STM3210E_EVAL BSP User Manual

LCD_CONTROLLER_TypeDef Struct Reference

Private Types Definitions

Data Fields

__IO uint16_t	**REG**
__IO uint16_t	**RAM**

Detailed Description

Definition at line **58** of file **stm3210e_eval.c**.

Field Documentation

__IO uint16_t LCD_CONTROLLER_TypeDef::RAM

Definition at line **61** of file **stm3210e_eval.c**.

__IO uint16_t LCD_CONTROLLER_TypeDef::REG

Definition at line **60** of file **stm3210e_eval.c**.

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

- **stm3210e_eval.c**

STM3210E_EVAL BSP User Manual

LCD_DrawPropTypeDef Struct Reference

Exported Types

`#include <stm3210e_eval_lcd.h>`

Data Fields

uint32_t	**TextColor**
uint32_t	**BackColor**
sFONT *	**pFont**

Detailed Description

Definition at line **70** of file **stm3210e_eval_lcd.h**.

Field Documentation

uint32_t LCD_DrawPropTypeDef::BackColor

Definition at line 73 of file stm3210e_eval_lcd.h.

Referenced by BSP_LCD_ClearStringLine(), BSP_LCD_GetBackColor(), BSP_LCD_Init(), BSP_LCD_SetBackColor(), and LCD_DrawChar().

sFONT* LCD_DrawPropTypeDef::pFont

Definition at line 74 of file stm3210e_eval_lcd.h.

Referenced by BSP_LCD_ClearStringLine(), BSP_LCD_DisplayChar(), BSP_LCD_DisplayStringAt(), BSP_LCD_GetFont(), BSP_LCD_Init(), BSP_LCD_SetFont(), and LCD_DrawChar().

uint32_t LCD_DrawPropTypeDef::TextColor

Definition at line 72 of file stm3210e_eval_lcd.h.

Referenced by BSP_LCD_Clear(), BSP_LCD_ClearStringLine(), BSP_LCD_DrawCircle(), BSP_LCD_DrawEllipse(), BSP_LCD_DrawHLine(), BSP_LCD_DrawLine(), BSP_LCD_DrawVLine(), BSP_LCD_FillCircle(), BSP_LCD_FillRect(), BSP_LCD_GetTextColor(), BSP_LCD_Init(), BSP_LCD_SetTextColor(), and LCD_DrawChar().

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

- stm3210e_eval_lcd.h

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP
User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Point Struct Reference

Exported Constants

```
#include <stm3210e_eval_lcd.h>
```

Data Fields

int16_t	**X**
int16_t	**Y**

Detailed Description

Definition at line **91** of file **stm3210e_eval_lcd.h**.

Field Documentation

int16_t Point::X

Definition at line **93** of file **stm3210e_eval_lcd.h**.

Referenced by **BSP_LCD_DrawPolygon()**, and **BSP_LCD_FillPolygon()**.

int16_t Point::Y

Definition at line **94** of file **stm3210e_eval_lcd.h**.

Referenced by **BSP_LCD_DrawPolygon()**, and **BSP_LCD_FillPolygon()**.

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

- **stm3210e_eval_lcd.h**

STM3210E_EVAL BSP User Manual

Private_Types_Definitions

STM3210E_EVAL NAND

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Private_Defines

STM3210E_EVAL NAND

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Private_Macros

STM3210E_EVAL NAND

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Private_Function_Prototypes

STM3210E_EVAL NAND

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Exported_Types

STM3210E_EVAL NAND

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Exported_Macro

STM3210E_EVAL NAND

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Private_Types_Definitions

STM3210E_EVAL NOR

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Private_Defines

STM3210E_EVAL NOR

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Private_Macros

STM3210E_EVAL NOR

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Private_Function_Prototypes

STM3210E_EVAL NOR

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Exported_Types

STM3210E_EVAL NOR

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Exported_Macro

STM3210E_EVAL NOR

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Exported_Types

STM3210E_EVAL Serial FLASH

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Exported_Macros

STM3210E_EVAL Serial FLASH

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Private_Types_Definitions

STM3210E_EVAL SRAM

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Private_Defines

STM3210E_EVAL SRAM

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Private_Macros

STM3210E_EVAL SRAM

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Private_Function_Prototypes

STM3210E_EVAL SRAM

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Exported_Types

STM3210E_EVAL SRAM

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Exported_Macro

STM3210E_EVAL SRAM

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

AUDIO_Private_Types

STM3210E_EVAL_AUDIO

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

AUDIO_Private_Defines

STM3210E_EVAL_AUDIO

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

AUDIO Private_Macros

STM3210E_EVAL_AUDIO

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

AUDIO_Private_Function_Prototypes

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

AUDIO_Exported_Types

STM3210E_EVAL_AUDIO

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

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

- BackColor : **LCD_DrawPropTypeDef**
- pFont : **LCD_DrawPropTypeDef**
- RAM : **LCD_CONTROLLER_TypeDef**
- REG : **LCD_CONTROLLER_TypeDef**
- TextColor : **LCD_DrawPropTypeDef**
- X : **Point**
- Y : **Point**

STM3210E_EVAL BSP User Manual

- BackColor : **LCD_DrawPropTypeDef**
- pFont : **LCD_DrawPropTypeDef**
- RAM : **LCD_CONTROLLER_TypeDef**
- REG : **LCD_CONTROLLER_TypeDef**
- TextColor : **LCD_DrawPropTypeDef**
- X : **Point**
- Y : **Point**

STM3210E_EVAL BSP User Manual

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

- _ -

- __DMAx_TxRx_CLK_ENABLE : **stm3210e_eval_sd.h**
- __SD_DETECT_GPIO_CLK_ENABLE : **stm3210e_eval_sd.h**
- __SRAM_DMAx_CLK_ENABLE : **stm3210e_eval_sram.h**
- __STM3210E_EVAL_BSP_VERSION : **stm3210e_eval.c**
- __STM3210E_EVAL_BSP_VERSION_MAIN : **stm3210e_eval.c**
- __STM3210E_EVAL_BSP_VERSION_RC : **stm3210e_eval.c**
- __STM3210E_EVAL_BSP_VERSION_SUB1 : **stm3210e_eval.c**
- __STM3210E_EVAL_BSP_VERSION_SUB2 : **stm3210e_eval.c**
- __STM3210e_EVAL_SRAM_H : **stm3210e_eval_sram.h**

STM3210E_EVAL BSP User Manual

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

- a -

- ABS : **stm3210e_eval_lcd.c**
- AUDIO_ERROR : **stm3210e_eval_audio.h**
- AUDIO_I2C_ADDRESS : **stm3210e_eval.h**
- AUDIO_IO_Init() : **stm3210e_eval.c**
- AUDIO_IO_Read() : **stm3210e_eval.c**
- AUDIO_IO_Write() : **stm3210e_eval.c**
- AUDIO_OK : **stm3210e_eval_audio.h**
- AUDIO_OUT_IRQ_PREPRIO : **stm3210e_eval_audio.h**
- AUDIO_RESET_GPIO : **stm3210e_eval.h**
- AUDIO_RESET_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- AUDIO_RESET_PIN : **stm3210e_eval.h**
- audio_sample_SendDummyData : **stm3210e_eval_audio.c**
- AUDIO_SAMPLE_SENDDUMMYDATA_SIZE : **stm3210e_eval_audio.c**
- AUDIO_TIMEOUT : **stm3210e_eval_audio.h**

