GPIO_CLK_ENABLE(__INDEX__)
                {DISCOVERY_COM1_RX_GPIO_CLK_ENABLE();}} while(0)
#define DISCOVERY_COMx_RX_GPIO_CLK_DISABLE(__INDEX__)
                {DISCOVERY_COM1_RX_GPIO_CLK_DISABLE();}} while(0)
#define DISCOVERY_I2Cx   I2C2
#define DISCOVERY_I2Cx_CLK_ENABLE()   __HAL_RCC_I2C2_CLI
#define DISCOVERY_I2Cx_CLK_DISABLE()   __HAL_RCC_I2C2_CL
#define DISCOVERY_DMAx_CLK_ENABLE()   __HAL_RCC_DMA1_
#define DISCOVERY_I2Cx_SCL_SDA_GPIO_CLK_ENABLE()   __HA
#define DISCOVERY_I2Cx_SCL_SDA_GPIO_CLK_DISABLE()   __H
#define DISCOVERY_I2Cx_FORCE_RESET()   __HAL_RCC_I2C2_F
#define DISCOVERY_I2Cx_RELEASE_RESET()   __HAL_RCC_I2C2_
#define DISCOVERY_I2Cx_SCL_PIN   GPIO_PIN_10
#define DISCOVERY_I2Cx_SDA_PIN   GPIO_PIN_11
#define DISCOVERY_I2Cx_SCL_SDA_GPIO_PORT   GPIOB
#define DISCOVERY_I2Cx_SCL_SDA_AF   GPIO_AF4_I2C2
#define DISCOVERY_I2Cx_EV_IRQn   I2C2_EV_IRQn
#define DISCOVERY_I2Cx_ER_IRQn   I2C2_ER_IRQn
#define LPS22HB_I2C_ADDRESS   (uint8_t)0xBA
#define HTS221_I2C_ADDRESS   (uint8_t)0xBE
#define TSENSOR_I2C_ADDRESS   HTS221_I2C_ADDRESS
```

Define Documentation

#define BUTTONn ((uint8_t)1)

Definition at line 126 of file stm32l475e_iot01.h.

#define COMn ((uint8_t)1)

Definition at line 139 of file stm32l475e_iot01.h.

#define DISCOVERY_COM1 USART1

Definition for COM port1, connected to USART1.

Definition at line 144 of file stm32l475e_iot01.h.

#define DISCOVERY_COM1_CLK_DISABLE ( ) __HAL_RCC_USA

Definition at line 146 of file stm32l475e_iot01.h.

#define DISCOVERY_COM1_CLK_ENABLE ( ) __HAL_RCC_USAF

Definition at line 145 of file stm32l475e_iot01.h.

#define DISCOVERY_COM1_IRQn USART1_IRQn

Definition at line 160 of file stm32l475e_iot01.h.

#define DISCOVERY_COM1_RX_AF GPIO_AF7_USART1

Definition at line 158 of file stm32l475e_iot01.h.

#define DISCOVERY_COM1_RX_GPIO_CLK_DISABLE ( ) __HAL_

Definition at line 157 of file stm32l475e_iot01.h.

#define DISCOVERY_COM1_RX_GPIO_CLK_ENABLE ( ) __HAL_I

Definition at line 156 of file stm32l475e_iot01.h.

#define DISCOVERY_COM1_RX_GPIO_PORT GPIOB

Definition at line 155 of file stm32l475e_iot01.h.

#define DISCOVERY_COM1_RX_PIN GPIO_PIN_7

Definition at line 154 of file stm32l475e_iot01.h.

#define DISCOVERY_COM1_TX_AF GPIO_AF7_USART1

Definition at line 152 of file stm32l475e_iot01.h.

#define DISCOVERY_COM1_TX_GPIO_CLK_DISABLE ( ) __HAL_

Definition at line 151 of file stm32l475e_iot01.h.

#define DISCOVERY_COM1_TX_GPIO_CLK_ENABLE ( ) __HAL_I

Definition at line 150 of file stm32l475e_iot01.h.

#define DISCOVERY_COM1_TX_GPIO_PORT GPIOB

Definition at line 149 of file stm32l475e_iot01.h.

#define DISCOVERY_COM1_TX_PIN GPIO_PIN_6

Definition at line 148 of file stm32l475e_iot01.h.

#define DISCOVERY_COMx_CLK_DISABLE (__INDEX__) do { if

Definition at line 164 of file stm32l475e_iot01.h.

Referenced by BSP_COM_DeInit().

#define DISCOVERY_COMx_CLK_ENABLE (__INDEX__) do { if(

Definition at line 163 of file stm32l475e_iot01.h.

Referenced by BSP_COM_Init().

#define DISCOVERY_COMx_RX_GPIO_CLK_DISABLE (__INDEX_

Definition at line 170 of file stm32l475e_iot01.h.

#define DISCOVERY_COMx_RX_GPIO_CLK_ENABLE (__INDEX__

Definition at line 169 of file stm32l475e_iot01.h.

Referenced by BSP_COM_Init().

#define DISCOVERY_COMx_TX_GPIO_CLK_DISABLE (__INDEX__

Definition at line 167 of file stm32l475e_iot01.h.

#define DISCOVERY_COMx_TX_GPIO_CLK_ENABLE (__INDEX__

Definition at line 166 of file stm32l475e_iot01.h.

Referenced by BSP_COM_Init().

#define DISCOVERY_DMAx_CLK_ENABLE ( ) __HAL_RCC_DMA1

Definition at line 178 of file stm32l475e_iot01.h.

#define DISCOVERY_I2Cx I2C2

Definition at line 175 of file stm32l475e_iot01.h.

Referenced by I2Cx_Init().

#define DISCOVERY_I2Cx_CLK_DISABLE ( ) __HAL_RCC_I2C2_C

Definition at line 177 of file stm32l475e_iot01.h.

Referenced by I2Cx_MspDeInit().

#define DISCOVERY_I2Cx_CLK_ENABLE ( ) __HAL_RCC_I2C2_C

Definition at line 176 of file stm32l475e_iot01.h.

Referenced by I2Cx_MspInit().

#define DISCOVERY_I2Cx_ER_IRQn I2C2_ER_IRQn

Definition at line 193 of file stm32l475e_iot01.h.

Referenced by I2Cx_MspInit().

#define DISCOVERY_I2Cx_EV_IRQn I2C2_EV_IRQn

Definition at line 192 of file stm32l475e_iot01.h.

Referenced by I2Cx_MspInit().

#define DISCOVERY_I2Cx_FORCE_RESET ( ) __HAL_RCC_I2C2_

Definition at line 182 of file stm32l475e_iot01.h.

Referenced by I2Cx_MspInit().

#define DISCOVERY_I2Cx_RELEASE_RESET ( ) __HAL_RCC_I2C

Definition at line 183 of file stm32l475e_iot01.h.

Referenced by I2Cx_MspInit().

#define DISCOVERY_I2Cx_SCL_PIN GPIO_PIN_10

Definition at line 186 of file stm32l475e_iot01.h.

Referenced by I2Cx_MspDeInit(), and I2Cx_MspInit().

#define DISCOVERY_I2Cx_SCL_SDA_AF GPIO_AF4_I2C2

Definition at line 189 of file stm32l475e_iot01.h.

Referenced by I2Cx_MspInit().

#define DISCOVERY_I2Cx_SCL_SDA_GPIO_CLK_DISABLE ( ) __

Definition at line 180 of file stm32l475e_iot01.h.

Referenced by I2Cx_MspDeInit().

#define DISCOVERY_I2Cx_SCL_SDA_GPIO_CLK_ENABLE ( ) __H

Definition at line 179 of file stm32l475e_iot01.h.

Referenced by I2Cx_MspInit().

#define DISCOVERY_I2Cx_SCL_SDA_GPIO_PORT GPIOB

Definition at line 188 of file stm32l475e_iot01.h.

Referenced by I2Cx_MspDeInit(), and I2Cx_MspInit().

#define DISCOVERY_I2Cx_SDA_PIN GPIO_PIN_11

Definition at line 187 of file stm32l475e_iot01.h.

Referenced by I2Cx_MspDeInit(), and I2Cx_MspInit().

#define HTS221_I2C_ADDRESS (uint8_t)0xBE

Definition at line 212 of file stm32l475e_iot01.h.

Referenced by **BSP_HSENSOR_Init()**, **BSP_HSENSOR_ReadHumidity()**, and **BSP_HSENSOR_ReadID()**.

#define LED2_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOB_CLK

Definition at line **117** of file **stm32l475e_iot01.h**.

#define LED2_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOB_CLK_

Definition at line **116** of file **stm32l475e_iot01.h**.

#define LED2_GPIO_PORT GPIOB

Definition at line **115** of file **stm32l475e_iot01.h**.

#define LED2_PIN GPIO_PIN_14

Definition at line **114** of file **stm32l475e_iot01.h**.

#define LEDn ((uint8_t)1)

Define for STM32L475E_IOT01 board.

Definition at line **112** of file **stm32l475e_iot01.h**.

#define LEDx_GPIO_CLK_DISABLE (__INDEX__) do{if((__INDE

Definition at line **123** of file **stm32l475e_iot01.h**.

#define LEDx_GPIO_CLK_ENABLE (__INDEX__) do{if((__INDEX

Definition at line 121 of file stm32l475e_iot01.h.

Referenced by BSP_LED_Init().

#define LPS22HB_I2C_ADDRESS (uint8_t)0xBA

Definition at line 210 of file stm32l475e_iot01.h.

Referenced by BSP_PSENSOR_Init(), BSP_PSENSOR_ReadID(), and BSP_PSENSOR_ReadPressure().

#define TSENSOR_I2C_ADDRESS HTS221_I2C_ADDRESS

Definition at line 219 of file stm32l475e_iot01.h.

Referenced by BSP_TSENSOR_Init(), and BSP_TSENSOR_ReadTemp().

#define USER_BUTTON_EXTI_IRQn EXTI15_10_IRQn

Definition at line 135 of file stm32l475e_iot01.h.

#define USER_BUTTON_GPIO_CLK_DISABLE ( ) __HAL_RCC_GI

Definition at line 134 of file stm32l475e_iot01.h.

#define USER_BUTTON_GPIO_CLK_ENABLE ( ) __HAL_RCC_GP

Definition at line 133 of file stm32l475e_iot01.h.

Referenced by BSP_PB_Init().

#define USER_BUTTON_GPIO_PORT GPIOC

Definition at line 132 of file stm32l475e_iot01.h.

#define USER_BUTTON_PIN GPIO_PIN_13

Wakeup push-button.

Definition at line 131 of file stm32l475e_iot01.h.

GYROSCOPE Exported Constants

GYROSCOPE

Enumerations

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

Enumeration Type Documentation

enum GYRO_StatusTypeDef

Enumerator:

GYRO_OK

GYRO_ERROR

GYRO_TIMEOUT

Definition at line **76** of file **stm32l475e_iot01_gyro.h**.

B-L475E-IOT01 BSP User Manual

GYROSCOPE Private Variables

GYROSCOPE

Variables

static GYRO_DrvTypeDef * **GyroscopeDrv**

Variable Documentation

GYRO_DrvTypeDef* GyroscopeDrv [static]

Definition at line **66** of file **stm32l475e_iot01_gyro.c**.

Referenced by **BSP_GYRO_DeInit()**, **BSP_GYRO_GetXYZ()**, **BSP_GYRO_Init()**, and **BSP_GYRO_LowPower()**.

B-L475E-IOT01 BSP User Manual

Variables

HUMIDITY Private Variables

HUMIDITY

Variables

static HSENSOR_DrvTypeDef *	**Hsensor_drv**

Variable Documentation

HSENSOR_DrvTypeDef* Hsensor_drv [static]

Definition at line **66** of file **stm32l475e_iot01_hsensor.c**.

Referenced by **BSP_HSENSOR_Init()**, **BSP_HSENSOR_ReadHumidity()**, and **BSP_HSENSOR_ReadID()**.

B-L475E-IOT01 BSP User Manual

Enumerations

HUMIDITY Exported Types

HUMIDITY

Enumerations

enum	**HSENSOR_Status_TypDef** { **HSENSOR_OK** = 0, **HSENSOR_ERROR** } HSENSOR Status. More...

Enumeration Type Documentation

enum HSENSOR_Status_TypDef

HSENSOR Status.

Enumerator:

HSENSOR_OK

HSENSOR_ERROR

Definition at line **80** of file **stm32l475e_iot01_hsensor.h**.

B-L475E-IOT01 BSP User Manual

Enumerations

MAGNETO Exported Types

MAGNETO

Enumerations

enum	**MAGNETO_StatusTypeDef** { **MAGNETO_OK** = 0, **MAGNETO_ERROR** = 1, **MAGNETO_TIMEOUT** = 2 }

Enumeration Type Documentation

enum MAGNETO_StatusTypeDef

Enumerator:

MAGNETO_OK

MAGNETO_ERROR

MAGNETO_TIMEOUT

Definition at line **77** of file **stm32l475e_iot01_magneto.h**.

B-L475E-IOT01 BSP User Manual

MAGNETO Private Variables

MAGNETO

Variables

static MAGNETO_DrvTypeDef *	**MagnetoDrv**

Variable Documentation

MAGNETO_DrvTypeDef* MagnetoDrv [static]

Definition at line **66** of file **stm32l475e_iot01_magneto.c**.

Referenced by **BSP_MAGNETO_DeInit()**, **BSP_MAGNETO_GetXYZ()**, **BSP_MAGNETO_Init()**, and **BSP_MAGNETO_LowPower()**.

B-L475E-IOT01 BSP User Manual

Variables

PRESSURE Private Variables

PRESSURE

Variables

static PSENSOR_DrvTypeDef * **Psensor_drv**

Variable Documentation

PSENSOR_DrvTypeDef* Psensor_drv [static]

Definition at line **66** of file **stm32l475e_iot01_psensor.c**.

Referenced by **BSP_PSENSOR_Init()**, **BSP_PSENSOR_ReadID()**, and **BSP_PSENSOR_ReadPressure()**.

B-L475E-IOT01 BSP User Manual

Enumerations

PRESSURE Exported Types

PRESSURE

Enumerations

enum	**PSENSOR_Status_TypDef** { **PSENSOR_OK** = 0, **PSENSOR_ERROR** } PSENSOR Status. More...

Enumeration Type Documentation

enum PSENSOR_Status_TypDef

PSENSOR Status.

Enumerator:

PSENSOR_OK

PSENSOR_ERROR

Definition at line **80** of file **stm32l475e_iot01_psensor.h**.

B-L475E-IOT01 BSP User Manual

QSPI Private Functions

QSPI

Functions

static uint8_t	**QSPI_ResetMemory** (QSPI_HandleTypeDef *hqspi) This function reset the QSPI memory.
static uint8_t	**QSPI_WriteEnable** (QSPI_HandleTypeDef *hqspi) This function send a Write Enable and wait it is effective.
static uint8_t	**QSPI_AutoPollingMemReady** (QSPI_HandleTypeDef *hqspi, uint32_t Timeout) This function read the SR of the memory and wait the EOP.
static uint8_t	**QSPI_QuadMode** (QSPI_HandleTypeDef *hqspi, uint8_t Operation) This function enables/disables the Quad mode of the memory.
static uint8_t	**QSPI_HighPerfMode** (QSPI_HandleTypeDef *hqspi, uint8_t Operation) This function enables/disables the high performance mode of the memory.

Function Documentation

**static uint8_t QSPI_AutoPollingMemReady (QSPI_HandleTypeDef *
uint32_t
)**

This function read the SR of the memory and wait the EOP.

Parameters:

hqspi : QSPI handle
Timeout : Timeout for auto-polling

Return values:

None

Definition at line **884** of file **stm32l475e_iot01_qspi.c**.

References **QSPI_ERROR**, **QSPI_OK**, and **QSPIHandle**.

Referenced by **BSP_QSPI_Erase_Block()**, **BSP_QSPI_Erase_Chip()**, **BSP_QSPI_Write()**, **QSPI_HighPerfMode()**, **QSPI_QuadMode()**, and **QSPI_ResetMemory()**.

**static uint8_t QSPI_HighPerfMode (QSPI_HandleTypeDef * hqspi,
uint8_t Operati
) [static**

This function enables/disables the high performance mode of the memory.

Parameters:

hqspi : QSPI handle
Operation : QSPI_HIGH_PERF_ENABLE or

QSPI_HIGH_PERF_DISABLE high performance mode

Return values:

None

Definition at line **1010** of file **stm32l475e_iot01_qspi.c**.

References **QSPI_AutoPollingMemReady()**, **QSPI_ERROR**, **QSPI_HIGH_PERF_DISABLE**, **QSPI_HIGH_PERF_ENABLE**, **QSPI_OK**, **QSPI_WriteEnable()**, and **QSPIHandle**.

Referenced by **BSP_QSPI_Init()**.

static uint8_t QSPI_QuadMode (QSPI_HandleTypeDef * hqspi, uint8_t Operation) [static]

This function enables/disables the Quad mode of the memory.

Parameters:

hqspi : QSPI handle

Operation : QSPI_QUAD_ENABLE or QSPI_QUAD_DISABLE mode

Return values:

None

Definition at line **921** of file **stm32l475e_iot01_qspi.c**.

References **QSPI_AutoPollingMemReady()**, **QSPI_ERROR**, **QSPI_OK**, **QSPI_QUAD_DISABLE**, **QSPI_QUAD_ENABLE**, **QSPI_WriteEnable()**, and **QSPIHandle**.

Referenced by **BSP_QSPI_Init()**.

static uint8_t QSPI_ResetMemory (QSPI_HandleTypeDef * hqspi)

This function reset the QSPI memory.

Parameters:

hqspi : QSPI handle

Return values:

None

Definition at line **796** of file **stm32l475e_iot01_qspi.c**.

References **QSPI_AutoPollingMemReady()**, **QSPI_ERROR**, **QSPI_OK**, and **QSPIHandle**.

Referenced by **BSP_QSPI_Init()**.

static uint8_t QSPI_WriteEnable (QSPI_HandleTypeDef * hqspi) [s

This function send a Write Enable and wait it is effective.

Parameters:

hqspi : QSPI handle

Return values:

None

Definition at line **838** of file **stm32l475e_iot01_qspi.c**.

References **QSPI_ERROR**, **QSPI_OK**, and **QSPIHandle**.

Referenced by **BSP_QSPI_Erase_Block()**, **BSP_QSPI_Erase_Chip()**, **BSP_QSPI_Erase_Sector()**, **BSP_QSPI_Write()**, **QSPI_HighPerfMode()**, and **QSPI_QuadMode()**.

Generated on Thu Mar 16 2017 10:38:33 for B-L475E-IOT01 BSP User Manual by doxygen 1.7.6.1

B-L475E-IOT01 BSP User Manual

Defines

QSPI Exported Constants

QSPI

Defines

#define **QSPI_OK** ((uint8_t)0x00)
#define **QSPI_ERROR** ((uint8_t)0x01)
#define **QSPI_BUSY** ((uint8_t)0x02)
#define **QSPI_NOT_SUPPORTED** ((uint8_t)0x04)
#define **QSPI_SUSPENDED** ((uint8_t)0x08)

Define Documentation

#define QSPI_BUSY ((uint8_t)0x02)

Definition at line 70 of file stm32l475e_iot01_qspi.h.

Referenced by BSP_QSPI_GetStatus(), BSP_QSPI_ResumeErase(), and BSP_QSPI_SuspendErase().

#define QSPI_ERROR ((uint8_t)0x01)

Definition at line 69 of file stm32l475e_iot01_qspi.h.

Referenced by BSP_QSPI_DeInit(), BSP_QSPI_EnableMemoryMappedMode(), BSP_QSPI_EnterDeepPowerDown(), BSP_QSPI_Erase_Block(), BSP_QSPI_Erase_Chip(), BSP_QSPI_Erase_Sector(), BSP_QSPI_GetStatus(), BSP_QSPI_Init(), BSP_QSPI_LeaveDeepPowerDown(), BSP_QSPI_Read(), BSP_QSPI_ResumeErase(), BSP_QSPI_SuspendErase(), BSP_QSPI_Write(), QSPI_AutoPollingMemReady(), QSPI_HighPerfMode(), QSPI_QuadMode(), QSPI_ResetMemory(), and QSPI_WriteEnable().

#define QSPI_NOT_SUPPORTED ((uint8_t)0x04)

Definition at line 71 of file stm32l475e_iot01_qspi.h.

Referenced by BSP_QSPI_Init().

#define QSPI_OK ((uint8_t)0x00)

Definition at line 68 of file stm32l475e_iot01_qspi.h.

Referenced by **BSP_QSPI_DeInit()**, **BSP_QSPI_EnableMemoryMappedMode()**, **BSP_QSPI_EnterDeepPowerDown()**, **BSP_QSPI_Erase_Block()**, **BSP_QSPI_Erase_Chip()**, **BSP_QSPI_Erase_Sector()**, **BSP_QSPI_GetInfo()**, **BSP_QSPI_GetStatus()**, **BSP_QSPI_Init()**, **BSP_QSPI_LeaveDeepPowerDown()**, **BSP_QSPI_Read()**, **BSP_QSPI_ResumeErase()**, **BSP_QSPI_SuspendErase()**, **BSP_QSPI_Write()**, **QSPI_AutoPollingMemReady()**, **QSPI_HighPerfMode()**, **QSPI_QuadMode()**, **QSPI_ResetMemory()**, and **QSPI_WriteEnable()**.

#define QSPI_SUSPENDED ((uint8_t)0x08)

Definition at line **72** of file **stm32l475e_iot01_qspi.h**.

Referenced by **BSP_QSPI_GetStatus()**, **BSP_QSPI_ResumeErase()**, and **BSP_QSPI_SuspendErase()**.

B-L475E-IOT01 BSP User Manual

Defines

QSPI Private Constants

QSPI

Defines

#define	**QSPI_QUAD_DISABLE** 0x0
#define	**QSPI_QUAD_ENABLE** 0x1
#define	**QSPI_HIGH_PERF_DISABLE** 0x0
#define	**QSPI_HIGH_PERF_ENABLE** 0x1

Define Documentation

#define QSPI_HIGH_PERF_DISABLE 0x0

Definition at line 99 of file stm32l475e_iot01_qspi.c.

Referenced by QSPI_HighPerfMode().

#define QSPI_HIGH_PERF_ENABLE 0x1

Definition at line 100 of file stm32l475e_iot01_qspi.c.

Referenced by BSP_QSPI_Init(), and QSPI_HighPerfMode().

#define QSPI_QUAD_DISABLE 0x0

Definition at line 96 of file stm32l475e_iot01_qspi.c.

Referenced by QSPI_QuadMode().

#define QSPI_QUAD_ENABLE 0x1

Definition at line 97 of file stm32l475e_iot01_qspi.c.

Referenced by BSP_QSPI_Init(), and QSPI_QuadMode().

B-L475E-IOT01 BSP User Manual

QSPI Private Variables

QSPI

Variables

QSPI_HandleTypeDef	**QSPIHandle**

Variable Documentation

QSPI_HandleTypeDef QSPIHandle

Definition at line **109** of file **stm32l475e_iot01_qspi.c**.

Referenced by **BSP_QSPI_DeInit()**, **BSP_QSPI_EnableMemoryMappedMode()**, **BSP_QSPI_EnterDeepPowerDown()**, **BSP_QSPI_Erase_Block()**, **BSP_QSPI_Erase_Chip()**, **BSP_QSPI_Erase_Sector()**, **BSP_QSPI_GetStatus()**, **BSP_QSPI_Init()**, **BSP_QSPI_LeaveDeepPowerDown()**, **BSP_QSPI_Read()**, **BSP_QSPI_ResumeErase()**, **BSP_QSPI_SuspendErase()**, **BSP_QSPI_Write()**, **QSPI_AutoPollingMemReady()**, **QSPI_HighPerfMode()**, **QSPI_QuadMode()**, **QSPI_ResetMemory()**, and **QSPI_WriteEnable()**.

B-L475E-IOT01 BSP User Manual

TEMPERATURE
Private Variables

TEMPERATURE

Variables

static TSENSOR_DrvTypeDef * **tsensor_drv**

Variable Documentation

TSENSOR_DrvTypeDef* tsensor_drv [static]

Definition at line **66** of file **stm32l475e_iot01_tsensor.c**.

Referenced by **BSP_TSENSOR_Init()**, and **BSP_TSENSOR_ReadTemp()**.

B-L475E-IOT01 BSP User Manual

TEMPERATURE
Exported Types

TEMPERATURE

Enumerations

enum	**TSENSOR_Status_TypDef** { **TSENSOR_OK** = 0, **TSENSOR_ERROR** } TSENSOR Status. More...

Enumeration Type Documentation

enum TSENSOR_Status_TypDef

TSENSOR Status.

Enumerator:

TSENSOR_OK

TSENSOR_ERROR

Definition at line **83** of file **stm32l475e_iot01_tsensor.h**.

B-L475E-IOT01 BSP User Manual

Drivers Directory Reference

Directories

directory	**BSP**

B-L475E-IOT01 BSP User Manual

BSP Directory Reference

Directories

directory	**B-L475E-IOT01**

B-L475E-IOT01 BSP User Manual

B-L475E-IOT01 Directory Reference

Files

file **stm32l475e_iot01.c** [code]

STM32L475E-IOT01 board support package.

file **stm32l475e_iot01.h** [code]

STM32L475E IOT01 board support package.

file **stm32l475e_iot01_accelero.c** [code]

This file provides a set of functions needed to manage the accelerometer sensor.

file **stm32l475e_iot01_accelero.h** [code]

This file provides a set of functions needed to manage the accelerometer sensor.

file **stm32l475e_iot01_gyro.c** [code]

This file provides a set of functions needed to manage the gyroscope sensor.

file **stm32l475e_iot01_gyro.h** [code]

This file contains definitions for the **stm32l475e_iot01_gyro.c**.

file **stm32l475e_iot01_hsensor.c** [code]

This file provides a set of functions needed to manage the humidity sensor.

file **stm32l475e_iot01_hsensor.h** [code]

This file provides a set of functions needed to manage the humidity sensor.

file **stm32l475e_iot01_magneto.c** [code]

This file provides a set of functions needed to manage the magnetometer sensor.

file **stm32l475e_iot01_magneto.h** [code]

This file provides a set of functions needed to manage the magnetometer sensor.

file **stm32l475e_iot01_psensor.c** [code]

This file provides a set of functions needed to manage the pressure sensor.

file **stm32l475e_iot01_psensor.h** [code]

This file provides a set of functions needed to manage the pressure sensor.

file **stm32l475e_iot01_qspi.c** [code]

This file includes a standard driver for the MX25R6435F QSPI memory mounted on STM32L475E IOT01 board.

file **stm32l475e_iot01_qspi.h** [code]

This file contains the common defines and functions prototypes for the **stm32l475e_iot01_qspi.c** driver.

file **stm32l475e_iot01_tsensor.c** [code]

This file provides a set of functions needed to manage the temperature sensor.

file **stm32l475e_iot01_tsensor.h** [code]

This file provides a set of functions needed to manage the temperature sensor.

B-L475E-IOT01 BSP User Manual

stm32l475e_iot01.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l475e_iot01.h
00004   * @author  MCD Application Team
00005   * @version V1.0.0
00006   * @date    17-March-2017
00007   * @brief   STM32L475E IOT01 board support package
00008   ******************************************************************************
00009   * @attention
00010   *
00011   * <h2><center>&copy; Copyright (c) 2017 STMicroelectronics International N.V.
00012   * All rights reserved.</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without
00015   * modification, are permitted, provided that the following conditions are met:
00016   *
00017   * 1. Redistribution of source code must retain the above copyright notice,
```

```
00018   *    this list of conditions and the follo
wing disclaimer.
00019   * 2. Redistributions in binary form must r
eproduce the above copyright notice,
00020   *    this list of conditions and the follo
wing disclaimer in the documentation
00021   *    and/or other materials provided with
the distribution.
00022   * 3. Neither the name of STMicroelectronic
s nor the names of other
00023   *    contributors to this software may be
used to endorse or promote products
00024   *    derived from this software without sp
ecific written permission.
00025   * 4. This software, including modification
s and/or derivative works of this
00026   *    software, must execute solely and exc
lusively on microcontroller or
00027   *    microprocessor devices manufactured b
y or for STMicroelectronics.
00028   * 5. Redistribution and use of this softwa
re other than as permitted under
00029   *    this license is void and will automat
ically terminate your rights under
00030   *    this license.
00031   *
00032   * THIS SOFTWARE IS PROVIDED BY STMICROELEC
TRONICS AND CONTRIBUTORS "AS IS"
00033   * AND ANY EXPRESS, IMPLIED OR STATUTORY WA
RRANTIES, INCLUDING, BUT NOT
00034   * LIMITED TO, THE IMPLIED WARRANTIES OF ME
RCHANTABILITY, FITNESS FOR A
00035   * PARTICULAR PURPOSE AND NON-INFRINGEMENT
OF THIRD PARTY INTELLECTUAL PROPERTY
00036   * RIGHTS ARE DISCLAIMED TO THE FULLEST EXT
ENT PERMITTED BY LAW. IN NO EVENT
00037   * SHALL STMICROELECTRONICS OR CONTRIBUTORS
```

```
 BE LIABLE FOR ANY DIRECT, INDIRECT,
00038   * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSE
QUENTIAL DAMAGES (INCLUDING, BUT NOT
00039   * LIMITED TO, PROCUREMENT OF SUBSTITUTE GO
ODS OR SERVICES; LOSS OF USE, DATA,
00040   * OR PROFITS; OR BUSINESS INTERRUPTION) HO
WEVER CAUSED AND ON ANY THEORY OF
00041   * LIABILITY, WHETHER IN CONTRACT, STRICT L
IABILITY, OR TORT (INCLUDING
00042   * NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE,
00043   * EVEN IF ADVISED OF THE POSSIBILITY OF SU
CH DAMAGE.
00044   *
00045   ****************************************
**********************************
00046   */
00047
00048 /* Define to prevent recursive inclusion ---
-----------------------------------*/
00049 #ifndef __STM32L475E_IOT01_H
00050 #define __STM32L475E_IOT01_H
00051
00052 #ifdef __cplusplus
00053  extern "C" {
00054 #endif
00055
00056 /* Includes ------------------------------
-----------------------------------*/
00057 #include "stm32l4xx_hal.h"
00058
00059 /** @addtogroup BSP
00060   * @{
00061   */
00062
00063 /** @addtogroup STM32L475E_IOT01
00064   * @{
```

```
00065   */
00066
00067 /** @addtogroup STM32L475E_IOT01_LOW_LEVEL
00068   * @{
00069   */
00070
00071 /** @defgroup STM32L475E_IOT01_LOW_LEVEL_Exp
orted_Types LOW LEVEL Exported Types
00072   * @{
00073   */
00074 typedef enum
00075 {
00076 LED2 = 0,
00077 LED_GREEN = LED2,
00078 }Led_TypeDef;
00079
00080
00081 typedef enum
00082 {
00083   BUTTON_USER = 0
00084 }Button_TypeDef;
00085
00086 typedef enum
00087 {
00088   BUTTON_MODE_GPIO = 0,
00089   BUTTON_MODE_EXTI = 1
00090 }ButtonMode_TypeDef;
00091
00092 typedef enum
00093 {
00094   COM1 = 0,
00095   COM2 = 0,
00096 }COM_TypeDef;
00097 /**
00098   * @}
00099   */
00100
```

```
00101 /** @defgroup STM32L475E_IOT01_LOW_LEVEL_Exp
orted_Constants LOW LEVEL Exported Constants
00102   * @{
00103   */
00104
00105 /**
00106   * @brief  Define for STM32L475E_IOT01 board

00107   */
00108 #if !defined (USE_STM32L475E_IOT01)
00109  #define USE_STM32L475E_IOT01
00110 #endif
00111
00112 #define LEDn                                ((u
int8_t)1)
00113
00114 #define LED2_PIN                            GPI
O_PIN_14
00115 #define LED2_GPIO_PORT                      GPI
OB
00116 #define LED2_GPIO_CLK_ENABLE()              __H
AL_RCC_GPIOB_CLK_ENABLE()
00117 #define LED2_GPIO_CLK_DISABLE()             __H
AL_RCC_GPIOB_CLK_DISABLE()
00118
00119
00120
00121 #define LEDx_GPIO_CLK_ENABLE(__INDEX__)  do{
if((__INDEX__) == 0) LED2_GPIO_CLK_ENABLE();}while
(0)
00122
00123 #define LEDx_GPIO_CLK_DISABLE(__INDEX__)  do
{if((__INDEX__) == 0) LED2_GPIO_CLK_DISABLE();}whi
le(0)
00124
00125 /* Only one User/Wakeup button */
00126 #define BUTTONn
```

```
((uint8_t)1)
00127
00128 /**
00129   * @brief Wakeup push-button
00130   */
00131 #define USER_BUTTON_PIN                    GP
IO_PIN_13
00132 #define USER_BUTTON_GPIO_PORT              GP
IOC
00133 #define USER_BUTTON_GPIO_CLK_ENABLE()      __
HAL_RCC_GPIOC_CLK_ENABLE()
00134 #define USER_BUTTON_GPIO_CLK_DISABLE()     __
HAL_RCC_GPIOC_CLK_DISABLE()
00135 #define USER_BUTTON_EXTI_IRQn              EX
TI15_10_IRQn
00136
00137
00138
00139 #define COMn                               ((
uint8_t)1)
00140
00141 /**
00142  * @brief Definition for COM port1, connecte
d to USART1
00143  */
00144 #define DISCOVERY_COM1
    USART1
00145 #define DISCOVERY_COM1_CLK_ENABLE()
    __HAL_RCC_USART1_CLK_ENABLE()
00146 #define DISCOVERY_COM1_CLK_DISABLE()
    __HAL_RCC_USART1_CLK_DISABLE()
00147
00148 #define DISCOVERY_COM1_TX_PIN
    GPIO_PIN_6
00149 #define DISCOVERY_COM1_TX_GPIO_PORT
    GPIOB
00150 #define DISCOVERY_COM1_TX_GPIO_CLK_ENABLE()
```

```
    __HAL_RCC_GPIOB_CLK_ENABLE()
00151 #define DISCOVERY_COM1_TX_GPIO_CLK_DISABLE()
    __HAL_RCC_GPIOB_CLK_DISABLE()
00152 #define DISCOVERY_COM1_TX_AF
    GPIO_AF7_USART1
00153
00154 #define DISCOVERY_COM1_RX_PIN
    GPIO_PIN_7
00155 #define DISCOVERY_COM1_RX_GPIO_PORT
    GPIOB
00156 #define DISCOVERY_COM1_RX_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOB_CLK_ENABLE()
00157 #define DISCOVERY_COM1_RX_GPIO_CLK_DISABLE()
    __HAL_RCC_GPIOB_CLK_DISABLE()
00158 #define DISCOVERY_COM1_RX_AF
    GPIO_AF7_USART1
00159
00160 #define DISCOVERY_COM1_IRQn
    USART1_IRQn
00161

00162
00163 #define DISCOVERY_COMx_CLK_ENABLE(__INDEX__)
            do { if((__INDEX__) == COM1) {DISCOVER
Y_COM1_CLK_ENABLE();}} while(0)
00164 #define DISCOVERY_COMx_CLK_DISABLE(__INDEX__
)           do { if((__INDEX__) == COM1) {DISCOVER
Y_COM1_CLK_DISABLE();}} while(0)
00165
00166 #define DISCOVERY_COMx_TX_GPIO_CLK_ENABLE(__
INDEX__)    do { if((__INDEX__) == COM1) {DISCOVER
Y_COM1_TX_GPIO_CLK_ENABLE();}} while(0)
00167 #define DISCOVERY_COMx_TX_GPIO_CLK_DISABLE(_
_INDEX__)   do { if((__INDEX__) == COM1) {DISCOVER
Y_COM1_TX_GPIO_CLK_DISABLE();}} while(0)
00168
00169 #define DISCOVERY_COMx_RX_GPIO_CLK_ENABLE(__
```

```
INDEX__)    do { if((__INDEX__) == COM1) {DISCOVER
Y_COM1_RX_GPIO_CLK_ENABLE();}} while(0)
00170 #define DISCOVERY_COMx_RX_GPIO_CLK_DISABLE(_
_INDEX__)   do { if((__INDEX__) == COM1) {DISCOVER
Y_COM1_RX_GPIO_CLK_DISABLE();}} while(0)
00171
00172
00173 /* User can use this section to tailor I2Cx
instance used and associated resources */