STM3210E_EVAL BSP User Manual

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

- b -

- bitmap : **stm3210e_eval_lcd.c**
- BLOCKERASE_TIMEOUT : **stm3210e_eval_nor.h**
- BSP_AUDIO_OUT_ChangeBuffer() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_Error_CallBack() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_HalfTransfer_CallBack() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_Init() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_Pause() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_Play() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_Resume() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_SetFrequency() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_SetMute() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_SetOutputMode() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_SetVolume() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_Stop() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_TransferComplete_CallBack() : **stm3210e_eval_audio.c**
- BSP_COM_Init() : **stm3210e_eval.c**
- BSP_GetVersion() : **stm3210e_eval.c**

- BSP_JOY_GetState() : **stm3210e_eval.c**
- BSP_JOY_Init() : **stm3210e_eval.c**
- BSP_LCD_Clear() : **stm3210e_eval_lcd.c**
- BSP_LCD_ClearStringLine() : **stm3210e_eval_lcd.c**
- BSP_LCD_DisplayChar() : **stm3210e_eval_lcd.c**
- BSP_LCD_DisplayOff() : **stm3210e_eval_lcd.c**
- BSP_LCD_DisplayOn() : **stm3210e_eval_lcd.c**
- BSP_LCD_DisplayStringAt() : **stm3210e_eval_lcd.c**
- BSP_LCD_DisplayStringAtLine() : **stm3210e_eval_lcd.c**
- BSP_LCD_DrawBitmap() : **stm3210e_eval_lcd.c**
- BSP_LCD_DrawCircle() : **stm3210e_eval_lcd.c**
- BSP_LCD_DrawEllipse() : **stm3210e_eval_lcd.c**
- BSP_LCD_DrawHLine() : **stm3210e_eval_lcd.c**
- BSP_LCD_DrawLine() : **stm3210e_eval_lcd.c**
- BSP_LCD_DrawPixel() : **stm3210e_eval_lcd.h**
- BSP_LCD_DrawPolygon() : **stm3210e_eval_lcd.c**
- BSP_LCD_DrawRect() : **stm3210e_eval_lcd.c**
- BSP_LCD_DrawRGBImage() : **stm3210e_eval_lcd.h**
- BSP_LCD_DrawVLine() : **stm3210e_eval_lcd.c**
- BSP_LCD_FillCircle() : **stm3210e_eval_lcd.c**
- BSP_LCD_FillEllipse() : **stm3210e_eval_lcd.c**
- BSP_LCD_FillPolygon() : **stm3210e_eval_lcd.c**
- BSP_LCD_FillRect() : **stm3210e_eval_lcd.c**
- BSP_LCD_FillTriangle() : **stm3210e_eval_lcd.c**
- BSP_LCD_GetBackColor() : **stm3210e_eval_lcd.c**
- BSP_LCD_GetFont() : **stm3210e_eval_lcd.c**
- BSP_LCD_GetTextColor() : **stm3210e_eval_lcd.c**
- BSP_LCD_GetXSize() : **stm3210e_eval_lcd.c**
- BSP_LCD_GetYSize() : **stm3210e_eval_lcd.c**
- BSP_LCD_Init() : **stm3210e_eval_lcd.c**
- BSP_LCD_ReadPixel() : **stm3210e_eval_lcd.c**
- BSP_LCD_SetBackColor() : **stm3210e_eval_lcd.c** , **stm3210e_eval_lcd.h**
- BSP_LCD_SetFont() : **stm3210e_eval_lcd.c**
- BSP_LCD_SetTextColor() : **stm3210e_eval_lcd.c** , **stm3210e_eval_lcd.h**
- BSP_LED_Init() : **stm3210e_eval.c**

- BSP_LED_Off() : **stm3210e_eval.c**
- BSP_LED_On() : **stm3210e_eval.c**
- BSP_LED_Toggle() : **stm3210e_eval.c**
- BSP_NAND_Erase_Block() : **stm3210e_eval_nand.c**
- BSP_NAND_Erase_Chip() : **stm3210e_eval_nand.h**
- BSP_NAND_Init() : **stm3210e_eval_nand.c**
- BSP_NAND_Read_ID() : **stm3210e_eval_nand.c**
- BSP_NAND_ReadData() : **stm3210e_eval_nand.c**
- BSP_NAND_WriteData() : **stm3210e_eval_nand.c**
- BSP_NOR_Erase_Block() : **stm3210e_eval_nor.c**
- BSP_NOR_Erase_Chip() : **stm3210e_eval_nor.c**
- BSP_NOR_Init() : **stm3210e_eval_nor.c**
- BSP_NOR_ProgramData() : **stm3210e_eval_nor.c**
- BSP_NOR_Read_ID() : **stm3210e_eval_nor.c**
- BSP_NOR_ReadData() : **stm3210e_eval_nor.c**
- BSP_NOR_ReturnToReadMode() : **stm3210e_eval_nor.c**
- BSP_NOR_WriteData() : **stm3210e_eval_nor.c**
- BSP_PB_GetState() : **stm3210e_eval.c**
- BSP_PB_Init() : **stm3210e_eval.c**
- BSP_SD_DetectCallback() : **stm3210e_eval_sd.h** , **stm3210e_eval_sd.c**
- BSP_SD_DetectIT() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_DMA_Rx_IRQHandler() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_DMA_Tx_IRQHandler() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_Erase() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_GetCardInfo() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_GetStatus() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_Init() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_IRQHandler() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_IsDetected() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**

- BSP_SD_ITConfig() : **stm3210e_eval_sd.h** , **stm3210e_eval_sd.c**
- BSP_SD_ReadBlocks() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_ReadBlocks_DMA() : **stm3210e_eval_sd.h** , **stm3210e_eval_sd.c**
- BSP_SD_WriteBlocks() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_WriteBlocks_DMA() : **stm3210e_eval_sd.h** , **stm3210e_eval_sd.c**
- BSP_SERIAL_FLASH_EraseBulk() : **stm3210e_eval_serialflash.c**
- BSP_SERIAL_FLASH_EraseSector() : **stm3210e_eval_serialflash.c**
- BSP_SERIAL_FLASH_Init() : **stm3210e_eval_serialflash.c**
- BSP_SERIAL_FLASH_ReadData() : **stm3210e_eval_serialflash.c**
- BSP_SERIAL_FLASH_ReadID() : **stm3210e_eval_serialflash.c**
- BSP_SERIAL_FLASH_WriteData() : **stm3210e_eval_serialflash.c**
- BSP_SERIAL_FLASH_WritePage() : **stm3210e_eval_serialflash.c**
- BSP_SRAM_DMA_IRQHandler() : **stm3210e_eval_sram.c**
- BSP_SRAM_Init() : **stm3210e_eval_sram.c**
- BSP_SRAM_ReadData() : **stm3210e_eval_sram.c**
- BSP_SRAM_ReadData_DMA() : **stm3210e_eval_sram.c**
- BSP_SRAM_WriteData() : **stm3210e_eval_sram.c**
- BSP_SRAM_WriteData_DMA() : **stm3210e_eval_sram.c**
- BSP_TSENSOR_Init() : **stm3210e_eval_tsensor.c**
- BSP_TSENSOR_ReadStatus() : **stm3210e_eval_tsensor.c**
- BSP_TSENSOR_ReadTemp() : **stm3210e_eval_tsensor.c**
- BUTTON_DOWN : **stm3210e_eval.h**
- BUTTON_IRQn : **stm3210e_eval.c**
- BUTTON_KEY : **stm3210e_eval.h**
- BUTTON_LEFT : **stm3210e_eval.h**
- BUTTON_MODE_EVT : **stm3210e_eval.h**
- BUTTON_MODE_EXTI : **stm3210e_eval.h**

- BUTTON_MODE_GPIO : **stm3210e_eval.h**
- BUTTON_PIN : **stm3210e_eval.c**
- BUTTON_PORT : **stm3210e_eval.c**
- BUTTON_RIGHT : **stm3210e_eval.h**
- BUTTON_SEL : **stm3210e_eval.h**
- BUTTON_TAMPER : **stm3210e_eval.h**
- Button_TypeDef : **stm3210e_eval.h**
- BUTTON_UP : **stm3210e_eval.h**
- BUTTON_WAKEUP : **stm3210e_eval.h**
- ButtonMode_TypeDef : **stm3210e_eval.h**
- BUTTONn : **stm3210e_eval.h**
- BUTTONx_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- BUTTONx_GPIO_CLK_ENABLE : **stm3210e_eval.h**

STM3210E_EVAL BSP User Manual

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

- c -

- CENTER_MODE : **stm3210e_eval_lcd.h**
- CHIPERASE_TIMEOUT : **stm3210e_eval_nor.h**
- COM1 : **stm3210e_eval.h**
- COM2 : **stm3210e_eval.h**
- COM_RX_PIN : **stm3210e_eval.c**
- COM_RX_PORT : **stm3210e_eval.c**
- COM_TX_PIN : **stm3210e_eval.c**
- COM_TX_PORT : **stm3210e_eval.c**
- COM_TypeDef : **stm3210e_eval.h**
- COM_USART : **stm3210e_eval.c**
- COMn : **stm3210e_eval.h**
- COMx_CLK_DISABLE : **stm3210e_eval.h**
- COMx_CLK_ENABLE : **stm3210e_eval.h**
- COMx_RX_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- COMx_RX_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- COMx_TX_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- COMx_TX_GPIO_CLK_ENABLE : **stm3210e_eval.h**

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP
User Manual by  1.7.6.1

STM3210E_EVAL BSP User Manual

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

- d -

- DMA_MAX : **stm3210e_eval_audio.h**
- DMA_MAX_SZE : **stm3210e_eval_audio.h**
- DOWN_JOY_EXTI_IRQn : **stm3210e_eval.h**
- DOWN_JOY_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- DOWN_JOY_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- DOWN_JOY_GPIO_PORT : **stm3210e_eval.h**
- DOWN_JOY_PIN : **stm3210e_eval.h**
- DrawProp : **stm3210e_eval_lcd.c**

STM3210E_EVAL BSP User Manual

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

- e -

- EVAL_COM1 : **stm3210e_eval.h**
- EVAL_COM1_CLK_DISABLE : **stm3210e_eval.h**
- EVAL_COM1_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_COM1_IRQn : **stm3210e_eval.h**
- EVAL_COM1_RX_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- EVAL_COM1_RX_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_COM1_RX_GPIO_PORT : **stm3210e_eval.h**
- EVAL_COM1_RX_PIN : **stm3210e_eval.h**
- EVAL_COM1_TX_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- EVAL_COM1_TX_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_COM1_TX_GPIO_PORT : **stm3210e_eval.h**
- EVAL_COM1_TX_PIN : **stm3210e_eval.h**
- EVAL_COM2 : **stm3210e_eval.h**
- EVAL_COM2_CLK_DISABLE : **stm3210e_eval.h**
- EVAL_COM2_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_COM2_IRQn : **stm3210e_eval.h**
- EVAL_COM2_RX_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- EVAL_COM2_RX_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_COM2_RX_GPIO_PORT : **stm3210e_eval.h**

- EVAL_COM2_RX_PIN : **stm3210e_eval.h**
- EVAL_COM2_TX_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- EVAL_COM2_TX_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_COM2_TX_GPIO_PORT : **stm3210e_eval.h**
- EVAL_COM2_TX_PIN : **stm3210e_eval.h**
- EVAL_I2Cx : **stm3210e_eval.h**
- EVAL_I2Cx_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_I2Cx_ER_IRQHandler : **stm3210e_eval.h**
- EVAL_I2Cx_ER_IRQn : **stm3210e_eval.h**
- EVAL_I2Cx_EV_IRQHandler : **stm3210e_eval.h**
- EVAL_I2Cx_EV_IRQn : **stm3210e_eval.h**
- EVAL_I2Cx_FORCE_RESET : **stm3210e_eval.h**
- EVAL_I2Cx_RELEASE_RESET : **stm3210e_eval.h**
- EVAL_I2Cx_SCL_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_I2Cx_SCL_GPIO_PORT : **stm3210e_eval.h**
- EVAL_I2Cx_SCL_PIN : **stm3210e_eval.h**
- EVAL_I2Cx_SDA_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_I2Cx_SDA_GPIO_PORT : **stm3210e_eval.h**
- EVAL_I2Cx_SDA_PIN : **stm3210e_eval.h**
- EVAL_I2Cx_TIMEOUT_MAX : **stm3210e_eval.h**
- EVAL_SPIx : **stm3210e_eval.h**
- EVAL_SPIx_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_SPIx_MISO_MOSI_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_SPIx_MISO_MOSI_GPIO_PORT : **stm3210e_eval.h**
- EVAL_SPIx_MISO_PIN : **stm3210e_eval.h**
- EVAL_SPIx_MOSI_PIN : **stm3210e_eval.h**
- EVAL_SPIx_SCK_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- EVAL_SPIx_SCK_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_SPIx_SCK_GPIO_PORT : **stm3210e_eval.h**
- EVAL_SPIx_SCK_PIN : **stm3210e_eval.h**
- EVAL_SPIx_TIMEOUT_MAX : **stm3210e_eval.h**