00174 /* Definition for I2Cx resources */
00175 #define DISCOVERY_I2Cx
       I2C2
00176 #define DISCOVERY_I2Cx_CLK_ENABLE()
       __HAL_RCC_I2C2_CLK_ENABLE()
00177 #define DISCOVERY_I2Cx_CLK_DISABLE()
       __HAL_RCC_I2C2_CLK_DISABLE()
00178 #define DISCOVERY_DMAx_CLK_ENABLE()
       __HAL_RCC_DMA1_CLK_ENABLE()
00179 #define DISCOVERY_I2Cx_SCL_SDA_GPIO_CLK_ENAB
LE()   __HAL_RCC_GPIOB_CLK_ENABLE()
00180 #define DISCOVERY_I2Cx_SCL_SDA_GPIO_CLK_DISA
BLE()  __HAL_RCC_GPIOB_CLK_DISABLE()
00181
00182 #define DISCOVERY_I2Cx_FORCE_RESET()
       __HAL_RCC_I2C2_FORCE_RESET()
00183 #define DISCOVERY_I2Cx_RELEASE_RESET()
       __HAL_RCC_I2C2_RELEASE_RESET()
00184
00185 /* Definition for I2Cx Pins */
00186 #define DISCOVERY_I2Cx_SCL_PIN
       GPIO_PIN_10
00187 #define DISCOVERY_I2Cx_SDA_PIN
       GPIO_PIN_11

00188 #define DISCOVERY_I2Cx_SCL_SDA_GPIO_PORT
       GPIOB
00189 #define DISCOVERY_I2Cx_SCL_SDA_AF
```

```
       GPIO_AF4_I2C2
00190
00191 /* I2C interrupt requests */
00192 #define DISCOVERY_I2Cx_EV_IRQn
       I2C2_EV_IRQn
00193 #define DISCOVERY_I2Cx_ER_IRQn
       I2C2_ER_IRQn
00194
00195 /* I2C clock speed configuration (in Hz)
00196   WARNING:
00197    Make sure that this define is not already
 declared in other files.
00198    It can be used in parallel by other modul
es. */
00199 #ifndef DISCOVERY_I2C_SPEED
00200  #define DISCOVERY_I2C_SPEED
             100000
00201 #endif /* DISCOVERY_I2C_SPEED */
00202
00203 #ifndef DISCOVERY_I2Cx_TIMING
00204 #define DISCOVERY_I2Cx_TIMING
      ((uint32_t)0x00702681)
00205 #endif /* DISCOVERY_I2Cx_TIMING */
00206
00207
00208 /* I2C Sensors address */
00209 /* LPS22HB (Pressure) I2C Address */
00210 #define LPS22HB_I2C_ADDRESS  (uint8_t)0xBA
00211 /* HTS221 (Humidity) I2C Address */
00212 #define HTS221_I2C_ADDRESS   (uint8_t)0xBE
00213
00214 #ifdef USE_LPS22HB_TEMP
00215 /* LPS22HB Sensor hardware I2C address */
00216 #define TSENSOR_I2C_ADDRESS     LPS22HB_I2C_
ADDRESS
00217 #else /* USE_HTS221_TEMP */
00218 /* HTS221 Sensor hardware I2C address */
```

```
00219 #define TSENSOR_I2C_ADDRESS     HTS221_I2C_A
DDRESS
00220 #endif
00221 /**
00222   * @}
00223   */
00224
00225 /* Exported types ------------------------
-----------------------------------*/
00226 /* Exported constants --------------------
-----------------------------------*/
00227 /* Exported macros -----------------------
-----------------------------------*/
00228 /* Private macros ------------------------
-----------------------------------*/
00229 /* Exported functions --------------------
-----------------------------------*/
00230

00231 /** @defgroup STM32L475E_IOT01_LOW_LEVEL_Exp
orted_Functions LOW LEVEL Exported Functions
00232   * @{
00233   */
00234 uint32_t         BSP_GetVersion(void);
00235 void             BSP_LED_Init(Led_TypeDef Le
d);
00236 void             BSP_LED_DeInit(Led_TypeDef
Led);
00237 void             BSP_LED_On(Led_TypeDef Led)
;
00238 void             BSP_LED_Off(Led_TypeDef Led
);
00239 void             BSP_LED_Toggle(Led_TypeDef
Led);
00240 void             BSP_PB_Init(Button_TypeDef
Button, ButtonMode_TypeDef ButtonMode);
00241 void             BSP_PB_DeInit(Button_TypeDef
```

```
 Button);
00242 uint32_t         BSP_PB_GetState(Button_Type
Def Button);
00243 void             BSP_COM_Init(COM_TypeDef CO
M, UART_HandleTypeDef *husart);
00244 void             BSP_COM_DeInit(COM_TypeDef
COM, UART_HandleTypeDef *huart);
00245 /**
00246   * @}
00247   */
00248
00249 /**
00250   * @}
00251   */
00252
00253 /**
00254   * @}
00255   */
00256
00257 /**
00258   * @}
00259   */
00260 #ifdef __cplusplus
00261 }
00262 #endif
00263
00264 #endif /* __STM32L475E_IOT01_H */
00265
00266 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

B-L475E-IOT01 BSP User Manual

stm32l475e_iot01.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l475e_iot01.c
00004   * @author  MCD Application Team
00005   * @version V1.0.0
00006   * @date    17-March-2017
00007   * @brief   STM32L475E-IOT01 board support package
00008   ******************************************************************************
00009   * @attention
00010   *
00011   * <h2><center>&copy; Copyright (c) 2017 STMicroelectronics International N.V.
00012   * All rights reserved.</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without
00015   * modification, are permitted, provided that the following conditions are met:
00016   *
00017   * 1. Redistribution of source code must retain the above copyright notice,
```

```
00018   *    this list of conditions and the follo
wing disclaimer.
00019   * 2. Redistributions in binary form must r
eproduce the above copyright notice,
00020   *    this list of conditions and the follo
wing disclaimer in the documentation
00021   *    and/or other materials provided with
the distribution.
00022   * 3. Neither the name of STMicroelectronic
s nor the names of other
00023   *    contributors to this software may be
used to endorse or promote products
00024   *    derived from this software without sp
ecific written permission.
00025   * 4. This software, including modification
s and/or derivative works of this
00026   *    software, must execute solely and exc
lusively on microcontroller or
00027   *    microprocessor devices manufactured b
y or for STMicroelectronics.
00028   * 5. Redistribution and use of this softwa
re other than as permitted under
00029   *    this license is void and will automat
ically terminate your rights under
00030   *    this license.
00031   *
00032   * THIS SOFTWARE IS PROVIDED BY STMICROELEC
TRONICS AND CONTRIBUTORS "AS IS"
00033   * AND ANY EXPRESS, IMPLIED OR STATUTORY WA
RRANTIES, INCLUDING, BUT NOT
00034   * LIMITED TO, THE IMPLIED WARRANTIES OF ME
RCHANTABILITY, FITNESS FOR A
00035   * PARTICULAR PURPOSE AND NON-INFRINGEMENT
OF THIRD PARTY INTELLECTUAL PROPERTY
00036   * RIGHTS ARE DISCLAIMED TO THE FULLEST EXT
ENT PERMITTED BY LAW. IN NO EVENT
00037   * SHALL STMICROELECTRONICS OR CONTRIBUTORS
```

```
 BE LIABLE FOR ANY DIRECT, INDIRECT,
00038   * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSE
QUENTIAL DAMAGES (INCLUDING, BUT NOT
00039   * LIMITED TO, PROCUREMENT OF SUBSTITUTE GO
ODS OR SERVICES; LOSS OF USE, DATA,
00040   * OR PROFITS; OR BUSINESS INTERRUPTION) HO
WEVER CAUSED AND ON ANY THEORY OF
00041   * LIABILITY, WHETHER IN CONTRACT, STRICT L
IABILITY, OR TORT (INCLUDING
00042   * NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE,
00043   * EVEN IF ADVISED OF THE POSSIBILITY OF SU
CH DAMAGE.
00044   *
00045   ****************************************
**********************************
00046   */
00047
00048 /* Includes ------------------------------
----------------------------------*/
00049 #include "stm32l475e_iot01.h"
00050 /** @defgroup BSP BSP
00051   * @{
00052   */
00053
00054 /** @defgroup STM32L475E_IOT01 STM32L475E_IO
T01
00055   * @{
00056   */
00057
00058 /** @defgroup STM32L475E_IOT01_LOW_LEVEL LOW
 LEVEL
00059   * @{
00060   */
00061
00062 /** @defgroup STM32L475E_IOT01_LOW_LEVEL_Pri
vate_Defines LOW LEVEL Private Def
```

```
00063   * @{
00064   */
00065 /**
00066  * @brief STM32L475E IOT01 BSP Driver versio
n number V1.0.0
00067    */
00068 #define __STM32L475E_IOT01_BSP_VERSION_MAIN
  (0x01) /*!< [31:24] main version */
00069 #define __STM32L475E_IOT01_BSP_VERSION_SUB1
  (0x00) /*!< [23:16] sub1 version */
00070 #define __STM32L475E_IOT01_BSP_VERSION_SUB2
  (0x00) /*!< [15:8]  sub2 version */
00071 #define __STM32L475E_IOT01_BSP_VERSION_RC
  (0x00) /*!< [7:0]  release candidate */
00072 #define __STM32L475E_IOT01_BSP_VERSION
  ((__STM32L475E_IOT01_BSP_VERSION_MAIN << 24)\
00073
     |(__STM32L475E_IOT01_BSP_VERSION_SUB1 << 16)\
00074
     |(__STM32L475E_IOT01_BSP_VERSION_SUB2 << 8 )\
00075
     |(__STM32L475E_IOT01_BSP_VERSION_RC))
00076 /**
00077   * @}
00078   */
00079
00080 /** @defgroup STM32L475E_IOT01_LOW_LEVEL_Pri
vate_Variables LOW LEVEL Variables
00081   * @{
00082   */
00083
00084 const uint32_t GPIO_PIN[LEDn] = {LED2_PIN};
00085
00086
00087 GPIO_TypeDef* GPIO_PORT[LEDn] = {LED2_GPIO_P
ORT};
00088
```

```
00089
00090 GPIO_TypeDef* BUTTON_PORT[BUTTONn] = {USER_B
UTTON_GPIO_PORT};
00091
00092 const uint16_t BUTTON_PIN[BUTTONn] = {USER_B
UTTON_PIN};
00093
00094 const uint16_t BUTTON_IRQn[BUTTONn] = {USER_
BUTTON_EXTI_IRQn};
00095
00096 USART_TypeDef* COM_USART[COMn] = {DISCOVERY_
COM1};
00097
00098 GPIO_TypeDef* COM_TX_PORT[COMn] = {DISCOVERY
_COM1_TX_GPIO_PORT};
00099
00100 GPIO_TypeDef* COM_RX_PORT[COMn] = {DISCOVERY
_COM1_RX_GPIO_PORT};
00101
00102 const uint16_t COM_TX_PIN[COMn] = {DISCOVERY
_COM1_TX_PIN};
00103
00104 const uint16_t COM_RX_PIN[COMn] = {DISCOVERY
_COM1_RX_PIN};
00105
00106 const uint16_t COM_TX_AF[COMn] = {DISCOVERY_
COM1_TX_AF};
00107
00108 const uint16_t COM_RX_AF[COMn] = {DISCOVERY_
COM1_RX_AF};
00109
00110 I2C_HandleTypeDef hI2cHandler;
00111 UART_HandleTypeDef hDiscoUart;
00112 /**
00113   * @}
00114   */
00115 /** @defgroup STM32L475E_IOT01_LOW_LEVEL_Pri
```

```
vate_FunctionPrototypes LOW LEVEL Private Function
 Prototypes
00116   * @{
00117   */
00118 static void     I2Cx_MspInit(I2C_HandleTypeD
ef *i2c_handler);
00119 static void     I2Cx_MspDeInit(I2C_HandleTyp
eDef *i2c_handler);
00120 static void     I2Cx_Init(I2C_HandleTypeDef
*i2c_handler);
00121 static void     I2Cx_DeInit(I2C_HandleTypeDe
f *i2c_handler);
00122 static HAL_StatusTypeDef I2Cx_ReadMultiple(I
2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint1
6_t Reg, uint16_t MemAddSize, uint8_t *Buffer, uin
t16_t Length);
00123 static HAL_StatusTypeDef I2Cx_WriteMultiple(
I2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint
16_t Reg, uint16_t MemAddSize, uint8_t *Buffer, ui
nt16_t Length);
00124 static HAL_StatusTypeDef I2Cx_IsDeviceReady(
I2C_HandleTypeDef *i2c_handler, uint16_t DevAddres
s, uint32_t Trials);
00125 static void             I2Cx_Error(I2C_Hand
leTypeDef *i2c_handler, uint8_t Addr);
00126
00127 /* Sensors IO functions */
00128 void     SENSOR_IO_Init(void);
00129 void     SENSOR_IO_DeInit(void);
00130 void     SENSOR_IO_Write(uint8_t Addr, uint8
_t Reg, uint8_t Value);
00131 uint8_t  SENSOR_IO_Read(uint8_t Addr, uint8_
t Reg);
00132 uint16_t SENSOR_IO_ReadMultiple(uint8_t Addr
, uint8_t Reg, uint8_t *Buffer, uint16_t Length);
00133 void     SENSOR_IO_WriteMultiple(uint8_t Add
r, uint8_t Reg, uint8_t *Buffer, uint16_t Length);
```

```
00134 HAL_StatusTypeDef SENSOR_IO_IsDeviceReady(ui
nt16_t DevAddress, uint32_t Trials);
00135 void     SENSOR_IO_Delay(uint32_t Delay);
00136
00137 /**
00138   * @}
00139   */
00140
00141 /** @defgroup STM32L475E_IOT01_LOW_LEVEL_Pri
vate_Functions LOW LEVEL Private Functions
00142   * @{
00143   */
00144
00145 /**
00146   * @brief  This method returns the STM32L47
5E IOT01 BSP Driver revision
00147   * @retval version: 0xXYZR (8bits for each
decimal, R for RC)
00148   */
00149 uint32_t BSP_GetVersion(void)
00150 {
00151   return __STM32L475E_IOT01_BSP_VERSION;
00152 }
00153
00154 /**
00155   * @brief  Configures LEDs.
00156   * @param  Led: LED to be configured.
00157   *          This parameter can be one of th
e following values:
00158   *            @arg  LED2
00159   */
00160 void BSP_LED_Init(Led_TypeDef Led)
00161 {
00162   GPIO_InitTypeDef  gpio_init_structure;
00163
00164   LEDx_GPIO_CLK_ENABLE(Led);
00165   /* Configure the GPIO_LED pin */
```

```
00166   gpio_init_structure.Pin   = GPIO_PIN[Led];
00167   gpio_init_structure.Mode  = GPIO_MODE_OUTP
UT_PP;
00168   gpio_init_structure.Pull  = GPIO_NOPULL;
00169   gpio_init_structure.Speed = GPIO_SPEED_FRE
Q_HIGH;
00170
00171   HAL_GPIO_Init(GPIO_PORT[Led], &gpio_init_s
tructure);
00172 }
00173
00174 /**
00175   * @brief  DeInit LEDs.
00176   * @param  Led: LED to be configured.
00177   *          This parameter can be one of th
e following values:
00178   *            @arg  LED2
00179   */
00180 void BSP_LED_DeInit(Led_TypeDef Led)
00181 {
00182   GPIO_InitTypeDef  gpio_init_structure;
00183
00184   /* DeInit the GPIO_LED pin */
00185   gpio_init_structure.Pin = GPIO_PIN[Led];
00186
00187   /* Turn off LED */
00188   HAL_GPIO_WritePin(GPIO_PORT[Led], GPIO_PIN
[Led], GPIO_PIN_RESET);
00189   HAL_GPIO_DeInit(GPIO_PORT[Led], gpio_init_
structure.Pin);
00190 }
00191
00192 /**
00193   * @brief  Turns selected LED On.
00194   * @param  Led: LED to be set on
00195   *          This parameter can be one of th
e following values:
```

```
00196   *            @arg  LED2
00197   */
00198 void BSP_LED_On(Led_TypeDef Led)
00199 {
00200   HAL_GPIO_WritePin(GPIO_PORT[Led], GPIO_PIN
[Led], GPIO_PIN_SET);
00201 }
00202
00203 /**
00204   * @brief  Turns selected LED Off.
00205   * @param  Led: LED to be set off
00206   *          This parameter can be one of th
e following values:
00207   *            @arg  LED2
00208   */
00209 void BSP_LED_Off(Led_TypeDef Led)
00210 {
00211   HAL_GPIO_WritePin(GPIO_PORT[Led], GPIO_PIN
[Led], GPIO_PIN_RESET);
00212 }
00213
00214 /**
00215   * @brief  Toggles the selected LED.
00216   * @param  Led: LED to be toggled
00217   *          This parameter can be one of th
e following values:
00218   *            @arg  LED2
00219   */
00220 void BSP_LED_Toggle(Led_TypeDef Led)
00221 {
00222   HAL_GPIO_TogglePin(GPIO_PORT[Led], GPIO_PIN
[Led]);
00223 }
00224
00225 /**
00226   * @brief  Configures button GPIO and EXTI
Line.
```

```
00227   * @param  Button: Button to be configured
00228   *           This parameter can be one of th
e following values:
00229   *             @arg  BUTTON_WAKEUP: Wakeup P
ush Button
00230   * @param  ButtonMode: Button mode
00231   *           This parameter can be one of th
e following values:
00232   *             @arg  BUTTON_MODE_GPIO: Butto
n will be used as simple IO
00233   *             @arg  BUTTON_MODE_EXTI: Butto
n will be connected to EXTI line
00234   *                                     with
interrupt generation capability
00235   */
00236 void BSP_PB_Init(Button_TypeDef Button, Butt
onMode_TypeDef ButtonMode)
00237 {
00238   GPIO_InitTypeDef gpio_init_structure;
00239
00240   /* Enable the BUTTON clock */
00241   USER_BUTTON_GPIO_CLK_ENABLE();
00242
00243   if(ButtonMode == BUTTON_MODE_GPIO)
00244   {
00245     /* Configure Button pin as input */
00246     gpio_init_structure.Pin = BUTTON_PIN[But
ton];
00247     gpio_init_structure.Mode = GPIO_MODE_INP
UT;
00248     gpio_init_structure.Pull = GPIO_PULLUP;
00249     gpio_init_structure.Speed = GPIO_SPEED_F
REQ_HIGH;
00250     HAL_GPIO_Init(BUTTON_PORT[Button], &gpio
_init_structure);
00251   }
00252
```

```
00253   if(ButtonMode == BUTTON_MODE_EXTI)
00254   {
00255     /* Configure Button pin as input with Ex
ternal interrupt */
00256     gpio_init_structure.Pin = BUTTON_PIN[But
ton];
00257     gpio_init_structure.Pull = GPIO_PULLUP;
00258     gpio_init_structure.Speed = GPIO_SPEED_F
REQ_VERY_HIGH;
00259
00260     gpio_init_structure.Mode = GPIO_MODE_IT_
RISING;
00261
00262     HAL_GPIO_Init(BUTTON_PORT[Button], &gpio
_init_structure);
00263
00264     /* Enable and set Button EXTI Interrupt
to the lowest priority */
00265     HAL_NVIC_SetPriority((IRQn_Type)(BUTTON_
IRQn[Button]), 0x0F, 0x00);
00266     HAL_NVIC_EnableIRQ((IRQn_Type)(BUTTON_IR
Qn[Button]));
00267   }
00268 }
00269
00270 /**
00271   * @brief  Push Button DeInit.
00272   * @param  Button: Button to be configured
00273   *          This parameter can be one of th
e following values:
00274   *            @arg  BUTTON_WAKEUP: Wakeup P
ush Button
00275   * @note PB DeInit does not disable the GPI
O clock
00276   */
00277 void BSP_PB_DeInit(Button_TypeDef Button)
00278 {
```

```
00279   GPIO_InitTypeDef gpio_init_structure;
00280
00281   gpio_init_structure.Pin = BUTTON_PIN[Butto
n];
00282   HAL_NVIC_DisableIRQ((IRQn_Type)(BUTTON_IRQn
[Button]));
00283   HAL_GPIO_DeInit(BUTTON_PORT[Button], gpio_
init_structure.Pin);
00284 }
00285
00286
00287 /**
00288   * @brief  Returns the selected button stat
e.
00289   * @param  Button: Button to be checked
00290   *          This parameter can be one of th
e following values:
00291   *            @arg  BUTTON_WAKEUP: Wakeup P
ush Button
00292   * @retval The Button GPIO pin value (GPIO_
PIN_RESET = button pressed)
00293   */
00294 uint32_t BSP_PB_GetState(Button_TypeDef Butt
on)
00295 {
00296   return HAL_GPIO_ReadPin(BUTTON_PORT[Button
], BUTTON_PIN[Button]);
00297 }
00298
00299 /**
00300   * @brief  Configures COM port.
00301   * @param  COM: COM port to be configured.
00302   *          This parameter can be one of th
e following values:
00303   *            @arg  COM1
00304   * @param  huart: Pointer to a UART_HandleT
ypeDef structure that contains the
```

```
00305   *                configuration information
 for the specified USART peripheral.
00306   */
00307 void BSP_COM_Init(COM_TypeDef COM, UART_Hand
leTypeDef *huart)
00308 {
00309   GPIO_InitTypeDef gpio_init_structure;
00310
00311   /* Enable GPIO clock */
00312   DISCOVERY_COMx_TX_GPIO_CLK_ENABLE(COM);
00313   DISCOVERY_COMx_RX_GPIO_CLK_ENABLE(COM);
00314
00315   /* Enable USART clock */
00316   DISCOVERY_COMx_CLK_ENABLE(COM);
00317
00318   /* Configure USART Tx as alternate functio
n */
00319   gpio_init_structure.Pin = COM_TX_PIN[COM];
00320   gpio_init_structure.Mode = GPIO_MODE_AF_PP
;
00321   gpio_init_structure.Speed = GPIO_SPEED_FRE
Q_HIGH;
00322   gpio_init_structure.Pull = GPIO_NOPULL;
00323   gpio_init_structure.Alternate = COM_TX_AF[
COM];
00324   HAL_GPIO_Init(COM_TX_PORT[COM], &gpio_init
_structure);
00325
00326   /* Configure USART Rx as alternate functio
n */
00327   gpio_init_structure.Pin = COM_RX_PIN[COM];
00328   gpio_init_structure.Mode = GPIO_MODE_AF_PP
;
00329   gpio_init_structure.Alternate = COM_RX_AF[
COM];
00330   HAL_GPIO_Init(COM_RX_PORT[COM], &gpio_init
_structure);
```

```
00331
00332   /* USART configuration */
00333   huart->Instance = COM_USART[COM];
00334   HAL_UART_Init(huart);
00335 }
00336
00337 /**
00338   * @brief  DeInit COM port.
00339   * @param  COM: COM port to be configured.
00340   *          This parameter can be one of th
e following values:
00341   *            @arg  COM1
00342   * @param  huart: Pointer to a UART_HandleT
ypeDef structure that contains the
00343   *                configuration information
 for the specified USART peripheral.
00344   */
00345 void BSP_COM_DeInit(COM_TypeDef COM, UART_Ha
ndleTypeDef *huart)
00346 {
00347   /* USART configuration */
00348   huart->Instance = COM_USART[COM];
00349   HAL_UART_DeInit(huart);
00350
00351   /* Enable USART clock */
00352   DISCOVERY_COMx_CLK_DISABLE(COM);
00353
00354   /* DeInit GPIO pins can be done in the app
lication
00355      (by surcharging this __weak function) */

00356
00357   /* GPIO pins clock, FMC clock and DMA cloc
k can be shut down in the application
00358      by surcharging this __weak function */
00359 }
00360
```

```
00361
00362 /*****************************************************
*********************************
00363                                       BUS OPERATIONS
00364 *****************************************************
*********************************/
00365
00366 /******************************** I2C Routine
s ********************************/
00367 /**
00368   * @brief  Initializes I2C MSP.
00369   * @param  i2c_handler : I2C handler
00370   * @retval None
00371   */
00372 static void I2Cx_MspInit(I2C_HandleTypeDef *
i2c_handler)
00373 {
00374   GPIO_InitTypeDef  gpio_init_structure;
00375
00376   /*** Configure the GPIOs ***/
00377   /* Enable GPIO clock */
00378   DISCOVERY_I2Cx_SCL_SDA_GPIO_CLK_ENABLE();
00379
00380   /* Configure I2C Tx, Rx as alternate funct
ion */
00381   gpio_init_structure.Pin = DISCOVERY_I2Cx_S
CL_PIN | DISCOVERY_I2Cx_SDA_PIN;
00382   gpio_init_structure.Mode = GPIO_MODE_AF_OD
;
00383   gpio_init_structure.Pull = GPIO_PULLUP;
00384   gpio_init_structure.Speed = GPIO_SPEED_FRE
Q_VERY_HIGH;
00385   gpio_init_structure.Alternate = DISCOVERY_
I2Cx_SCL_SDA_AF;
00386   HAL_GPIO_Init(DISCOVERY_I2Cx_SCL_SDA_GPIO_
PORT, &gpio_init_structure);
00387
```

```
00388   HAL_GPIO_Init(DISCOVERY_I2Cx_SCL_SDA_GPIO_
PORT, &gpio_init_structure);
00389
00390   /*** Configure the I2C peripheral ***/
00391   /* Enable I2C clock */
00392   DISCOVERY_I2Cx_CLK_ENABLE();
00393
00394   /* Force the I2C peripheral clock reset */
00395   DISCOVERY_I2Cx_FORCE_RESET();
00396
00397   /* Release the I2C peripheral clock reset
*/
00398   DISCOVERY_I2Cx_RELEASE_RESET();
00399
00400   /* Enable and set I2Cx Interrupt to a lowe
r priority */
00401   HAL_NVIC_SetPriority(DISCOVERY_I2Cx_EV_IRQn
, 0x0F, 0);
00402   HAL_NVIC_EnableIRQ(DISCOVERY_I2Cx_EV_IRQn)
;
00403
00404   /* Enable and set I2Cx Interrupt to a lowe
r priority */
00405   HAL_NVIC_SetPriority(DISCOVERY_I2Cx_ER_IRQn
, 0x0F, 0);
00406   HAL_NVIC_EnableIRQ(DISCOVERY_I2Cx_ER_IRQn)
;
00407 }
00408
00409 /**
00410   * @brief  DeInitializes I2C MSP.
00411   * @param  i2c_handler : I2C handler
00412   * @retval None
00413   */
00414 static void I2Cx_MspDeInit(I2C_HandleTypeDef
 *i2c_handler)
00415 {
```

```
00416   GPIO_InitTypeDef  gpio_init_structure;
00417
00418   /* Configure I2C Tx, Rx as alternate funct
ion */
00419   gpio_init_structure.Pin = DISCOVERY_I2Cx_S
CL_PIN | DISCOVERY_I2Cx_SDA_PIN;
00420   HAL_GPIO_DeInit(DISCOVERY_I2Cx_SCL_SDA_GPI
O_PORT, gpio_init_structure.Pin);
00421   /* Disable GPIO clock */
00422   DISCOVERY_I2Cx_SCL_SDA_GPIO_CLK_DISABLE();
00423
00424   /* Disable I2C clock */
00425   DISCOVERY_I2Cx_CLK_DISABLE();
00426 }
00427
00428 /**
00429   * @brief  Initializes I2C HAL.
00430   * @param  i2c_handler : I2C handler
00431   * @retval None
00432   */
00433 static void I2Cx_Init(I2C_HandleTypeDef *i2c
_handler)
00434 {
00435   /* I2C configuration */
00436   i2c_handler->Instance             = DISCO
VERY_I2Cx;
00437   i2c_handler->Init.Timing          = DISCO
VERY_I2Cx_TIMING;
00438   i2c_handler->Init.OwnAddress1     = 0;
00439   i2c_handler->Init.AddressingMode  = I2C_A
DDRESSINGMODE_7BIT;
00440   i2c_handler->Init.DualAddressMode = I2C_D
UALADDRESS_DISABLE;
00441   i2c_handler->Init.OwnAddress2     = 0;
00442   i2c_handler->Init.GeneralCallMode = I2C_G
ENERALCALL_DISABLE;
00443   i2c_handler->Init.NoStretchMode   = I2C_N
```

```
OSTRETCH_DISABLE;
00444
00445   /* Init the I2C */
00446   I2Cx_MspInit(i2c_handler);
00447   HAL_I2C_Init(i2c_handler);
00448
00449   /**Configure Analogue filter */
00450   HAL_I2CEx_ConfigAnalogFilter(i2c_handler,
I2C_ANALOGFILTER_ENABLE);
00451 }
00452
00453 /**
00454   * @brief  DeInitializes I2C HAL.
00455   * @param  i2c_handler : I2C handler
00456   * @retval None
00457   */
00458 static void I2Cx_DeInit(I2C_HandleTypeDef *i
2c_handler)
00459 {  /* DeInit the I2C */
00460   I2Cx_MspDeInit(i2c_handler);
00461   HAL_I2C_DeInit(i2c_handler);
00462 }
00463
00464 /**
00465   * @brief  Reads multiple data.
00466   * @param  i2c_handler : I2C handler
00467   * @param  Addr: I2C address
00468   * @param  Reg: Reg address
00469   * @param  MemAddress: memory address
00470   * @param  Buffer: Pointer to data buffer
00471   * @param  Length: Length of the data
00472   * @retval HAL status
00473   */
00474 static HAL_StatusTypeDef I2Cx_ReadMultiple(I
2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint1
6_t Reg, uint16_t MemAddress, uint8_t *Buffer, uin
t16_t Length)
```

```
00475 {
00476   HAL_StatusTypeDef status = HAL_OK;
00477
00478   status = HAL_I2C_Mem_Read(i2c_handler, Add
r, (uint16_t)Reg, MemAddress, Buffer, Length, 1000
);
00479
00480   /* Check the communication status */
00481   if(status != HAL_OK)
00482   {
00483     /* I2C error occured */
00484     I2Cx_Error(i2c_handler, Addr);
00485   }
00486   return status;
00487 }
00488
00489
00490 /**
00491   * @brief  Writes a value in a register of
the device through BUS in using DMA mode.
00492   * @param  i2c_handler : I2C handler
00493   * @param  Addr: Device address on BUS Bus.
00494   * @param  Reg: The target register address
 to write
00495   * @param  MemAddress: memory address
00496   * @param  Buffer: The target register valu
e to be written
00497   * @param  Length: buffer size to be written

00498   * @retval HAL status
00499   */
00500 static HAL_StatusTypeDef I2Cx_WriteMultiple(
I2C_HandleTypeDef *i2c_handler, uint8_t Addr, uint
16_t Reg, uint16_t MemAddress, uint8_t *Buffer, ui
nt16_t Length)
00501 {
00502   HAL_StatusTypeDef status = HAL_OK;
```

```
00503
00504   status = HAL_I2C_Mem_Write(i2c_handler, Ad
dr, (uint16_t)Reg, MemAddress, Buffer, Length, 100
0);
00505
00506   /* Check the communication status */
00507   if(status != HAL_OK)
00508   {
00509     /* Re-Initiaize the I2C Bus */
00510     I2Cx_Error(i2c_handler, Addr);
00511   }
00512   return status;
00513 }
00514
00515 /**
00516   * @brief  Checks if target device is ready
 for communication.
00517   * @note   This function is used with Memor
y devices
00518   * @param  i2c_handler : I2C handler
00519   * @param  DevAddress: Target device address

00520   * @param  Trials: Number of trials
00521   * @retval HAL status
00522   */
00523 static HAL_StatusTypeDef I2Cx_IsDeviceReady(
I2C_HandleTypeDef *i2c_handler, uint16_t DevAddres
s, uint32_t Trials)
00524 {
00525   return (HAL_I2C_IsDeviceReady(i2c_handler,
 DevAddress, Trials, 1000));
00526 }
00527
00528 /**
00529   * @brief  Manages error callback by re-ini
tializing I2C.
00530   * @param  i2c_handler : I2C handler
```

```
00531   * @param  Addr: I2C Address
00532   * @retval None
00533   */
00534 static void I2Cx_Error(I2C_HandleTypeDef *i2
c_handler, uint8_t Addr)
00535 {
00536   /* De-initialize the I2C communication bus
 */
00537   HAL_I2C_DeInit(i2c_handler);
00538
00539   /* Re-Initialize the I2C communication bus
 */
00540   I2Cx_Init(i2c_handler);
00541 }
00542
00543 /**
00544   * @}
00545   */
00546
00547 /**********************************************
**********************************
00548                                     LINK OPERATIONS
00549 *********************************************
**********************************/
00550 /********************************* LINK Senso
rs ********************************/
00551
00552 /**
00553   * @brief  Initializes Sensors low level.
00554   * @retval None
00555   */
00556 void SENSOR_IO_Init(void)
00557 {
00558   I2Cx_Init(&hI2cHandler);
00559 }
00560
00561 /**
```

```
00562   * @brief  DeInitializes Sensors low level.
00563   * @retval None
00564   */
00565 void SENSOR_IO_DeInit(void)
00566 {
00567   I2Cx_DeInit(&hI2cHandler);
00568 }
00569
00570 /**
00571   * @brief  Writes a single data.
00572   * @param  Addr: I2C address
00573   * @param  Reg: Reg address
00574   * @param  Value: Data to be written
00575   * @retval None
00576   */
00577 void SENSOR_IO_Write(uint8_t Addr, uint8_t R
eg, uint8_t Value)
00578 {
00579   I2Cx_WriteMultiple(&hI2cHandler, Addr, (ui
nt16_t)Reg, I2C_MEMADD_SIZE_8BIT,(uint8_t*)&Value,
 1);
00580 }
00581
00582 /**
00583   * @brief  Reads a single data.
00584   * @param  Addr: I2C address
00585   * @param  Reg: Reg address
00586   * @retval Data to be read
00587   */
00588 uint8_t SENSOR_IO_Read(uint8_t Addr, uint8_t
 Reg)
00589 {
00590   uint8_t read_value = 0;
00591
00592   I2Cx_ReadMultiple(&hI2cHandler, Addr, Reg,
 I2C_MEMADD_SIZE_8BIT, (uint8_t*)&read_value, 1);
00593
```

```
00594   return read_value;
00595 }
00596
00597 /**