User Manual by  1.7.6.1

STM3210E_EVAL BSP User Manual

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

- f -

- FLASH_ERROR : **stm3210e_eval_serialflash.h**
- FLASH_OK : **stm3210e_eval_serialflash.h**
- FLASH_SPI_CMD_BE : **stm3210e_eval.h**
- FLASH_SPI_CMD_RDID : **stm3210e_eval.h**
- FLASH_SPI_CMD_RDSR : **stm3210e_eval.h**
- FLASH_SPI_CMD_READ : **stm3210e_eval.h**
- FLASH_SPI_CMD_SE : **stm3210e_eval.h**
- FLASH_SPI_CMD_WREN : **stm3210e_eval.h**
- FLASH_SPI_CMD_WRITE : **stm3210e_eval.h**
- FLASH_SPI_CMD_WRSR : **stm3210e_eval.h**
- FLASH_SPI_CS_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- FLASH_SPI_CS_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- FLASH_SPI_CS_GPIO_PORT : **stm3210e_eval.h**
- FLASH_SPI_CS_HIGH : **stm3210e_eval.h**
- FLASH_SPI_CS_LOW : **stm3210e_eval.h**
- FLASH_SPI_CS_PIN : **stm3210e_eval.h**
- FLASH_SPI_DUMMY_BYTE : **stm3210e_eval.h**
- FLASH_SPI_IO_Init() : **stm3210e_eval.c** , **stm3210e_eval_serialflash.h**

- FLASH_SPI_IO_ReadByte() : **stm3210e_eval.c** , **stm3210e_eval_serialflash.h**
- FLASH_SPI_IO_ReadData() : **stm3210e_eval_serialflash.h** , **stm3210e_eval.c**
- FLASH_SPI_IO_ReadID() : **stm3210e_eval.c** , **stm3210e_eval_serialflash.h**
- FLASH_SPI_IO_WaitForWriteEnd() : **stm3210e_eval_serialflash.h** , **stm3210e_eval.c**
- FLASH_SPI_IO_WriteByte() : **stm3210e_eval_serialflash.h** , **stm3210e_eval.c**
- FLASH_SPI_IO_WriteEnable() : **stm3210e_eval_serialflash.h** , **stm3210e_eval.c**
- FLASH_SPI_M25P128_ID : **stm3210e_eval.h**
- FLASH_SPI_M25P64_ID : **stm3210e_eval.h**
- FLASH_SPI_PAGESIZE : **stm3210e_eval.h**
- FLASH_SPI_WIP_FLAG : **stm3210e_eval.h**
- FSMC_BANK1NORSRAM4_Init() : **stm3210e_eval.c**
- FSMC_BANK1NORSRAM4_MspInit() : **stm3210e_eval.c**
- FSMC_BANK1NORSRAM4_ReadData() : **stm3210e_eval.c**
- FSMC_BANK1NORSRAM4_WriteData() : **stm3210e_eval.c**
- FSMC_BANK1NORSRAM4_WriteReg() : **stm3210e_eval.c**

STM3210E_EVAL BSP User Manual

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

- h -

- HAL_I2S_ErrorCallback() : **stm3210e_eval_audio.c**
- HAL_I2S_TxCpltCallback() : **stm3210e_eval_audio.c**
- HAL_I2S_TxHalfCpltCallback() : **stm3210e_eval_audio.c**
- HAL_NOR_MspWait() : **stm3210e_eval_nor.c**
- hAudioOutI2s : **stm3210e_eval_audio.c**
- heval_I2c : **stm3210e_eval.c**
- heval_Spi : **stm3210e_eval.c**

STM3210E_EVAL 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_Error() : **stm3210e_eval.c**
- I2Cx_Init() : **stm3210e_eval.c**
- I2Cx_IsDeviceReady() : **stm3210e_eval.c**
- I2Cx_MspInit() : **stm3210e_eval.c**
- I2Cx_ReadBuffer() : **stm3210e_eval.c**
- I2Cx_ReadData() : **stm3210e_eval.c**
- I2Cx_WriteBuffer() : **stm3210e_eval.c**
- I2Cx_WriteData() : **stm3210e_eval.c**
- I2cxTimeout : **stm3210e_eval.c**
- I2SOUT : **stm3210e_eval_audio.h**
- I2SOUT_CLK_ENABLE : **stm3210e_eval_audio.h**
- I2SOUT_DMAx_CHANNEL : **stm3210e_eval_audio.h**
- I2SOUT_DMAx_CLK_ENABLE : **stm3210e_eval_audio.h**
- I2SOUT_DMAx_IRQ : **stm3210e_eval_audio.h**
- I2SOUT_DMAx_MEM_DATA_SIZE : **stm3210e_eval_audio.h**
- I2SOUT_DMAx_PERIPH_DATA_SIZE : **stm3210e_eval_audio.h**
- I2SOUT_Init() : **stm3210e_eval_audio.c**
- I2SOUT_IRQHandler : **stm3210e_eval_audio.h**
- I2SOUT_MCK_CLK_ENABLE : **stm3210e_eval_audio.h**

- I2SOUT_MCK_GPIO_PORT : **stm3210e_eval_audio.h**
- I2SOUT_MCK_PIN : **stm3210e_eval_audio.h**
- I2SOUT_MspInit() : **stm3210e_eval_audio.c**
- I2SOUT_SCK_PIN : **stm3210e_eval_audio.h**
- I2SOUT_SCK_SD_CLK_ENABLE : **stm3210e_eval_audio.h**
- I2SOUT_SCK_SD_GPIO_PORT : **stm3210e_eval_audio.h**
- I2SOUT_SD_PIN : **stm3210e_eval_audio.h**
- I2SOUT_SendDummyData_Start() : **stm3210e_eval_audio.c**
- I2SOUT_SendDummyData_Stop() : **stm3210e_eval_audio.c**
- I2SOUT_WS_CLK_ENABLE : **stm3210e_eval_audio.h**
- I2SOUT_WS_GPIO_PORT : **stm3210e_eval_audio.h**
- I2SOUT_WS_PIN : **stm3210e_eval_audio.h**

STM3210E_EVAL BSP User Manual

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

- j -

- JOY_DOWN : **stm3210e_eval.h**
- JOY_IRQn : **stm3210e_eval.c**
- JOY_LEFT : **stm3210e_eval.h**
- JOY_MODE_EXTI : **stm3210e_eval.h**
- JOY_MODE_GPIO : **stm3210e_eval.h**
- JOY_NONE : **stm3210e_eval.h**
- JOY_PIN : **stm3210e_eval.c**
- JOY_PORT : **stm3210e_eval.c**
- JOY_RIGHT : **stm3210e_eval.h**
- JOY_SEL : **stm3210e_eval.h**
- JOY_UP : **stm3210e_eval.h**
- JOYMode_TypeDef : **stm3210e_eval.h**
- JOYn : **stm3210e_eval.h**
- JOYState_TypeDef : **stm3210e_eval.h**
- JOYx_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- JOYx_GPIO_CLK_ENABLE : **stm3210e_eval.h**

User Manual by  1.7.6.1

STM3210E_EVAL BSP User Manual

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

- k -

- KEY_BUTTON_EXTI_IRQn : **stm3210e_eval.h**
- KEY_BUTTON_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- KEY_BUTTON_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- KEY_BUTTON_GPIO_PORT : **stm3210e_eval.h**
- KEY_BUTTON_PIN : **stm3210e_eval.h**