00598   * @brief  Reads multiple data with I2C com
munication
00599   *         channel from TouchScreen.
00600   * @param  Addr: I2C address
00601   * @param  Reg: Register address
00602   * @param  Buffer: Pointer to data buffer
00603   * @param  Length: Length of the data
00604   * @retval HAL status
00605   */
00606 uint16_t SENSOR_IO_ReadMultiple(uint8_t Addr
, uint8_t Reg, uint8_t *Buffer, uint16_t Length)
00607 {
00608  return I2Cx_ReadMultiple(&hI2cHandler, Addr
, (uint16_t)Reg, I2C_MEMADD_SIZE_8BIT, Buffer, Len
gth);
00609 }
00610
00611 /**
00612   * @brief  Writes multiple data with I2C co
mmunication
00613   *         channel from MCU to TouchScreen.
00614   * @param  Addr: I2C address
00615   * @param  Reg: Register address
00616   * @param  Buffer: Pointer to data buffer
00617   * @param  Length: Length of the data
00618   * @retval None
00619   */
00620 void SENSOR_IO_WriteMultiple(uint8_t Addr, u
int8_t Reg, uint8_t *Buffer, uint16_t Length)
00621 {
00622   I2Cx_WriteMultiple(&hI2cHandler, Addr, (ui
nt16_t)Reg, I2C_MEMADD_SIZE_8BIT, Buffer, Length);
00623 }
```

```
00624
00625 /**
00626   * @brief  Checks if target device is ready
 for communication.
00627   * @note   This function is used with Memor
y devices
00628   * @param  DevAddress: Target device address

00629   * @param  Trials: Number of trials
00630   * @retval HAL status
00631   */
00632 HAL_StatusTypeDef SENSOR_IO_IsDeviceReady(ui
nt16_t DevAddress, uint32_t Trials)
00633 {
00634   return (I2Cx_IsDeviceReady(&hI2cHandler, D
evAddress, Trials));
00635 }
00636
00637 /**
00638   * @brief  Delay function used in TouchScre
en low level driver.
00639   * @param  Delay: Delay in ms
00640   * @retval None
00641   */
00642 void SENSOR_IO_Delay(uint32_t Delay)
00643 {
00644   HAL_Delay(Delay);
00645 }
00646
00647 /**
00648   * @}
00649   */
00650
00651 /**
00652   * @}
00653   */
00654
```

```
00655 /**
00656   * @}
00657   */
00658
00659 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

B-L475E-IOT01 BSP User Manual

stm32l475e_iot01_accelero.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l475e_iot01_accelero.h
00004   * @author  MCD Application Team
00005   * @version V1.0.0
00006   * @date    17-March-2017
00007   * @brief   This file provides a set of functions needed to manage the accelerometer sensor
00008   ******************************************************************************
00009   * @attention
00010   *
00011   * <h2><center>&copy; Copyright (c) 2017 STMicroelectronics International N.V.
00012   * All rights reserved.</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without
00015   * modification, are permitted, provided that the following conditions are met:
00016   *
00017   * 1. Redistribution of source code must retain the above copyright notice,
```

```
00018   *    this list of conditions and the follo
wing disclaimer.
00019   * 2. Redistributions in binary form must r
eproduce the above copyright notice,
00020   *    this list of conditions and the follo
wing disclaimer in the documentation
00021   *    and/or other materials provided with
the distribution.
00022   * 3. Neither the name of STMicroelectronic
s nor the names of other
00023   *    contributors to this software may be
used to endorse or promote products
00024   *    derived from this software without sp
ecific written permission.
00025   * 4. This software, including modification
s and/or derivative works of this
00026   *    software, must execute solely and exc
lusively on microcontroller or
00027   *    microprocessor devices manufactured b
y or for STMicroelectronics.
00028   * 5. Redistribution and use of this softwa
re other than as permitted under
00029   *    this license is void and will automat
ically terminate your rights under
00030   *    this license.
00031   *
00032   * THIS SOFTWARE IS PROVIDED BY STMICROELEC
TRONICS AND CONTRIBUTORS "AS IS"
00033   * AND ANY EXPRESS, IMPLIED OR STATUTORY WA
RRANTIES, INCLUDING, BUT NOT
00034   * LIMITED TO, THE IMPLIED WARRANTIES OF ME
RCHANTABILITY, FITNESS FOR A
00035   * PARTICULAR PURPOSE AND NON-INFRINGEMENT
OF THIRD PARTY INTELLECTUAL PROPERTY
00036   * RIGHTS ARE DISCLAIMED TO THE FULLEST EXT
ENT PERMITTED BY LAW. IN NO EVENT
00037   * SHALL STMICROELECTRONICS OR CONTRIBUTORS
```

```
 BE LIABLE FOR ANY DIRECT, INDIRECT,
00038   * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSE
QUENTIAL DAMAGES (INCLUDING, BUT NOT
00039   * LIMITED TO, PROCUREMENT OF SUBSTITUTE GO
ODS OR SERVICES; LOSS OF USE, DATA,
00040   * OR PROFITS; OR BUSINESS INTERRUPTION) HO
WEVER CAUSED AND ON ANY THEORY OF
00041   * LIABILITY, WHETHER IN CONTRACT, STRICT L
IABILITY, OR TORT (INCLUDING
00042   * NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE,
00043   * EVEN IF ADVISED OF THE POSSIBILITY OF SU
CH DAMAGE.
00044   *
00045   ****************************************
**********************************
00046   */
00047
00048 /* Define to prevent recursive inclusion ---
-----------------------------------*/
00049 #ifndef __STM32L475E_IOT01_ACCELERO_H
00050 #define __STM32L475E_IOT01_ACCELERO_H
00051
00052 #ifdef __cplusplus
00053  extern "C" {
00054 #endif
00055
00056 /* Includes ------------------------------
-----------------------------------*/
00057 #include "stm32l475e_iot01.h"
00058 /* Include Accelero component driver */
00059 #include "../Components/lsm6dsl/lsm6dsl.h"
00060
00061 /** @addtogroup BSP
00062   * @{
00063   */
00064
```

```
00065 /** @addtogroup STM32L475E_IOT01
00066   * @{
00067   */
00068
00069 /** @addtogroup STM32L475E_IOT01_ACCELERO
00070   * @{
00071   */
00072
00073 /** @defgroup STM32L475_DISCOVERY_ACCELERO_E
xported_Types ACCELERO Exported Types
00074   * @{
00075   */
00076 typedef enum
00077 {
00078   ACCELERO_OK = 0,
00079   ACCELERO_ERROR = 1,
00080   ACCELERO_TIMEOUT = 2
00081 }
00082 ACCELERO_StatusTypeDef;
00083
00084 /**
00085   * @}
00086   */
00087
00088 /** @defgroup STM32L475E_IOT01_ACCELERO_Expo
rted_Functions ACCELERO Exported Functions
00089   * @{
00090   */
00091 /* Sensor Configuration Functions */
00092 ACCELERO_StatusTypeDef BSP_ACCELERO_Init(void
);
00093 void BSP_ACCELERO_DeInit(void);
00094 void BSP_ACCELERO_LowPower(uint16_t status);
/* 0 Means Disable Low Power Mode, otherwise Low P
ower Mode is enabled */
00095 void BSP_ACCELERO_AccGetXYZ(int16_t *pDataXY
Z);
```

```
00096 /**
00097   * @}
00098   */
00099
00100 /**
00101   * @}
00102   */
00103
00104 /**
00105   * @}
00106   */
00107
00108 /**
00109   * @}
00110   */
00111 #ifdef __cplusplus
00112 }
00113 #endif
00114
00115 #endif /* __STM32L475E_IOT01_ACCELERO_H */
00116
00117 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

B-L475E-IOT01 BSP User Manual

stm32l475e_iot01_accelero.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l475e_iot01_accelero.c
00004   * @author  MCD Application Team
00005   * @version V1.0.0
00006   * @date    17-March-2017
00007   * @brief   This file provides a set of functions needed to manage the accelerometer sensor
00008   ******************************************************************************
00009   * @attention
00010   *
00011   * <h2><center>&copy; Copyright (c) 2017 STMicroelectronics International N.V.
00012   * All rights reserved.</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without
00015   * modification, are permitted, provided that the following conditions are met:
00016   *
00017   * 1. Redistribution of source code must retain the above copyright notice,
```

```
00018   *    this list of conditions and the follo
wing disclaimer.
00019   * 2. Redistributions in binary form must r
eproduce the above copyright notice,
00020   *    this list of conditions and the follo
wing disclaimer in the documentation
00021   *    and/or other materials provided with
the distribution.
00022   * 3. Neither the name of STMicroelectronic
s nor the names of other
00023   *    contributors to this software may be
used to endorse or promote products
00024   *    derived from this software without sp
ecific written permission.
00025   * 4. This software, including modification
s and/or derivative works of this
00026   *    software, must execute solely and exc
lusively on microcontroller or
00027   *    microprocessor devices manufactured b
y or for STMicroelectronics.
00028   * 5. Redistribution and use of this softwa
re other than as permitted under
00029   *    this license is void and will automat
ically terminate your rights under
00030   *    this license.
00031   *
00032   * THIS SOFTWARE IS PROVIDED BY STMICROELEC
TRONICS AND CONTRIBUTORS "AS IS"
00033   * AND ANY EXPRESS, IMPLIED OR STATUTORY WA
RRANTIES, INCLUDING, BUT NOT
00034   * LIMITED TO, THE IMPLIED WARRANTIES OF ME
RCHANTABILITY, FITNESS FOR A
00035   * PARTICULAR PURPOSE AND NON-INFRINGEMENT
OF THIRD PARTY INTELLECTUAL PROPERTY
00036   * RIGHTS ARE DISCLAIMED TO THE FULLEST EXT
ENT PERMITTED BY LAW. IN NO EVENT
00037   * SHALL STMICROELECTRONICS OR CONTRIBUTORS
```

```
 BE LIABLE FOR ANY DIRECT, INDIRECT,
00038   * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSE
QUENTIAL DAMAGES (INCLUDING, BUT NOT
00039   * LIMITED TO, PROCUREMENT OF SUBSTITUTE GO
ODS OR SERVICES; LOSS OF USE, DATA,
00040   * OR PROFITS; OR BUSINESS INTERRUPTION) HO
WEVER CAUSED AND ON ANY THEORY OF
00041   * LIABILITY, WHETHER IN CONTRACT, STRICT L
IABILITY, OR TORT (INCLUDING
00042   * NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE,
00043   * EVEN IF ADVISED OF THE POSSIBILITY OF SU
CH DAMAGE.
00044   *
00045   ****************************************
**********************************
00046   */
00047
00048 /* Includes ------------------------------
----------------------------------*/
00049 #include "stm32l475e_iot01_accelero.h"
00050 /** @addtogroup BSP
00051   * @{
00052   */
00053
00054 /** @addtogroup STM32L475E_IOT01
00055   * @{
00056   */
00057
00058 /** @defgroup STM32L475E_IOT01_ACCELERO ACCE
LERO
00059   * @{
00060   */
00061
00062 /** @defgroup STM32L475E_IOT01_ACCELERO_Priv
ate_Variables ACCELERO Private Variables
00063   * @{
```

```
00064   */
00065 static ACCELERO_DrvTypeDef *AccelerometerDrv
;
00066 /**
00067   * @}
00068   */
00069
00070 /** @defgroup STM32L475E_IOT01_ACCELERO_Priv
ate_Functions ACCELERO Private Functions
00071   * @{
00072   */
00073 /**
00074   * @brief  Initialize the ACCELERO.
00075   * @retval ACCELERO_OK or ACCELERO_ERROR
00076   */
00077 ACCELERO_StatusTypeDef BSP_ACCELERO_Init(void
)
00078 {
00079   ACCELERO_StatusTypeDef ret = ACCELERO_OK;
00080   uint16_t ctrl = 0x0000;
00081   ACCELERO_InitTypeDef LSM6DSL_InitStructure
;
00082
00083   if(Lsm6dslAccDrv.ReadID() != LSM6DSL_ACC_G
YRO_WHO_AM_I)
00084   {
00085     ret = ACCELERO_ERROR;
00086   }
00087   else
00088   {
00089     /* Initialize the ACCELERO accelerometer
 driver structure */
00090     AccelerometerDrv = &Lsm6dslAccDrv;
00091
00092     /* MEMS configuration ----------------
-----------------------------------*/
00093     /* Fill the ACCELERO accelerometer struc
```

```
ture */
00094     LSM6DSL_InitStructure.AccOutput_DataRate
 = LSM6DSL_ODR_52Hz;
00095     LSM6DSL_InitStructure.Axes_Enable = 0;
00096     LSM6DSL_InitStructure.AccFull_Scale = LS
M6DSL_ACC_FULLSCALE_2G;
00097     LSM6DSL_InitStructure.BlockData_Update =
 LSM6DSL_BDU_BLOCK_UPDATE;
00098     LSM6DSL_InitStructure.High_Resolution =
0;
00099     LSM6DSL_InitStructure.Communication_Mode
 = 0;
00100
00101     /* Configure MEMS: data rate, full scale
  */
00102     ctrl =  (LSM6DSL_InitStructure.AccOutput
_DataRate | LSM6DSL_InitStructure.AccFull_Scale);
00103
00104     /* Configure MEMS: BDU and Auto-incremen
t for multi read/write */
00105     ctrl |= ((LSM6DSL_InitStructure.BlockDat
a_Update | LSM6DSL_ACC_GYRO_IF_INC_ENABLED) << 8);
00106
00107     /* Configure the ACCELERO accelerometer
main parameters */
00108     AccelerometerDrv->Init(ctrl);
00109   }
00110
00111   return ret;
00112 }
00113
00114 /**
00115   * @brief  DeInitialize the ACCELERO.
00116   * @retval None.
00117   */
00118 void BSP_ACCELERO_DeInit(void)
00119 {
```

```
00120   /* DeInitialize the accelerometer IO inter
faces */
00121   if(AccelerometerDrv != NULL)
00122   {
00123     if(AccelerometerDrv->DeInit != NULL)
00124     {
00125       AccelerometerDrv->DeInit();
00126     }
00127   }
00128 }
00129
00130 /**
00131   * @brief  Set/Unset the ACCELERO in low po
wer mode.
00132   * @param  status 0 means disable Low Power
 Mode, otherwise Low Power Mode is enabled
00133   * @retval None
00134   */
00135 void BSP_ACCELERO_LowPower(uint16_t status)
00136 {
00137   /* Set/Unset the ACCELERO in low power mod
e */
00138   if(AccelerometerDrv != NULL)
00139   {
00140     if(AccelerometerDrv->LowPower != NULL)
00141     {
00142       AccelerometerDrv->LowPower(status);
00143     }
00144   }
00145 }
00146
00147 /**
00148   * @brief  Get XYZ acceleration values.
00149   * @param  pDataXYZ Pointer on 3 angular ac
celerations table with
00150   *                 pDataXYZ[0] = X axis, p
DataXYZ[1] = Y axis, pDataXYZ[2] = Z axis
```

```
00151   * @retval None
00152   */
00153 void BSP_ACCELERO_AccGetXYZ(int16_t *pDataXY
Z)
00154 {
00155   if(AccelerometerDrv != NULL)
00156   {
00157     if(AccelerometerDrv->GetXYZ != NULL)
00158     {
00159       AccelerometerDrv->GetXYZ(pDataXYZ);
00160     }
00161   }
00162 }
00163 /**
00164   * @}
00165   */
00166
00167 /**
00168   * @}
00169   */
00170
00171 /**
00172   * @}
00173   */
00174
00175 /**
00176   * @}
00177   */
00178
00179 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

B-L475E-IOT01 BSP User Manual

stm32l475e_iot01_gyro.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l475e_iot01_gyro.h
00004   * @author  MCD Application Team
00005   * @version V1.0.0
00006   * @date    17-March-2017
00007   * @brief   This file contains definitions for the stm32l475e_iot01_gyro.c
00008   ******************************************************************************
00009   * @attention
00010   *
00011   * <h2><center>&copy; Copyright (c) 2017 STMicroelectronics International N.V.
00012   * All rights reserved.</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without
00015   * modification, are permitted, provided that the following conditions are met:
00016   *
00017   * 1. Redistribution of source code must retain the above copyright notice,
```

```
00018   *    this list of conditions and the follo
wing disclaimer.
00019   * 2. Redistributions in binary form must r
eproduce the above copyright notice,
00020   *    this list of conditions and the follo
wing disclaimer in the documentation
00021   *    and/or other materials provided with
the distribution.
00022   * 3. Neither the name of STMicroelectronic
s nor the names of other
00023   *    contributors to this software may be
used to endorse or promote products
00024   *    derived from this software without sp
ecific written permission.
00025   * 4. This software, including modification
s and/or derivative works of this
00026   *    software, must execute solely and exc
lusively on microcontroller or
00027   *    microprocessor devices manufactured b
y or for STMicroelectronics.
00028   * 5. Redistribution and use of this softwa
re other than as permitted under
00029   *    this license is void and will automat
ically terminate your rights under
00030   *    this license.
00031   *
00032   * THIS SOFTWARE IS PROVIDED BY STMICROELEC
TRONICS AND CONTRIBUTORS "AS IS"
00033   * AND ANY EXPRESS, IMPLIED OR STATUTORY WA
RRANTIES, INCLUDING, BUT NOT
00034   * LIMITED TO, THE IMPLIED WARRANTIES OF ME
RCHANTABILITY, FITNESS FOR A
00035   * PARTICULAR PURPOSE AND NON-INFRINGEMENT
OF THIRD PARTY INTELLECTUAL PROPERTY
00036   * RIGHTS ARE DISCLAIMED TO THE FULLEST EXT
ENT PERMITTED BY LAW. IN NO EVENT
00037   * SHALL STMICROELECTRONICS OR CONTRIBUTORS
```

```
 BE LIABLE FOR ANY DIRECT, INDIRECT,
00038   * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSE
QUENTIAL DAMAGES (INCLUDING, BUT NOT
00039   * LIMITED TO, PROCUREMENT OF SUBSTITUTE GO
ODS OR SERVICES; LOSS OF USE, DATA,
00040   * OR PROFITS; OR BUSINESS INTERRUPTION) HO
WEVER CAUSED AND ON ANY THEORY OF
00041   * LIABILITY, WHETHER IN CONTRACT, STRICT L
IABILITY, OR TORT (INCLUDING
00042   * NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE,
00043   * EVEN IF ADVISED OF THE POSSIBILITY OF SU
CH DAMAGE.
00044   *
00045   ****************************************
**********************************
00046   */
00047
00048 /* Define to prevent recursive inclusion ---
-----------------------------------*/
00049 #ifndef __STM32L475E_IOT01_GYRO_H
00050 #define __STM32L475E_IOT01_GYRO_H
00051
00052 #ifdef __cplusplus
00053  extern "C" {
00054 #endif
00055
00056 /* Includes ------------------------------
-----------------------------------*/
00057 #include "stm32l475e_iot01.h"
00058 /* Include Gyro component driver */
00059 #include "../Components/lsm6dsl/lsm6dsl.h"

00060
00061 /** @addtogroup BSP
00062   * @{
00063   */
```

```
00064
00065 /** @addtogroup STM32L475E_IOT01
00066   * @{
00067   */
00068
00069 /** @addtogroup STM32L475E_IOT01_GYROSCOPE
00070   * @{
00071   */
00072
00073 /** @defgroup STM32L475_IOT01_GYROSCOPE_Expo
rted_Constants GYROSCOPE Exported Constants
00074   * @{
00075   */
00076 typedef enum
00077 {
00078   GYRO_OK = 0,
00079   GYRO_ERROR = 1,
00080   GYRO_TIMEOUT = 2
00081 }
00082 GYRO_StatusTypeDef;
00083
00084 /**
00085   * @}
00086   */
00087
00088 /** @defgroup STM32L475E_IOT01_GYROSCOPE_Exp
orted_Functions GYROSCOPE Exported Functions
00089   * @{
00090   */
00091 uint8_t BSP_GYRO_Init(void);
00092 void BSP_GYRO_DeInit(void);
00093 void BSP_GYRO_LowPower(uint16_t status);   /
* 0 Means Disable Low Power Mode, otherwise Low Po
wer Mode is enabled */
00094 void BSP_GYRO_GetXYZ(float* pfData);
00095 /**
00096   * @}
```

```
00097   */
00098
00099 /**
00100   * @}
00101   */
00102
00103 /**
00104   * @}
00105   */
00106
00107 /**
00108   * @}
00109   */
00110 #ifdef __cplusplus
00111 }
00112 #endif
00113
00114 #endif /* __STM32L475E_IOT01_GYRO_H */
00115
00116 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

stm32l475e_iot01_gyro.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l475e_iot01_gyro.c
00004   * @author  MCD Application Team
00005   * @version V1.0.0
00006   * @date    17-March-2017
00007   * @brief   This file provides a set of functions needed to manage the gyroscope sensor
00008   ******************************************************************************
00009   * @attention
00010   *
00011   * <h2><center>&copy; Copyright (c) 2017 STMicroelectronics International N.V.
00012   * All rights reserved.</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without
00015   * modification, are permitted, provided that the following conditions are met:
00016   *
00017   * 1. Redistribution of source code must retain the above copyright notice,
```

```
00018   *    this list of conditions and the follo
wing disclaimer.
00019   * 2. Redistributions in binary form must r
eproduce the above copyright notice,
00020   *    this list of conditions and the follo
wing disclaimer in the documentation
00021   *    and/or other materials provided with
the distribution.
00022   * 3. Neither the name of STMicroelectronic
s nor the names of other
00023   *    contributors to this software may be
used to endorse or promote products
00024   *    derived from this software without sp
ecific written permission.
00025   * 4. This software, including modification
s and/or derivative works of this
00026   *    software, must execute solely and exc
lusively on microcontroller or
00027   *    microprocessor devices manufactured b
y or for STMicroelectronics.
00028   * 5. Redistribution and use of this softwa
re other than as permitted under
00029   *    this license is void and will automat
ically terminate your rights under
00030   *    this license.
00031   *
00032   * THIS SOFTWARE IS PROVIDED BY STMICROELEC
TRONICS AND CONTRIBUTORS "AS IS"
00033   * AND ANY EXPRESS, IMPLIED OR STATUTORY WA
RRANTIES, INCLUDING, BUT NOT
00034   * LIMITED TO, THE IMPLIED WARRANTIES OF ME
RCHANTABILITY, FITNESS FOR A
00035   * PARTICULAR PURPOSE AND NON-INFRINGEMENT
OF THIRD PARTY INTELLECTUAL PROPERTY
00036   * RIGHTS ARE DISCLAIMED TO THE FULLEST EXT
ENT PERMITTED BY LAW. IN NO EVENT
00037   * SHALL STMICROELECTRONICS OR CONTRIBUTORS
```

```
 BE LIABLE FOR ANY DIRECT, INDIRECT,
00038   * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSE
QUENTIAL DAMAGES (INCLUDING, BUT NOT
00039   * LIMITED TO, PROCUREMENT OF SUBSTITUTE GO
ODS OR SERVICES; LOSS OF USE, DATA,
00040   * OR PROFITS; OR BUSINESS INTERRUPTION) HO
WEVER CAUSED AND ON ANY THEORY OF
00041   * LIABILITY, WHETHER IN CONTRACT, STRICT L
IABILITY, OR TORT (INCLUDING
00042   * NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE,
00043   * EVEN IF ADVISED OF THE POSSIBILITY OF SU
CH DAMAGE.
00044   *
00045   ****************************************
**********************************
00046   */
00047
00048 /* Includes ------------------------------
----------------------------------*/
00049 #include "stm32l475e_iot01_gyro.h"
00050
00051 /** @addtogroup BSP
00052   * @{
00053   */
00054
00055 /** @addtogroup STM32L475E_IOT01
00056   * @{
00057   */
00058
00059 /** @defgroup STM32L475E_IOT01_GYROSCOPE GYR
OSCOPE
00060   * @{
00061   */
00062
00063 /** @defgroup STM32L475E_IOT01_GYROSCOPE_Pri
vate_Variables GYROSCOPE Private Variables
```

```
00064   * @{
00065   */
00066 static GYRO_DrvTypeDef *GyroscopeDrv;
00067
00068 /**
00069   * @}
00070   */
00071
00072
00073 /** @defgroup STM32L475E_IOT01_GYROSCOPE_Pri
vate_Functions GYROSCOPE Private Functions
00074   * @{
00075   */
00076 /**
00077   * @brief  Initialize Gyroscope.
00078   * @retval GYRO_OK or GYRO_ERROR
00079   */
00080 uint8_t BSP_GYRO_Init(void)
00081 {
00082   uint8_t ret = GYRO_ERROR;
00083   uint16_t ctrl = 0x0000;
00084   GYRO_InitTypeDef LSM6DSL_InitStructure;
00085
00086   if(Lsm6dslGyroDrv.ReadID() != LSM6DSL_ACC_
GYRO_WHO_AM_I)
00087   {
00088     ret = GYRO_ERROR;
00089   }
00090   else
00091   {
00092     /* Initialize the gyroscope driver struc
ture */
00093     GyroscopeDrv = &Lsm6dslGyroDrv;
00094
00095     /* Configure Mems : data rate, power mod
e, full scale and axes */
00096     LSM6DSL_InitStructure.Power_Mode = 0;
```

```
00097     LSM6DSL_InitStructure.Output_DataRate =
LSM6DSL_ODR_52Hz;
00098     LSM6DSL_InitStructure.Axes_Enable = 0;
00099     LSM6DSL_InitStructure.Band_Width = 0;
00100     LSM6DSL_InitStructure.BlockData_Update =
 LSM6DSL_BDU_BLOCK_UPDATE;
00101     LSM6DSL_InitStructure.Endianness = 0;
00102     LSM6DSL_InitStructure.Full_Scale = LSM6D
SL_GYRO_FS_2000;
00103
00104     /* Configure MEMS: data rate, full scale
  */
00105     ctrl = (LSM6DSL_InitStructure.Full_Scale
 | LSM6DSL_InitStructure.Output_DataRate);
00106
00107     /* Configure MEMS: BDU and Auto-incremen
t for multi read/write */
00108     ctrl |= ((LSM6DSL_InitStructure.BlockDat
a_Update | LSM6DSL_ACC_GYRO_IF_INC_ENABLED) << 8);
00109
00110     /* Initialize component */
00111     GyroscopeDrv->Init(ctrl);
00112
00113     ret = GYRO_OK;
00114   }
00115
00116   return ret;
00117 }
00118
00119
00120 /**
00121   * @brief  DeInitialize Gyroscope.
00122   */
00123 void BSP_GYRO_DeInit(void)
00124 {
00125   /* DeInitialize the Gyroscope IO interface
s */
```

```
00126   if(GyroscopeDrv != NULL)
00127   {
00128     if(GyroscopeDrv->DeInit!= NULL)
00129     {
00130       GyroscopeDrv->DeInit();
00131     }
00132   }
00133 }
00134
00135
00136 /**
00137   * @brief  Set/Unset Gyroscope in low power
 mode.
00138   * @param  status 0 means disable Low Power
 Mode, otherwise Low Power Mode is enabled
00139   */
00140 void BSP_GYRO_LowPower(uint16_t status)
00141 {
00142   /* Set/Unset component in low-power mode */

00143   if(GyroscopeDrv != NULL)
00144   {
00145     if(GyroscopeDrv->LowPower!= NULL)
00146     {
00147       GyroscopeDrv->LowPower(status);
00148     }
00149   }
00150 }
00151
00152 /**
00153   * @brief  Get XYZ angular acceleration fro
m the Gyroscope.
00154   * @param  pfData: pointer on floating arra
y
00155   */
00156 void BSP_GYRO_GetXYZ(float* pfData)
00157 {
```

```
00158   if(GyroscopeDrv != NULL)
00159   {
00160     if(GyroscopeDrv->GetXYZ!= NULL)
00161     {
00162       GyroscopeDrv->GetXYZ(pfData);
00163     }
00164   }
00165 }
00166
00167 /**
00168   * @}
00169   */
00170
00171 /**
00172   * @}
00173   */
00174
00175 /**
00176   * @}
00177   */
00178
00179 /**
00180   * @}
00181   */
00182
00183 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

B-L475E-IOT01 BSP User Manual

stm32l475e_iot01_hsensor.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l475e_iot01_hsensor.h
00004   * @author  MCD Application Team
00005   * @version V1.0.0
00006   * @date    17-March-2017
00007   * @brief   This file provides a set of functions needed to manage the humidity sensor
00008   ******************************************************************************
00009   * @attention
00010   *
00011   * <h2><center>&copy; Copyright (c) 2017 STMicroelectronics International N.V.
00012   * All rights reserved.</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without
00015   * modification, are permitted, provided that the following conditions are met:
00016   *
00017   * 1. Redistribution of source code must retain the above copyright notice,
```

```
00018   *    this list of conditions and the follo
wing disclaimer.
00019   * 2. Redistributions in binary form must r
eproduce the above copyright notice,
00020   *    this list of conditions and the follo
wing disclaimer in the documentation
00021   *    and/or other materials provided with
the distribution.
00022   * 3. Neither the name of STMicroelectronic
s nor the names of other
00023   *    contributors to this software may be
used to endorse or promote products
00024   *    derived from this software without sp
ecific written permission.
00025   * 4. This software, including modification
s and/or derivative works of this
00026   *    software, must execute solely and exc
lusively on microcontroller or
00027   *    microprocessor devices manufactured b
y or for STMicroelectronics.
00028   * 5. Redistribution and use of this softwa
re other than as permitted under
00029   *    this license is void and will automat
ically terminate your rights under
00030   *    this license.
00031   *
00032   * THIS SOFTWARE IS PROVIDED BY STMICROELEC
TRONICS AND CONTRIBUTORS "AS IS"
00033   * AND ANY EXPRESS, IMPLIED OR STATUTORY WA
RRANTIES, INCLUDING, BUT NOT
00034   * LIMITED TO, THE IMPLIED WARRANTIES OF ME
RCHANTABILITY, FITNESS FOR A
00035   * PARTICULAR PURPOSE AND NON-INFRINGEMENT
OF THIRD PARTY INTELLECTUAL PROPERTY
00036   * RIGHTS ARE DISCLAIMED TO THE FULLEST EXT
ENT PERMITTED BY LAW. IN NO EVENT
00037   * SHALL STMICROELECTRONICS OR CONTRIBUTORS
```

```
 BE LIABLE FOR ANY DIRECT, INDIRECT,
00038   * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSE
QUENTIAL DAMAGES (INCLUDING, BUT NOT
00039   * LIMITED TO, PROCUREMENT OF SUBSTITUTE GO
ODS OR SERVICES; LOSS OF USE, DATA,
00040   * OR PROFITS; OR BUSINESS INTERRUPTION) HO
WEVER CAUSED AND ON ANY THEORY OF
00041   * LIABILITY, WHETHER IN CONTRACT, STRICT L
IABILITY, OR TORT (INCLUDING
00042   * NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE,
00043   * EVEN IF ADVISED OF THE POSSIBILITY OF SU
CH DAMAGE.
00044   *
00045   ****************************************
**********************************
00046   */
00047
00048 /* Define to prevent recursive inclusion ---
-----------------------------------*/
00049 #ifndef __STM32L475E_IOT01_HSENSOR_H
00050 #define __STM32L475E_IOT01_HSENSOR_H
00051
00052 #ifdef __cplusplus
00053  extern "C" {
00054 #endif
00055
00056 /* Includes ------------------------------
-----------------------------------*/
00057 #include "stm32l475e_iot01.h"
00058 #include "../Components/hts221/hts221.h"
00059
00060 /** @addtogroup BSP
00061   * @{
00062   */
00063
00064 /** @addtogroup STM32L475E_IOT01
```

```
00065   * @{
00066   */
00067
00068 /** @addtogroup STM32L475E_IOT01_HUMIDITY
00069   * @{
00070   */
00071
00072 /* Exported types ------------------------
-----------------------------------*/
00073 /** @defgroup STM32L475E_IOT01_HUMIDITY_Expo
rted_Types HUMIDITY Exported Types
00074   * @{
00075   */
00076
00077 /**
00078   * @brief  HSENSOR Status
00079   */
00080 typedef enum
00081 {
00082   HSENSOR_OK = 0,
00083   HSENSOR_ERROR
00084 }HSENSOR_Status_TypDef;
00085
00086 /**
00087   * @}
00088   */
00089
00090 /** @defgroup STM32L475E_IOT01_HUMIDITY_Expo
rted_Functions HUMIDITY Exported Functions
00091   * @{
00092   */
00093 /* Sensor Configuration Functions */
00094 uint32_t BSP_HSENSOR_Init(void);
00095 uint8_t  BSP_HSENSOR_ReadID(void);
00096 float    BSP_HSENSOR_ReadHumidity(void);
00097 /**
00098   * @}
```

```
00099   */
00100
00101 #ifdef __cplusplus
00102 }
00103 #endif
00104
00105 /**
00106   * @}
00107   */
00108
00109 /**
00110   * @}
00111   */
00112
00113 /**
00114   * @}
00115   */
00116
00117 #endif /* __STM32L475E_IOT01_HSENSOR_H */
00118
00119 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

B-L475E-IOT01 BSP User Manual

stm32l475e_iot01_hsensor.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l475e_iot01_hsensor.c
00004   * @author  MCD Application Team
00005   * @version V1.0.0
00006   * @date    17-March-2017
00007   * @brief   This file provides a set of functions needed to manage the humidity sensor
00008   ******************************************************************************
00009   * @attention
00010   *
00011   * <h2><center>&copy; Copyright (c) 2017 STMicroelectronics International N.V.
00012   * All rights reserved.</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without
00015   * modification, are permitted, provided that the following conditions are met:
00016   *
00017   * 1. Redistribution of source code must retain the above copyright notice,
```

```
00018   *    this list of conditions and the follo
wing disclaimer.
00019   * 2. Redistributions in binary form must r
eproduce the above copyright notice,
00020   *    this list of conditions and the follo
wing disclaimer in the documentation
00021   *    and/or other materials provided with
the distribution.
00022   * 3. Neither the name of STMicroelectronic
s nor the names of other
00023   *    contributors to this software may be
used to endorse or promote products
00024   *    derived from this software without sp
ecific written permission.
00025   * 4. This software, including modification
s and/or derivative works of this
00026   *    software, must execute solely and exc
lusively on microcontroller or
00027   *    microprocessor devices manufactured b
y or for STMicroelectronics.
00028   * 5. Redistribution and use of this softwa
re other than as permitted under
00029   *    this license is void and will automat
ically terminate your rights under
00030   *    this license.
00031   *
00032   * THIS SOFTWARE IS PROVIDED BY STMICROELEC
TRONICS AND CONTRIBUTORS "AS IS"
00033   * AND ANY EXPRESS, IMPLIED OR STATUTORY WA
RRANTIES, INCLUDING, BUT NOT
00034   * LIMITED TO, THE IMPLIED WARRANTIES OF ME
RCHANTABILITY, FITNESS FOR A
00035   * PARTICULAR PURPOSE AND NON-INFRINGEMENT
OF THIRD PARTY INTELLECTUAL PROPERTY
00036   * RIGHTS ARE DISCLAIMED TO THE FULLEST EXT
ENT PERMITTED BY LAW. IN NO EVENT
00037   * SHALL STMICROELECTRONICS OR CONTRIBUTORS
```

```
 BE LIABLE FOR ANY DIRECT, INDIRECT,
00038   * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSE
QUENTIAL DAMAGES (INCLUDING, BUT NOT
00039   * LIMITED TO, PROCUREMENT OF SUBSTITUTE GO
ODS OR SERVICES; LOSS OF USE, DATA,
00040   * OR PROFITS; OR BUSINESS INTERRUPTION) HO
WEVER CAUSED AND ON ANY THEORY OF
00041   * LIABILITY, WHETHER IN CONTRACT, STRICT L
IABILITY, OR TORT (INCLUDING
00042   * NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE,
00043   * EVEN IF ADVISED OF THE POSSIBILITY OF SU
CH DAMAGE.
00044   *
00045   ****************************************
**********************************
00046   */
00047
00048 /* Includes ------------------------------
----------------------------------*/
00049 #include "stm32l475e_iot01_hsensor.h"
00050
00051 /** @addtogroup BSP
00052   * @{
00053   */
00054
00055 /** @addtogroup STM32L475E_IOT01
00056   * @{
00057   */
00058
00059 /** @defgroup STM32L475E_IOT01_HUMIDITY HUMI
DITY
00060   * @{
00061   */
00062
00063 /** @defgroup STM32L475E_IOT01_HUMIDITY_Priv
ate_Variables HUMIDITY Private Variables
```

```
00064   * @{
00065   */
00066 static HSENSOR_DrvTypeDef *Hsensor_drv;
00067 /**
00068   * @}
00069   */
00070
00071 /** @defgroup STM32L475E_IOT01_HUMIDITY_Priv
ate_Functions HUMIDITY Private Functions
00072   * @{
00073   */
00074
00075 /**
00076   * @brief  Initializes peripherals used by
the I2C Humidity Sensor driver.
00077   * @retval HSENSOR status
00078   */
00079 uint32_t BSP_HSENSOR_Init(void)
00080 {
00081   uint32_t ret;
00082
00083   if(HTS221_H_Drv.ReadID(HTS221_I2C_ADDRESS)
 != HTS221_WHO_AM_I_VAL)
00084   {
00085     ret = HSENSOR_ERROR;
00086   }
00087   else
00088   {
00089     Hsensor_drv = &HTS221_H_Drv;
00090     /* HSENSOR Init */
00091     Hsensor_drv->Init(HTS221_I2C_ADDRESS);
00092     ret = HSENSOR_OK;
00093   }
00094
00095   return ret;
00096 }
00097
```

```
00098 /**
00099   * @brief  Read ID of HTS221.
00100   * @retval HTS221 ID value.
00101   */
00102 uint8_t BSP_HSENSOR_ReadID(void)
00103 {
00104   return Hsensor_drv->ReadID(HTS221_I2C_ADDR
ESS);
00105 }
00106
00107 /**
00108   * @brief  Read Humidity register of HTS221.

00109   * @retval HTS221 measured humidity value.
00110   */
00111 float BSP_HSENSOR_ReadHumidity(void)
00112 {
00113   return Hsensor_drv->ReadHumidity(HTS221_I2
C_ADDRESS);
00114 }
00115 /**
00116   * @}
00117   */
00118
00119 /**
00120   * @}
00121   */
00122
00123 /**
00124   * @}
00125   */
00126
00127 /**
00128   * @}
00129   */
00130 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

B-L475E-IOT01 BSP User Manual

stm32l475e_iot01_magneto.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l475e_iot01_magneto.h
00004   * @author  MCD Application Team
00005   * @version V1.0.0
00006   * @date    17-March-2017
00007   * @brief   This file provides a set of functions needed to manage the magnetometer sensor
00008   ******************************************************************************
00009   * @attention
00010   *
00011   * <h2><center>&copy; Copyright (c) 2017 STMicroelectronics International N.V.
00012   * All rights reserved.</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without
00015   * modification, are permitted, provided that the following conditions are met:
00016   *
00017   * 1. Redistribution of source code must retain the above copyright notice,
```

```
00018   *    this list of conditions and the follo
wing disclaimer.
00019   * 2. Redistributions in binary form must r
eproduce the above copyright notice,
00020   *    this list of conditions and the follo
wing disclaimer in the documentation
00021   *    and/or other materials provided with
the distribution.
00022   * 3. Neither the name of STMicroelectronic
s nor the names of other
00023   *    contributors to this software may be
used to endorse or promote products
00024   *    derived from this software without sp
ecific written permission.
00025   * 4. This software, including modification
s and/or derivative works of this
00026   *    software, must execute solely and exc
lusively on microcontroller or
00027   *    microprocessor devices manufactured b
y or for STMicroelectronics.
00028   * 5. Redistribution and use of this softwa
re other than as permitted under
00029   *    this license is void and will automat
ically terminate your rights under
00030   *    this license.
00031   *
00032   * THIS SOFTWARE IS PROVIDED BY STMICROELEC
TRONICS AND CONTRIBUTORS "AS IS"
00033   * AND ANY EXPRESS, IMPLIED OR STATUTORY WA
RRANTIES, INCLUDING, BUT NOT
00034   * LIMITED TO, THE IMPLIED WARRANTIES OF ME
RCHANTABILITY, FITNESS FOR A
00035   * PARTICULAR PURPOSE AND NON-INFRINGEMENT
OF THIRD PARTY INTELLECTUAL PROPERTY
00036   * RIGHTS ARE DISCLAIMED TO THE FULLEST EXT
ENT PERMITTED BY LAW. IN NO EVENT
00037   * SHALL STMICROELECTRONICS OR CONTRIBUTORS
```

```
 BE LIABLE FOR ANY DIRECT, INDIRECT,
00038   * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSE
QUENTIAL DAMAGES (INCLUDING, BUT NOT
00039   * LIMITED TO, PROCUREMENT OF SUBSTITUTE GO
ODS OR SERVICES; LOSS OF USE, DATA,
00040   * OR PROFITS; OR BUSINESS INTERRUPTION) HO
WEVER CAUSED AND ON ANY THEORY OF
00041   * LIABILITY, WHETHER IN CONTRACT, STRICT L
IABILITY, OR TORT (INCLUDING
00042   * NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE,
00043   * EVEN IF ADVISED OF THE POSSIBILITY OF SU
CH DAMAGE.
00044   *
00045   ****************************************
**********************************
00046   */
00047
00048 /* Define to prevent recursive inclusion ---
-----------------------------------*/
00049 #ifndef __STM32L475E_IOT01_MAGNETO_H
00050 #define __STM32L475E_IOT01_MAGNETO_H
00051
00052 #ifdef __cplusplus
00053  extern "C" {
00054 #endif
00055
00056 /* Includes ------------------------------
-----------------------------------*/
00057 #include "stm32l475e_iot01.h"
00058 /* Include Magnetometer component driver */
00059 #include "../Components/lis3mdl/lis3mdl.h"
00060
00061 /** @addtogroup BSP
00062   * @{
00063   */
00064
```

```
00065 /** @addtogroup STM32L475E_IOT01
00066   * @{
00067   */
00068
00069 /** @addtogroup STM32L475E_IOT01_MAGNETO
00070   * @{
00071   */
00072
00073 /** @defgroup STM32L475_IOT01_MAGNETO_Export
ed_Types MAGNETO Exported Types
00074   * @{
00075   */
00076 /* Exported types ------------------------
-----------------------------------*/
00077 typedef enum
00078 {
00079   MAGNETO_OK = 0,
00080   MAGNETO_ERROR = 1,
00081   MAGNETO_TIMEOUT = 2
00082 }
00083 MAGNETO_StatusTypeDef;
00084 /**
00085   * @}
00086   */
00087
00088 /** @defgroup STM32L475E_IOT01_MAGNETO_Expor
ted_Functions MAGNETO Exported Functions
00089   * @{
00090   */
00091 MAGNETO_StatusTypeDef BSP_MAGNETO_Init(void)
;
00092 void BSP_MAGNETO_DeInit(void);
00093 void BSP_MAGNETO_LowPower(uint16_t status);
/* 0 Means Disable Low Power Mode, otherwise Low P
ower Mode is enabled */
00094 void BSP_MAGNETO_GetXYZ(int16_t *pDataXYZ);
00095 /**
```

```
00096   * @}
00097   */
00098
00099 /**
00100   * @}
00101   */
00102
00103 /**
00104   * @}
00105   */
00106
00107 /**
00108   * @}
00109   */
00110
00111 #ifdef __cplusplus
00112 }
00113 #endif
00114
00115 #endif /* __STM32L475E_IOT01_MAGNETO_H */
00116
00117 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

B-L475E-IOT01 BSP User Manual

stm32l475e_iot01_magneto.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l475e_iot01_magneto.c
00004   * @author  MCD Application Team
00005   * @version V1.0.0
00006   * @date    17-March-2017
00007   * @brief   This file provides a set of functions needed to manage the magnetometer sensor
00008   ******************************************************************************
00009   * @attention
00010   *
00011   * <h2><center>&copy; Copyright (c) 2017 STMicroelectronics International N.V.
00012   * All rights reserved.</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without
00015   * modification, are permitted, provided that the following conditions are met:
00016   *
00017   * 1. Redistribution of source code must retain the above copyright notice,
```

```
00018   *    this list of conditions and the follo
wing disclaimer.
00019   * 2. Redistributions in binary form must r
eproduce the above copyright notice,
00020   *    this list of conditions and the follo
wing disclaimer in the documentation
00021   *    and/or other materials provided with
the distribution.
00022   * 3. Neither the name of STMicroelectronic
s nor the names of other
00023   *    contributors to this software may be
used to endorse or promote products
00024   *    derived from this software without sp
ecific written permission.
00025   * 4. This software, including modification
s and/or derivative works of this
00026   *    software, must execute solely and exc
lusively on microcontroller or
00027   *    microprocessor devices manufactured b
y or for STMicroelectronics.
00028   * 5. Redistribution and use of this softwa
re other than as permitted under
00029   *    this license is void and will automat
ically terminate your rights under
00030   *    this license.
00031   *
00032   * THIS SOFTWARE IS PROVIDED BY STMICROELEC
TRONICS AND CONTRIBUTORS "AS IS"
00033   * AND ANY EXPRESS, IMPLIED OR STATUTORY WA
RRANTIES, INCLUDING, BUT NOT
00034   * LIMITED TO, THE IMPLIED WARRANTIES OF ME
RCHANTABILITY, FITNESS FOR A
00035   * PARTICULAR PURPOSE AND NON-INFRINGEMENT
OF THIRD PARTY INTELLECTUAL PROPERTY
00036   * RIGHTS ARE DISCLAIMED TO THE FULLEST EXT
ENT PERMITTED BY LAW. IN NO EVENT
00037   * SHALL STMICROELECTRONICS OR CONTRIBUTORS
```

```
 BE LIABLE FOR ANY DIRECT, INDIRECT,
00038   * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSE
QUENTIAL DAMAGES (INCLUDING, BUT NOT
00039   * LIMITED TO, PROCUREMENT OF SUBSTITUTE GO
ODS OR SERVICES; LOSS OF USE, DATA,
00040   * OR PROFITS; OR BUSINESS INTERRUPTION) HO
WEVER CAUSED AND ON ANY THEORY OF
00041   * LIABILITY, WHETHER IN CONTRACT, STRICT L
IABILITY, OR TORT (INCLUDING
00042   * NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE,
00043   * EVEN IF ADVISED OF THE POSSIBILITY OF SU
CH DAMAGE.
00044   *
00045   ****************************************
**********************************
00046   */
00047
00048 /* Includes ------------------------------
----------------------------------*/
00049 #include "stm32l475e_iot01_magneto.h"
00050
00051 /** @addtogroup BSP
00052   * @{
00053   */
00054
00055 /** @addtogroup STM32L475E_IOT01
00056   * @{
00057   */
00058
00059 /** @defgroup STM32L475E_IOT01_MAGNETO MAGNE
TO
00060   * @{
00061   */
00062
00063 /** @defgroup STM32L475E_IOT01_MAGNETO_Priva
te_Variables MAGNETO Private Variables
```

```
00064   * @{
00065   */
00066 static MAGNETO_DrvTypeDef  *MagnetoDrv;
00067 /**
00068   * @}
00069   */
00070
00071
00072 /** @defgroup STM32L475E_IOT01_MAGNETO_Priva
te_Functions MAGNETO Private Functions
00073   * @{
00074   */
00075
00076 /**
00077  * @brief Initialize a magnetometer sensor
00078  * @retval COMPONENT_ERROR in case of failure

00079  */
00080 MAGNETO_StatusTypeDef BSP_MAGNETO_Init(void)
00081 {
00082   MAGNETO_StatusTypeDef ret = MAGNETO_OK;
00083   MAGNETO_InitTypeDef LIS3MDL_InitStructureM
ag;
00084
00085   if(Lis3mdlMagDrv.ReadID() != I_AM_LIS3MDL)
00086   {
00087     ret = MAGNETO_ERROR;
00088   }
00089   else
00090   {
00091     /* Initialize the MAGNETO magnetometer d
river structure */
00092     MagnetoDrv = &Lis3mdlMagDrv;
00093
00094     /* MEMS configuration ----------------
-----------------------------------*/
00095     /* Fill the MAGNETO magnetometer structu
```

```
re */
00096     LIS3MDL_InitStructureMag.Register1 = LIS
3MDL_MAG_TEMPSENSOR_DISABLE | LIS3MDL_MAG_OM_XY_HI
GH | LIS3MDL_MAG_ODR_40_HZ;
00097     LIS3MDL_InitStructureMag.Register2 = LIS
3MDL_MAG_FS_4_GA | LIS3MDL_MAG_REBOOT_DEFAULT | LI
S3MDL_MAG_SOFT_RESET_DEFAULT;
00098     LIS3MDL_InitStructureMag.Register3 = LIS
3MDL_MAG_CONFIG_NORMAL_MODE | LIS3MDL_MAG_CONTINUO
US_MODE;
00099     LIS3MDL_InitStructureMag.Register4 = LIS
3MDL_MAG_OM_Z_HIGH | LIS3MDL_MAG_BLE_LSB;
00100     LIS3MDL_InitStructureMag.Register5 = LIS
3MDL_MAG_BDU_MSBLSB;
00101     /* Configure the MAGNETO magnetometer ma
in parameters */
00102     MagnetoDrv->Init(LIS3MDL_InitStructureMa
g);
00103   }
00104
00105   return ret;
00106 }
00107
00108 /**
00109   * @brief  DeInitialize the MAGNETO.
00110   */
00111 void BSP_MAGNETO_DeInit(void)
00112 {
00113   /* DeInitialize the  magnetometer IO inter
faces */
00114   if(MagnetoDrv != NULL)
00115   {
00116     if(MagnetoDrv->DeInit != NULL)
00117     {
00118       MagnetoDrv->DeInit();
00119     }
00120   }
```

```
00121 }
00122
00123 /**
00124   * @brief  Set/Unset the MAGNETO in low pow
er mode.
00125   */
00126 void BSP_MAGNETO_LowPower(uint16_t status)
00127 {
00128   /* Put the magnetometer in low power mode
*/
00129   if(MagnetoDrv != NULL)
00130   {
00131     if(MagnetoDrv->LowPower != NULL)
00132     {
00133       MagnetoDrv->LowPower(status);
00134     }
00135   }
00136 }
00137
00138 /**
00139   * @brief  Get XYZ magnetometer values.
00140   * @param  pDataXYZ Pointer on 3 magnetomet
er values table with
00141   *                  pDataXYZ[0] = X axis, p
DataXYZ[1] = Y axis, pDataXYZ[2] = Z axis
00142   */
00143 void BSP_MAGNETO_GetXYZ(int16_t *pDataXYZ)
00144 {
00145   if(MagnetoDrv != NULL)
00146   {
00147     if(MagnetoDrv->GetXYZ != NULL)
00148     {
00149       MagnetoDrv->GetXYZ(pDataXYZ);
00150     }
00151   }
00152 }
00153
```

```
00154 /**
00155   * @}
00156   */
00157
00158 /**
00159   * @}
00160   */
00161
00162 /**
00163   * @}
00164   */
00165
00166 /**
00167   * @}
00168   */
00169 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

stm32l475e_iot01_psensor.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l475e_iot01_psensor.h
00004   * @author  MCD Application Team
00005   * @version V1.0.0
00006   * @date    17-March-2017
00007   * @brief   This file provides a set of functions needed to manage the pressure sensor
00008   ******************************************************************************
00009   * @attention
00010   *
00011   * <h2><center>&copy; Copyright (c) 2017 STMicroelectronics International N.V.
00012   * All rights reserved.</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without
00015   * modification, are permitted, provided that the following conditions are met:
00016   *
00017   * 1. Redistribution of source code must retain the above copyright notice,
```

```
00018   *    this list of conditions and the follo
wing disclaimer.
00019   * 2. Redistributions in binary form must r
eproduce the above copyright notice,
00020   *    this list of conditions and the follo
wing disclaimer in the documentation
00021   *    and/or other materials provided with
the distribution.
00022   * 3. Neither the name of STMicroelectronic
s nor the names of other
00023   *    contributors to this software may be
used to endorse or promote products
00024   *    derived from this software without sp
ecific written permission.
00025   * 4. This software, including modification
s and/or derivative works of this
00026   *    software, must execute solely and exc
lusively on microcontroller or
00027   *    microprocessor devices manufactured b
y or for STMicroelectronics.
00028   * 5. Redistribution and use of this softwa
re other than as permitted under
00029   *    this license is void and will automat
ically terminate your rights under
00030   *    this license.
00031   *
00032   * THIS SOFTWARE IS PROVIDED BY STMICROELEC
TRONICS AND CONTRIBUTORS "AS IS"
00033   * AND ANY EXPRESS, IMPLIED OR STATUTORY WA
RRANTIES, INCLUDING, BUT NOT
00034   * LIMITED TO, THE IMPLIED WARRANTIES OF ME
RCHANTABILITY, FITNESS FOR A
00035   * PARTICULAR PURPOSE AND NON-INFRINGEMENT
OF THIRD PARTY INTELLECTUAL PROPERTY
00036   * RIGHTS ARE DISCLAIMED TO THE FULLEST EXT
ENT PERMITTED BY LAW. IN NO EVENT
00037   * SHALL STMICROELECTRONICS OR CONTRIBUTORS
```

```
 BE LIABLE FOR ANY DIRECT, INDIRECT,
00038   * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSE
QUENTIAL DAMAGES (INCLUDING, BUT NOT
00039   * LIMITED TO, PROCUREMENT OF SUBSTITUTE GO
ODS OR SERVICES; LOSS OF USE, DATA,
00040   * OR PROFITS; OR BUSINESS INTERRUPTION) HO
WEVER CAUSED AND ON ANY THEORY OF
00041   * LIABILITY, WHETHER IN CONTRACT, STRICT L
IABILITY, OR TORT (INCLUDING
00042   * NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE,
00043   * EVEN IF ADVISED OF THE POSSIBILITY OF SU
CH DAMAGE.
00044   *
00045   ****************************************
**********************************
00046   */
00047
00048 /* Define to prevent recursive inclusion ---
-----------------------------------*/
00049 #ifndef __STM32L475E_IOT01_PSENSOR_H
00050 #define __STM32L475E_IOT01_PSENSOR_H
00051
00052 #ifdef __cplusplus
00053  extern "C" {
00054 #endif
00055
00056 /* Includes ------------------------------
-----------------------------------*/
00057 #include "stm32l475e_iot01.h"
00058 #include "../Components/lps22hb/lps22hb.h"
00059
00060 /** @addtogroup BSP
00061   * @{
00062   */
00063
00064 /** @addtogroup STM32L475E_IOT01
```

```
00065   * @{
00066   */
00067
00068 /** @addtogroup STM32L475E_IOT01_PRESSURE
00069   * @{
00070   */
00071
00072 /* Exported types ------------------------
-----------------------------------*/
00073 /** @defgroup STM32L475E_IOT01_PRESSURE_Expo
rted_Types PRESSURE Exported Types
00074   * @{
00075   */
00076
00077 /**
00078   * @brief  PSENSOR Status
00079   */
00080 typedef enum
00081 {
00082   PSENSOR_OK = 0,
00083   PSENSOR_ERROR
00084 }PSENSOR_Status_TypDef;
00085
00086 /**
00087   * @}
00088   */
00089
00090 /** @defgroup STM32L475E_IOT01_PRESSURE_Expo
rted_Functions PRESSURE Exported Functions
00091   * @{
00092   */
00093 /* Sensor Configuration Functions */
00094 uint32_t BSP_PSENSOR_Init(void);
00095 uint8_t  BSP_PSENSOR_ReadID(void);
00096 float    BSP_PSENSOR_ReadPressure(void);
00097 /**
00098   * @}
```

```
00099   */
00100
00101 /**
00102   * @}
00103   */
00104
00105 /**
00106   * @}
00107   */
00108
00109 /**
00110   * @}
00111   */
00112
00113 #ifdef __cplusplus
00114 }
00115 #endif
00116
00117 #endif /* __STM32L475E_IOT01_PSENSOR_H */
00118
00119 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

B-L475E-IOT01 BSP User Manual

stm32l475e_iot01_psensor.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l475e_iot01_psensor.c
00004   * @author  MCD Application Team
00005   * @version V1.0.0
00006   * @date    17-March-2017
00007   * @brief   This file provides a set of functions needed to manage the pressure sensor
00008   ******************************************************************************
00009   * @attention
00010   *
00011   * <h2><center>&copy; Copyright (c) 2017 STMicroelectronics International N.V.
00012   * All rights reserved.</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without
00015   * modification, are permitted, provided that the following conditions are met:
00016   *
00017   * 1. Redistribution of source code must retain the above copyright notice,
```

```
00018   *    this list of conditions and the follo
wing disclaimer.
00019   * 2. Redistributions in binary form must r
eproduce the above copyright notice,
00020   *    this list of conditions and the follo
wing disclaimer in the documentation
00021   *    and/or other materials provided with
the distribution.
00022   * 3. Neither the name of STMicroelectronic
s nor the names of other
00023   *    contributors to this software may be
used to endorse or promote products
00024   *    derived from this software without sp
ecific written permission.
00025   * 4. This software, including modification
s and/or derivative works of this
00026   *    software, must execute solely and exc
lusively on microcontroller or
00027   *    microprocessor devices manufactured b
y or for STMicroelectronics.
00028   * 5. Redistribution and use of this softwa
re other than as permitted under
00029   *    this license is void and will automat
ically terminate your rights under
00030   *    this license.
00031   *
00032   * THIS SOFTWARE IS PROVIDED BY STMICROELEC
TRONICS AND CONTRIBUTORS "AS IS"
00033   * AND ANY EXPRESS, IMPLIED OR STATUTORY WA
RRANTIES, INCLUDING, BUT NOT
00034   * LIMITED TO, THE IMPLIED WARRANTIES OF ME
RCHANTABILITY, FITNESS FOR A
00035   * PARTICULAR PURPOSE AND NON-INFRINGEMENT
OF THIRD PARTY INTELLECTUAL PROPERTY
00036   * RIGHTS ARE DISCLAIMED TO THE FULLEST EXT
ENT PERMITTED BY LAW. IN NO EVENT
00037   * SHALL STMICROELECTRONICS OR CONTRIBUTORS
```

```
 BE LIABLE FOR ANY DIRECT, INDIRECT,
00038   * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSE
QUENTIAL DAMAGES (INCLUDING, BUT NOT
00039   * LIMITED TO, PROCUREMENT OF SUBSTITUTE GO
ODS OR SERVICES; LOSS OF USE, DATA,
00040   * OR PROFITS; OR BUSINESS INTERRUPTION) HO
WEVER CAUSED AND ON ANY THEORY OF
00041   * LIABILITY, WHETHER IN CONTRACT, STRICT L
IABILITY, OR TORT (INCLUDING
00042   * NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE,
00043   * EVEN IF ADVISED OF THE POSSIBILITY OF SU
CH DAMAGE.
00044   *
00045   ****************************************
**********************************
00046   */
00047
00048 /* Includes ------------------------------
----------------------------------*/
00049 #include "stm32l475e_iot01_psensor.h"
00050
00051 /** @addtogroup BSP
00052   * @{
00053   */
00054
00055 /** @addtogroup STM32L475E_IOT01
00056   * @{
00057   */
00058
00059 /** @defgroup STM32L475E_IOT01_PRESSURE PRES
SURE
00060   * @{
00061   */
00062
00063 /** @defgroup STM32L475E_IOT01_PRESSURE_Priv
ate_Variables PRESSURE Private Variables
```

```
00064   * @{
00065   */
00066 static PSENSOR_DrvTypeDef *Psensor_drv;
00067 /**
00068   * @}
00069   */
00070
00071 /** @defgroup STM32L475E_IOT01_PRESSURE_Priv
ate_Functions PRESSURE Private Functions
00072   * @{
00073   */
00074
00075 /**
00076   * @brief  Initializes peripherals used by
the I2C Pressure Sensor driver.
00077   * @retval PSENSOR status
00078   */
00079 uint32_t BSP_PSENSOR_Init(void)
00080 {
00081   uint32_t ret;
00082
00083   if(LPS22HB_P_Drv.ReadID(LPS22HB_I2C_ADDRESS
) != LPS22HB_WHO_AM_I_VAL)
00084   {
00085     ret = PSENSOR_ERROR;
00086   }
00087   else
00088   {
00089      Psensor_drv = &LPS22HB_P_Drv;
00090
00091     /* PSENSOR Init */
00092     Psensor_drv->Init(LPS22HB_I2C_ADDRESS);
00093     ret = PSENSOR_OK;
00094   }
00095
00096   return ret;
00097 }
```

```
00098
00099 /**
00100   * @brief  Read ID of LPS22HB.
00101   * @retval LPS22HB ID value.
00102   */
00103 uint8_t BSP_PSENSOR_ReadID(void)
00104 {
00105   return Psensor_drv->ReadID(LPS22HB_I2C_ADD
RESS);
00106 }
00107
00108 /**
00109   * @brief  Read Pressure register of LPS22H
B.
00110   * @retval LPS22HB measured pressure value.
00111   */
00112 float BSP_PSENSOR_ReadPressure(void)
00113 {
00114   return Psensor_drv->ReadPressure(LPS22HB_I
2C_ADDRESS);
00115 }
00116 /**
00117   * @}
00118   */
00119
00120 /**
00121   * @}
00122   */
00123
00124 /**
00125   * @}
00126   */
00127
00128 /**
00129   * @}
00130   */
00131 /************************ (C) COPYRIGHT STMi
```

```
croelectronics *****END OF FILE****/
```