STM3210E_EVAL BSP User Manual

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

- l -

- LCD_COLOR_BLACK : **stm3210e_eval_lcd.h**
- LCD_COLOR_BLUE : **stm3210e_eval_lcd.h**
- LCD_COLOR_BROWN : **stm3210e_eval_lcd.h**
- LCD_COLOR_CYAN : **stm3210e_eval_lcd.h**
- LCD_COLOR_DARKBLUE : **stm3210e_eval_lcd.h**
- LCD_COLOR_DARKCYAN : **stm3210e_eval_lcd.h**
- LCD_COLOR_DARKGRAY : **stm3210e_eval_lcd.h**
- LCD_COLOR_DARKGREEN : **stm3210e_eval_lcd.h**
- LCD_COLOR_DARKMAGENTA : **stm3210e_eval_lcd.h**
- LCD_COLOR_DARKRED : **stm3210e_eval_lcd.h**
- LCD_COLOR_DARKYELLOW : **stm3210e_eval_lcd.h**
- LCD_COLOR_GRAY : **stm3210e_eval_lcd.h**
- LCD_COLOR_GREEN : **stm3210e_eval_lcd.h**
- LCD_COLOR_LIGHTBLUE : **stm3210e_eval_lcd.h**
- LCD_COLOR_LIGHTCYAN : **stm3210e_eval_lcd.h**
- LCD_COLOR_LIGHTGRAY : **stm3210e_eval_lcd.h**
- LCD_COLOR_LIGHTGREEN : **stm3210e_eval_lcd.h**
- LCD_COLOR_LIGHTMAGENTA : **stm3210e_eval_lcd.h**
- LCD_COLOR_LIGHTRED : **stm3210e_eval_lcd.h**

- LCD_COLOR_LIGHTYELLOW : **stm3210e_eval_lcd.h**
- LCD_COLOR_MAGENTA : **stm3210e_eval_lcd.h**
- LCD_COLOR_ORANGE : **stm3210e_eval_lcd.h**
- LCD_COLOR_RED : **stm3210e_eval_lcd.h**
- LCD_COLOR_WHITE : **stm3210e_eval_lcd.h**
- LCD_COLOR_YELLOW : **stm3210e_eval_lcd.h**
- LCD_DEFAULT_FONT : **stm3210e_eval_lcd.h**
- LCD_Delay() : **stm3210e_eval.c**
- LCD_DrawChar() : **stm3210e_eval_lcd.c**
- LCD_DrawPixel() : **stm3210e_eval_lcd.c**
- lcd_drv : **stm3210e_eval_lcd.c**
- LCD_ERROR : **stm3210e_eval_lcd.h**
- LCD_IO_Init() : **stm3210e_eval.c**
- LCD_IO_ReadData() : **stm3210e_eval.c**
- LCD_IO_WriteData() : **stm3210e_eval.c**
- LCD_IO_WriteMultipleData() : **stm3210e_eval.c**
- LCD_IO_WriteReg() : **stm3210e_eval.c**
- LCD_OK : **stm3210e_eval_lcd.h**
- LCD_SetDisplayWindow() : **stm3210e_eval_lcd.c**
- LCD_SwapXY : **stm3210e_eval_lcd.c**
- LCD_TIMEOUT : **stm3210e_eval_lcd.h**
- LED1 : **stm3210e_eval.h**
- LED1_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- LED1_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- LED1_GPIO_PORT : **stm3210e_eval.h**
- LED1_PIN : **stm3210e_eval.h**
- LED2 : **stm3210e_eval.h**
- LED2_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- LED2_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- LED2_GPIO_PORT : **stm3210e_eval.h**
- LED2_PIN : **stm3210e_eval.h**
- LED3 : **stm3210e_eval.h**
- LED3_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- LED3_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- LED3_GPIO_PORT : **stm3210e_eval.h**
- LED3_PIN : **stm3210e_eval.h**
- LED4 : **stm3210e_eval.h**

- LED4_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- LED4_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- LED4_GPIO_PORT : **stm3210e_eval.h**
- LED4_PIN : **stm3210e_eval.h**
- LED_BLUE : **stm3210e_eval.h**
- LED_GREEN : **stm3210e_eval.h**
- LED_ORANGE : **stm3210e_eval.h**
- LED_PIN : **stm3210e_eval.c**
- LED_PORT : **stm3210e_eval.c**
- LED_RED : **stm3210e_eval.h**
- Led_TypeDef : **stm3210e_eval.h**
- LEDn : **stm3210e_eval.h**
- LEDx_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- LEDx_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- LEFT_JOY_EXTI_IRQn : **stm3210e_eval.h**
- LEFT_JOY_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- LEFT_JOY_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- LEFT_JOY_GPIO_PORT : **stm3210e_eval.h**
- LEFT_JOY_PIN : **stm3210e_eval.h**
- LEFT_MODE : **stm3210e_eval_lcd.h**
- Line_ModeTypdef : **stm3210e_eval_lcd.h**

STM3210E_EVAL BSP User Manual

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

- m -

- MAX_HEIGHT_FONT : **stm3210e_eval_lcd.c**
- MAX_WIDTH_FONT : **stm3210e_eval_lcd.c**
- MSD_ERROR : **stm3210e_eval_sd.h**
- MSD_OK : **stm3210e_eval_sd.h**

STM3210E_EVAL BSP User Manual

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

- n -

- NAND_BLOCK_SIZE : **stm3210e_eval_nand.h**
- NAND_DEVICE_ADDR : **stm3210e_eval_nand.h**
- NAND_MAX_ZONE : **stm3210e_eval_nand.h**
- NAND_MspInit() : **stm3210e_eval_nand.c**
- NAND_OK : **stm3210e_eval_nand.h**
- NAND_PAGE_SIZE : **stm3210e_eval_nand.h**
- NAND_SPARE_AREA_SIZE : **stm3210e_eval_nand.h**
- NAND_ZONE_SIZE : **stm3210e_eval_nand.h**
- nandHandle : **stm3210e_eval_nand.c**
- NOR_BURSTACCESS : **stm3210e_eval_nor.h**
- NOR_BUSY_STATE : **stm3210e_eval_nor.h**
- NOR_DEVICE_ADDR : **stm3210e_eval_nor.h**
- NOR_MEMORY_WIDTH : **stm3210e_eval_nor.h**
- NOR_MspInit() : **stm3210e_eval_nor.c**
- NOR_READY_BUSY_GPIO : **stm3210e_eval_nor.h**
- NOR_READY_BUSY_PIN : **stm3210e_eval_nor.h**
- NOR_READY_STATE : **stm3210e_eval_nor.h**
- NOR_STATUS_ERROR : **stm3210e_eval_nor.h**
- NOR_STATUS_OK : **stm3210e_eval_nor.h**