stm32l475e_iot01_tsensor.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l475e_iot01_tsensor.h
00004   * @author  MCD Application Team
00005   * @version V1.0.0
00006   * @date    17-March-2017
00007   * @brief   This file provides a set of functions needed to manage the temperature sensor
00008   ******************************************************************************
00009   * @attention
00010   *
00011   * <h2><center>&copy; Copyright (c) 2017 STMicroelectronics International N.V.
00012   * All rights reserved.</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without
00015   * modification, are permitted, provided that the following conditions are met:
00016   *
00017   * 1. Redistribution of source code must retain the above copyright notice,
```

```
00018   *    this list of conditions and the follo
wing disclaimer.
00019   * 2. Redistributions in binary form must r
eproduce the above copyright notice,
00020   *    this list of conditions and the follo
wing disclaimer in the documentation
00021   *    and/or other materials provided with
the distribution.
00022   * 3. Neither the name of STMicroelectronic
s nor the names of other
00023   *    contributors to this software may be
used to endorse or promote products
00024   *    derived from this software without sp
ecific written permission.
00025   * 4. This software, including modification
s and/or derivative works of this
00026   *    software, must execute solely and exc
lusively on microcontroller or
00027   *    microprocessor devices manufactured b
y or for STMicroelectronics.
00028   * 5. Redistribution and use of this softwa
re other than as permitted under
00029   *    this license is void and will automat
ically terminate your rights under
00030   *    this license.
00031   *
00032   * THIS SOFTWARE IS PROVIDED BY STMICROELEC
TRONICS AND CONTRIBUTORS "AS IS"
00033   * AND ANY EXPRESS, IMPLIED OR STATUTORY WA
RRANTIES, INCLUDING, BUT NOT
00034   * LIMITED TO, THE IMPLIED WARRANTIES OF ME
RCHANTABILITY, FITNESS FOR A
00035   * PARTICULAR PURPOSE AND NON-INFRINGEMENT
OF THIRD PARTY INTELLECTUAL PROPERTY
00036   * RIGHTS ARE DISCLAIMED TO THE FULLEST EXT
ENT PERMITTED BY LAW. IN NO EVENT
00037   * SHALL STMICROELECTRONICS OR CONTRIBUTORS
```

```
 BE LIABLE FOR ANY DIRECT, INDIRECT,
00038   * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSE
QUENTIAL DAMAGES (INCLUDING, BUT NOT
00039   * LIMITED TO, PROCUREMENT OF SUBSTITUTE GO
ODS OR SERVICES; LOSS OF USE, DATA,
00040   * OR PROFITS; OR BUSINESS INTERRUPTION) HO
WEVER CAUSED AND ON ANY THEORY OF
00041   * LIABILITY, WHETHER IN CONTRACT, STRICT L
IABILITY, OR TORT (INCLUDING
00042   * NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE,
00043   * EVEN IF ADVISED OF THE POSSIBILITY OF SU
CH DAMAGE.
00044   *
00045   ****************************************
**********************************
00046   */
00047
00048 /* Define to prevent recursive inclusion ---
-----------------------------------*/
00049 #ifndef __STM32L475E_IOT01_TSENSOR_H
00050 #define __STM32L475E_IOT01_TSENSOR_H
00051
00052 #ifdef __cplusplus
00053  extern "C" {
00054 #endif
00055
00056 /* Includes ------------------------------
-----------------------------------*/
00057 #include "stm32l475e_iot01.h"
00058 #ifdef USE_LPS22HB_TEMP
00059 #include "../Components/lps22hb/lps22hb.h"
00060 #else /* USE_HTS221_TEMP */
00061 #include "../Components/hts221/hts221.h"
00062 #endif
00063
00064 /** @addtogroup BSP
```