- NOR_WRITEBURST : **stm3210e_eval_nor.h**
- norHandle : **stm3210e_eval_nor.c**

STM3210E_EVAL BSP User Manual

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

- o -

- OFFSET_BITMAP : **stm3210e_eval_lcd.c**

STM3210E_EVAL BSP User Manual

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

- p -

- pAudioDrv : **stm3210e_eval_audio.c**
- POLY_X : **stm3210e_eval_lcd.c**
- POLY_Y : **stm3210e_eval_lcd.c**
- pPoint : **stm3210e_eval_lcd.h**
- PROGRAM_TIMEOUT : **stm3210e_eval_nor.h**

STM3210E_EVAL BSP User Manual

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

- r -

- RIGHT_JOY_EXTI_IRQn : **stm3210e_eval.h**
- RIGHT_JOY_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- RIGHT_JOY_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- RIGHT_JOY_GPIO_PORT : **stm3210e_eval.h**
- RIGHT_JOY_PIN : **stm3210e_eval.h**
- RIGHT_MODE : **stm3210e_eval_lcd.h**

STM3210E_EVAL BSP User Manual

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

- s -

- SD_CardInfo : **stm3210e_eval_sd.h**
- SD_DATATIMEOUT : **stm3210e_eval_sd.h**
- SD_DETECT_GPIO_PORT : **stm3210e_eval_sd.h**
- SD_DETECT_IRQn : **stm3210e_eval_sd.h**
- SD_DETECT_PIN : **stm3210e_eval_sd.h**
- SD_DMAConfigRx() : **stm3210e_eval_sd.c**
- SD_DMAConfigTx() : **stm3210e_eval_sd.c**
- SD_DMAx_Rx_INSTANCE : **stm3210e_eval_sd.h**
- SD_DMAx_Rx_IRQHandler : **stm3210e_eval_sd.h**
- SD_DMAx_Rx_IRQn : **stm3210e_eval_sd.h**
- SD_DMAx_Tx_INSTANCE : **stm3210e_eval_sd.h**
- SD_DMAx_Tx_IRQHandler : **stm3210e_eval_sd.h**
- SD_DMAx_Tx_IRQn : **stm3210e_eval_sd.h**
- SD_MspInit() : **stm3210e_eval_sd.c**
- SD_NOT_PRESENT : **stm3210e_eval_sd.h**
- SD_PRESENT : **stm3210e_eval_sd.h**
- SEL_JOY_EXTI_IRQn : **stm3210e_eval.h**
- SEL_JOY_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- SEL_JOY_GPIO_CLK_ENABLE : **stm3210e_eval.h**

- SEL_JOY_GPIO_PORT : **stm3210e_eval.h**
- SEL_JOY_PIN : **stm3210e_eval.h**
- SendDummyData : **stm3210e_eval_audio.c**
- SPIx_Error() : **stm3210e_eval.c**
- SPIx_Init() : **stm3210e_eval.c**
- SPIx_MspInit() : **stm3210e_eval.c**
- SPIx_Read() : **stm3210e_eval.c**
- SPIx_Write() : **stm3210e_eval.c**
- SpixTimeout : **stm3210e_eval.c**
- SRAM_BURSTACCESS : **stm3210e_eval_sram.h**
- SRAM_DEVICE_ADDR : **stm3210e_eval_sram.h**
- SRAM_DEVICE_SIZE : **stm3210e_eval_sram.h**
- SRAM_DMAx_IRQHandler : **stm3210e_eval_sram.h**
- SRAM_DMAx_IRQn : **stm3210e_eval_sram.h**
- SRAM_DMAx_STREAM : **stm3210e_eval_sram.h**
- SRAM_ERROR : **stm3210e_eval_sram.h**
- SRAM_MEMORY_WIDTH : **stm3210e_eval_sram.h**
- SRAM_MspInit() : **stm3210e_eval_sram.c**
- SRAM_OK : **stm3210e_eval_sram.h**
- SRAM_WRITEBURST : **stm3210e_eval_sram.h**
- sramHandle : **stm3210e_eval_sram.c**

STM3210E_EVAL BSP User Manual

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

- t -

- TAMPER_BUTTON_EXTI_IRQn : **stm3210e_eval.h**
- TAMPER_BUTTON_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- TAMPER_BUTTON_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- TAMPER_BUTTON_GPIO_PORT : **stm3210e_eval.h**
- TAMPER_BUTTON_PIN : **stm3210e_eval.h**
- TFT_LCD : **stm3210e_eval.c**
- TFT_LCD_BASE : **stm3210e_eval.c**
- Timing : **stm3210e_eval_nand.c** , **stm3210e_eval_nor.c** , **stm3210e_eval_sram.c**
- tsensor_drv : **stm3210e_eval_tsensor.c**
- TSENSOR_ERROR : **stm3210e_eval_tsensor.h**
- TSENSOR_I2C_ADDRESS_A01 : **stm3210e_eval_tsensor.h**
- TSENSOR_I2C_ADDRESS_A02 : **stm3210e_eval_tsensor.h**
- TSENSOR_IO_Init() : **stm3210e_eval.c**
- TSENSOR_IO_IsDeviceReady() : **stm3210e_eval.c**
- TSENSOR_IO_Read() : **stm3210e_eval.c**
- TSENSOR_IO_Write() : **stm3210e_eval.c**
- TSENSOR_MAX_TRIALS : **stm3210e_eval_tsensor.h**
- TSENSOR_OK : **stm3210e_eval_tsensor.h**

- TSENSOR_Status_TypDef : **stm3210e_eval_tsensor.h**
- TSENSORAddress : **stm3210e_eval_tsensor.c**

STM3210E_EVAL BSP User Manual

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

- u -

- UP_JOY_EXTI_IRQn : **stm3210e_eval.h**
- UP_JOY_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- UP_JOY_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- UP_JOY_GPIO_PORT : **stm3210e_eval.h**
- UP_JOY_PIN : **stm3210e_eval.h**
- uSdCardInfo : **stm3210e_eval_sd.c**
- uSdHandle : **stm3210e_eval_sd.c**

STM3210E_EVAL BSP User Manual

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

- w -

- WAKEUP_BUTTON_EXTI_IRQn : **stm3210e_eval.h**
- WAKEUP_BUTTON_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- WAKEUP_BUTTON_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- WAKEUP_BUTTON_GPIO_PORT : **stm3210e_eval.h**
- WAKEUP_BUTTON_PIN : **stm3210e_eval.h**

STM3210E_EVAL BSP User Manual

- a -

- AUDIO_IO_Init() : **stm3210e_eval.c**
- AUDIO_IO_Read() : **stm3210e_eval.c**
- AUDIO_IO_Write() : **stm3210e_eval.c**