```
00065   * @{
00066   */
00067
00068 /** @addtogroup STM32L475E_IOT01
00069   * @{
00070   */
00071
00072 /** @addtogroup STM32L475E_IOT01_TEMPERATURE
00073   * @{
00074   */
00075
00076 /** @defgroup STM32L475E_IOT01_TEMPERATURE_E
xported_Types TEMPERATURE Exported Types
00077   * @{
00078   */
00079
00080 /**
00081   * @brief  TSENSOR Status
00082   */
00083 typedef enum
00084 {
00085   TSENSOR_OK = 0,
00086   TSENSOR_ERROR
00087 }TSENSOR_Status_TypDef;
00088
00089 /**
00090   * @}
00091   */
00092
00093
00094 /** @defgroup STM32L475E_IOT01_TEMPERATURE_E
xported_Functions TEMPERATURE Exported Constants
00095   * @{
00096   */
00097 /* Exported macros ------------------------
-----------------------------------*/
```

```
00098 /* Private macros ------------------------
-----------------------------------*/
00099 /* Exported functions ---------------------
-----------------------------------*/
00100 /* Sensor Configuration Functions */
00101 uint32_t BSP_TSENSOR_Init(void);
00102 float BSP_TSENSOR_ReadTemp(void);
00103 /**
00104   * @}
00105   */
00106
00107 #ifdef __cplusplus
00108 }
00109 #endif
00110
00111 /**
00112   * @}
00113   */
00114
00115 /**
00116   * @}
00117   */
00118
00119 /**
00120   * @}
00121   */
00122
00123 #endif /* __STM32L475E_IOT01_TSENSOR_H */
00124
00125 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

stm32l475e_iot01_tsensor.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32l475e_iot01_tsensor.c
00004   * @author  MCD Application Team
00005   * @version V1.0.0
00006   * @date    17-March-2017
00007   * @brief   This file provides a set of functions needed to manage the temperature sensor
00008   ******************************************************************************
00009   * @attention
00010   *
00011   * <h2><center>&copy; Copyright (c) 2017 STMicroelectronics International N.V.
00012   * All rights reserved.</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without
00015   * modification, are permitted, provided that the following conditions are met:
00016   *
00017   * 1. Redistribution of source code must retain the above copyright notice,
```

```
00018   *    this list of conditions and the follo
wing disclaimer.
00019   * 2. Redistributions in binary form must r
eproduce the above copyright notice,
00020   *    this list of conditions and the follo
wing disclaimer in the documentation
00021   *    and/or other materials provided with
the distribution.
00022   * 3. Neither the name of STMicroelectronic
s nor the names of other
00023   *    contributors to this software may be
used to endorse or promote products
00024   *    derived from this software without sp
ecific written permission.
00025   * 4. This software, including modification
s and/or derivative works of this
00026   *    software, must execute solely and exc
lusively on microcontroller or
00027   *    microprocessor devices manufactured b
y or for STMicroelectronics.
00028   * 5. Redistribution and use of this softwa
re other than as permitted under
00029   *    this license is void and will automat
ically terminate your rights under
00030   *    this license.
00031   *
00032   * THIS SOFTWARE IS PROVIDED BY STMICROELEC
TRONICS AND CONTRIBUTORS "AS IS"
00033   * AND ANY EXPRESS, IMPLIED OR STATUTORY WA
RRANTIES, INCLUDING, BUT NOT
00034   * LIMITED TO, THE IMPLIED WARRANTIES OF ME
RCHANTABILITY, FITNESS FOR A
00035   * PARTICULAR PURPOSE AND NON-INFRINGEMENT
OF THIRD PARTY INTELLECTUAL PROPERTY
00036   * RIGHTS ARE DISCLAIMED TO THE FULLEST EXT
ENT PERMITTED BY LAW. IN NO EVENT
00037   * SHALL STMICROELECTRONICS OR CONTRIBUTORS
```

```
 BE LIABLE FOR ANY DIRECT, INDIRECT,
00038   * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSE
QUENTIAL DAMAGES (INCLUDING, BUT NOT
00039   * LIMITED TO, PROCUREMENT OF SUBSTITUTE GO
ODS OR SERVICES; LOSS OF USE, DATA,
00040   * OR PROFITS; OR BUSINESS INTERRUPTION) HO
WEVER CAUSED AND ON ANY THEORY OF
00041   * LIABILITY, WHETHER IN CONTRACT, STRICT L
IABILITY, OR TORT (INCLUDING
00042   * NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE,
00043   * EVEN IF ADVISED OF THE POSSIBILITY OF SU
CH DAMAGE.
00044   *
00045   ****************************************
**********************************
00046   */
00047
00048 /* Includes ------------------------------
-----------------------------------*/
00049 #include "stm32l475e_iot01_tsensor.h"
00050
00051 /** @addtogroup BSP
00052   * @{
00053   */
00054
00055 /** @addtogroup STM32L475E_IOT01
00056   * @{
00057   */
00058
00059 /** @defgroup STM32L475E_IOT01_TEMPERATURE T
EMPERATURE
00060   * @{
00061   */
00062
00063 /** @defgroup STM32L475E_IOT01_TEMPERATURE_P
rivate_Variables TEMPERATURE Private Variables
```

```
00064   * @{
00065   */
00066 static TSENSOR_DrvTypeDef *tsensor_drv;
00067 /**
00068   * @}
00069   */
00070
00071 /** @defgroup STM32L475E_IOT01_TEMPERATURE_P
rivate_Functions TEMPERATURE Private Functions
00072   * @{
00073   */
00074
00075 /**
00076   * @brief  Initializes peripherals used by
the I2C Temperature Sensor driver.
00077   * @retval TSENSOR status
00078   */
00079 uint32_t BSP_TSENSOR_Init(void)
00080 {
00081   uint8_t ret = TSENSOR_ERROR;
00082
00083 #ifdef USE_LPS22HB_TEMP
00084   tsensor_drv = &LPS22HB_T_Drv;
00085 #else /* USE_HTS221_TEMP */
00086   tsensor_drv = &HTS221_T_Drv;
00087 #endif
00088
00089   /* Low level init */
00090   SENSOR_IO_Init();
00091
00092   /* TSENSOR Init */
00093   tsensor_drv->Init(TSENSOR_I2C_ADDRESS, NUL
L);
00094
00095   ret = TSENSOR_OK;
00096
00097   return ret;
```

```
00098 }
00099
00100 /**
00101   * @brief  Read Temperature register of TS7
51.
00102   * @retval STTS751 measured temperature val
ue.
00103   */
00104 float BSP_TSENSOR_ReadTemp(void)
00105 {
00106   return tsensor_drv->ReadTemp(TSENSOR_I2C_A
DDRESS);
00107 }
00108
00109 /**
00110   * @}
00111   */
00112
00113 /**
00114   * @}
00115   */
00116
00117 /**
00118   * @}
00119   */
00120
00121 /**
00122   * @}
00123   */
00124 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

B-L475E-IOT01 BSP User Manual

BSP

Modules

STM32L475E_IOT01

Generated on Thu Mar 16 2017 10:38:33 for B-L475E-IOT01 BSP User Manual by doxygen 1.7.6.1

B-L475E-IOT01 BSP User Manual

STM32L475E_IOT01

BSP

Modules

- LOW LEVEL
- ACCELERO
- GYROSCOPE
- HUMIDITY
- MAGNETO
- PRESSURE
- QSPI
- TEMPERATURE

B-L475E-IOT01 BSP User Manual

ACCELERO

STM32L475E_IOT01

Modules

- ACCELERO Private Variables
- ACCELERO Private Functions
- ACCELERO Exported Types
- ACCELERO Exported Functions

B-L475E-IOT01 BSP User Manual

GYROSCOPE

STM32L475E_IOT01

Modules

- GYROSCOPE Private Variables
- GYROSCOPE Private Functions
- GYROSCOPE Exported Constants
- GYROSCOPE Exported Functions

B-L475E-IOT01 BSP User Manual

HUMIDITY

STM32L475E_IOT01

Modules

- HUMIDITY Private Variables
- HUMIDITY Private Functions
- HUMIDITY Exported Types
- HUMIDITY Exported Functions

B-L475E-IOT01 BSP User Manual

MAGNETO

STM32L475E_IOT01

Modules

- MAGNETO Private Variables
- MAGNETO Private Functions
- MAGNETO Exported Types
- MAGNETO Exported Functions

B-L475E-IOT01 BSP User Manual

PRESSURE

STM32L475E_IOT01

Modules

- PRESSURE Private Variables
- PRESSURE Private Functions
- PRESSURE Exported Types
- PRESSURE Exported Functions

B-L475E-IOT01 BSP User Manual

QSPI

STM32L475E_IOT01

Modules

- QSPI Private Constants
- QSPI Private Variables
- QSPI Private Functions
- QSPI Exported Constants
- QSPI Exported Types
- QSPI Exported Functions

B-L475E-IOT01 BSP User Manual

TEMPERATURE

STM32L475E_IOT01

Modules

- TEMPERATURE Private Variables
- TEMPERATURE Private Functions
- TEMPERATURE Exported Types
- TEMPERATURE Exported Constants