- b -

- BSP_AUDIO_OUT_ChangeBuffer() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_Error_CallBack() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_HalfTransfer_CallBack() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_Init() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_Pause() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_Play() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_Resume() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_SetFrequency() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_SetMute() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_SetOutputMode() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_SetVolume() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_Stop() : **stm3210e_eval_audio.c**
- BSP_AUDIO_OUT_TransferComplete_CallBack() :

stm3210e_eval_audio.c

- BSP_COM_Init() : **stm3210e_eval.c**
- BSP_GetVersion() : **stm3210e_eval.c**
- BSP_JOY_GetState() : **stm3210e_eval.c**
- BSP_JOY_Init() : **stm3210e_eval.c**
- BSP_LCD_Clear() : **stm3210e_eval_lcd.c**
- BSP_LCD_ClearStringLine() : **stm3210e_eval_lcd.c**
- BSP_LCD_DisplayChar() : **stm3210e_eval_lcd.c**
- BSP_LCD_DisplayOff() : **stm3210e_eval_lcd.c**
- BSP_LCD_DisplayOn() : **stm3210e_eval_lcd.c**
- BSP_LCD_DisplayStringAt() : **stm3210e_eval_lcd.c**
- BSP_LCD_DisplayStringAtLine() : **stm3210e_eval_lcd.c**
- BSP_LCD_DrawBitmap() : **stm3210e_eval_lcd.c**
- BSP_LCD_DrawCircle() : **stm3210e_eval_lcd.c**
- BSP_LCD_DrawEllipse() : **stm3210e_eval_lcd.c**
- BSP_LCD_DrawHLine() : **stm3210e_eval_lcd.c**
- BSP_LCD_DrawLine() : **stm3210e_eval_lcd.c**
- BSP_LCD_DrawPixel() : **stm3210e_eval_lcd.h**
- BSP_LCD_DrawPolygon() : **stm3210e_eval_lcd.c**
- BSP_LCD_DrawRect() : **stm3210e_eval_lcd.c**
- BSP_LCD_DrawRGBImage() : **stm3210e_eval_lcd.h**
- BSP_LCD_DrawVLine() : **stm3210e_eval_lcd.c**
- BSP_LCD_FillCircle() : **stm3210e_eval_lcd.c**
- BSP_LCD_FillEllipse() : **stm3210e_eval_lcd.c**
- BSP_LCD_FillPolygon() : **stm3210e_eval_lcd.c**
- BSP_LCD_FillRect() : **stm3210e_eval_lcd.c**
- BSP_LCD_FillTriangle() : **stm3210e_eval_lcd.c**
- BSP_LCD_GetBackColor() : **stm3210e_eval_lcd.c**
- BSP_LCD_GetFont() : **stm3210e_eval_lcd.c**
- BSP_LCD_GetTextColor() : **stm3210e_eval_lcd.c**
- BSP_LCD_GetXSize() : **stm3210e_eval_lcd.c**
- BSP_LCD_GetYSize() : **stm3210e_eval_lcd.c**
- BSP_LCD_Init() : **stm3210e_eval_lcd.c**
- BSP_LCD_ReadPixel() : **stm3210e_eval_lcd.c**
- BSP_LCD_SetBackColor() : **stm3210e_eval_lcd.c** , **stm3210e_eval_lcd.h**
- BSP_LCD_SetFont() : **stm3210e_eval_lcd.c**

- BSP_LCD_SetTextColor() : **stm3210e_eval_lcd.c** , **stm3210e_eval_lcd.h**
- BSP_LED_Init() : **stm3210e_eval.c**
- BSP_LED_Off() : **stm3210e_eval.c**
- BSP_LED_On() : **stm3210e_eval.c**
- BSP_LED_Toggle() : **stm3210e_eval.c**
- BSP_NAND_Erase_Block() : **stm3210e_eval_nand.c**
- BSP_NAND_Erase_Chip() : **stm3210e_eval_nand.h**
- BSP_NAND_Init() : **stm3210e_eval_nand.c**
- BSP_NAND_Read_ID() : **stm3210e_eval_nand.c**
- BSP_NAND_ReadData() : **stm3210e_eval_nand.c**
- BSP_NAND_WriteData() : **stm3210e_eval_nand.c**
- BSP_NOR_Erase_Block() : **stm3210e_eval_nor.c**
- BSP_NOR_Erase_Chip() : **stm3210e_eval_nor.c**
- BSP_NOR_Init() : **stm3210e_eval_nor.c**
- BSP_NOR_ProgramData() : **stm3210e_eval_nor.c**
- BSP_NOR_Read_ID() : **stm3210e_eval_nor.c**
- BSP_NOR_ReadData() : **stm3210e_eval_nor.c**
- BSP_NOR_ReturnToReadMode() : **stm3210e_eval_nor.c**
- BSP_NOR_WriteData() : **stm3210e_eval_nor.c**
- BSP_PB_GetState() : **stm3210e_eval.c**
- BSP_PB_Init() : **stm3210e_eval.c**
- BSP_SD_DetectCallback() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_DetectIT() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_DMA_Rx_IRQHandler() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_DMA_Tx_IRQHandler() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_Erase() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_GetCardInfo() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_GetStatus() : **stm3210e_eval_sd.h** , **stm3210e_eval_sd.c**
- BSP_SD_Init() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_IRQHandler() : **stm3210e_eval_sd.c** ,

stm3210e_eval_sd.h

- BSP_SD_IsDetected() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_ITConfig() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_ReadBlocks() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_ReadBlocks_DMA() : **stm3210e_eval_sd.h** , **stm3210e_eval_sd.c**
- BSP_SD_WriteBlocks() : **stm3210e_eval_sd.c** , **stm3210e_eval_sd.h**
- BSP_SD_WriteBlocks_DMA() : **stm3210e_eval_sd.h** , **stm3210e_eval_sd.c**
- BSP_SERIAL_FLASH_EraseBulk() : **stm3210e_eval_serialflash.c**
- BSP_SERIAL_FLASH_EraseSector() : **stm3210e_eval_serialflash.c**
- BSP_SERIAL_FLASH_Init() : **stm3210e_eval_serialflash.c**
- BSP_SERIAL_FLASH_ReadData() : **stm3210e_eval_serialflash.c**
- BSP_SERIAL_FLASH_ReadID() : **stm3210e_eval_serialflash.c**
- BSP_SERIAL_FLASH_WriteData() : **stm3210e_eval_serialflash.c**
- BSP_SERIAL_FLASH_WritePage() : **stm3210e_eval_serialflash.c**
- BSP_SRAM_DMA_IRQHandler() : **stm3210e_eval_sram.c**
- BSP_SRAM_Init() : **stm3210e_eval_sram.c**
- BSP_SRAM_ReadData() : **stm3210e_eval_sram.c**
- BSP_SRAM_ReadData_DMA() : **stm3210e_eval_sram.c**
- BSP_SRAM_WriteData() : **stm3210e_eval_sram.c**
- BSP_SRAM_WriteData_DMA() : **stm3210e_eval_sram.c**
- BSP_TSENSOR_Init() : **stm3210e_eval_tsensor.c**
- BSP_TSENSOR_ReadStatus() : **stm3210e_eval_tsensor.c**
- BSP_TSENSOR_ReadTemp() : **stm3210e_eval_tsensor.c**

- f -

- FLASH_SPI_IO_Init() : **stm3210e_eval.c** , **stm3210e_eval_serialflash.h**
- FLASH_SPI_IO_ReadByte() : **stm3210e_eval_serialflash.h** , **stm3210e_eval.c**
- FLASH_SPI_IO_ReadData() : **stm3210e_eval.c** , **stm3210e_eval_serialflash.h**
- FLASH_SPI_IO_ReadID() : **stm3210e_eval_serialflash.h** , **stm3210e_eval.c**
- FLASH_SPI_IO_WaitForWriteEnd() : **stm3210e_eval.c** , **stm3210e_eval_serialflash.h**
- FLASH_SPI_IO_WriteByte() : **stm3210e_eval.c** , **stm3210e_eval_serialflash.h**
- FLASH_SPI_IO_WriteEnable() : **stm3210e_eval.c** , **stm3210e_eval_serialflash.h**
- FSMC_BANK1NORSRAM4_Init() : **stm3210e_eval.c**
- FSMC_BANK1NORSRAM4_MspInit() : **stm3210e_eval.c**
- FSMC_BANK1NORSRAM4_ReadData() : **stm3210e_eval.c**
- FSMC_BANK1NORSRAM4_WriteData() : **stm3210e_eval.c**
- FSMC_BANK1NORSRAM4_WriteReg() : **stm3210e_eval.c**

- h -

- HAL_I2S_ErrorCallback() : **stm3210e_eval_audio.c**
- HAL_I2S_TxCpltCallback() : **stm3210e_eval_audio.c**
- HAL_I2S_TxHalfCpltCallback() : **stm3210e_eval_audio.c**
- HAL_NOR_MspWait() : **stm3210e_eval_nor.c**

- i -

- I2Cx_Error() : **stm3210e_eval.c**
- I2Cx_Init() : **stm3210e_eval.c**
- I2Cx_IsDeviceReady() : **stm3210e_eval.c**
- I2Cx_MspInit() : **stm3210e_eval.c**
- I2Cx_ReadBuffer() : **stm3210e_eval.c**
- I2Cx_ReadData() : **stm3210e_eval.c**
- I2Cx_WriteBuffer() : **stm3210e_eval.c**
- I2Cx_WriteData() : **stm3210e_eval.c**
- I2SOUT_Init() : **stm3210e_eval_audio.c**

- I2SOUT_MspInit() : **stm3210e_eval_audio.c**
- I2SOUT_SendDummyData_Start() : **stm3210e_eval_audio.c**
- I2SOUT_SendDummyData_Stop() : **stm3210e_eval_audio.c**

- l -

- LCD_Delay() : **stm3210e_eval.c**
- LCD_DrawChar() : **stm3210e_eval_lcd.c**
- LCD_DrawPixel() : **stm3210e_eval_lcd.c**
- LCD_IO_Init() : **stm3210e_eval.c**
- LCD_IO_ReadData() : **stm3210e_eval.c**
- LCD_IO_WriteData() : **stm3210e_eval.c**
- LCD_IO_WriteMultipleData() : **stm3210e_eval.c**
- LCD_IO_WriteReg() : **stm3210e_eval.c**
- LCD_SetDisplayWindow() : **stm3210e_eval_lcd.c**

- n -

- NAND_MspInit() : **stm3210e_eval_nand.c**
- NOR_MspInit() : **stm3210e_eval_nor.c**

- s -

- SD_DMAConfigRx() : **stm3210e_eval_sd.c**
- SD_DMAConfigTx() : **stm3210e_eval_sd.c**
- SD_MspInit() : **stm3210e_eval_sd.c**
- SPIx_Error() : **stm3210e_eval.c**
- SPIx_Init() : **stm3210e_eval.c**
- SPIx_MspInit() : **stm3210e_eval.c**
- SPIx_Read() : **stm3210e_eval.c**
- SPIx_Write() : **stm3210e_eval.c**
- SRAM_MspInit() : **stm3210e_eval_sram.c**

- t -

- TSENSOR_IO_Init() : **stm3210e_eval.c**
- TSENSOR_IO_IsDeviceReady() : **stm3210e_eval.c**
- TSENSOR_IO_Read() : **stm3210e_eval.c**

- TSENSOR_IO_Write() : **stm3210e_eval.c**

STM3210E_EVAL BSP User Manual

- a -

- audio_sample_SendDummyData : **stm3210e_eval_audio.c**

- b -

- bitmap : **stm3210e_eval_lcd.c**
- BUTTON_IRQn : **stm3210e_eval.c**
- BUTTON_PIN : **stm3210e_eval.c**
- BUTTON_PORT : **stm3210e_eval.c**

- c -

- COM_RX_PIN : **stm3210e_eval.c**
- COM_RX_PORT : **stm3210e_eval.c**
- COM_TX_PIN : **stm3210e_eval.c**
- COM_TX_PORT : **stm3210e_eval.c**
- COM_USART : **stm3210e_eval.c**

- d -

- DrawProp : **stm3210e_eval_lcd.c**

- h -

- hAudioOutI2s : **stm3210e_eval_audio.c**
- heval_I2c : **stm3210e_eval.c**
- heval_Spi : **stm3210e_eval.c**

- i -

- I2cxTimeout : **stm3210e_eval.c**

- j -

- JOY_IRQn : **stm3210e_eval.c**
- JOY_PIN : **stm3210e_eval.c**
- JOY_PORT : **stm3210e_eval.c**

- l -

- lcd_drv : **stm3210e_eval_lcd.c**
- LCD_SwapXY : **stm3210e_eval_lcd.c**
- LED_PIN : **stm3210e_eval.c**
- LED_PORT : **stm3210e_eval.c**

- n -

- nandHandle : **stm3210e_eval_nand.c**
- norHandle : **stm3210e_eval_nor.c**

- p -

- pAudioDrv : **stm3210e_eval_audio.c**

- s -

- SendDummyData : **stm3210e_eval_audio.c**
- SpixTimeout : **stm3210e_eval.c**
- sramHandle : **stm3210e_eval_sram.c**

- t -

- Timing : **stm3210e_eval_nand.c** , **stm3210e_eval_nor.c** , **stm3210e_eval_sram.c**
- tsensor_drv : **stm3210e_eval_tsensor.c**
- TSENSORAddress : **stm3210e_eval_tsensor.c**

- u -

- uSdCardInfo : **stm3210e_eval_sd.c**
- uSdHandle : **stm3210e_eval_sd.c**

STM3210E_EVAL BSP User Manual

- pPoint : **stm3210e_eval_lcd.h**

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

- Button_TypeDef : **stm3210e_eval.h**
- ButtonMode_TypeDef : **stm3210e_eval.h**
- COM_TypeDef : **stm3210e_eval.h**
- JOYMode_TypeDef : **stm3210e_eval.h**
- JOYState_TypeDef : **stm3210e_eval.h**
- Led_TypeDef : **stm3210e_eval.h**
- Line_ModeTypdef : **stm3210e_eval_lcd.h**
- TSENSOR_Status_TypDef : **stm3210e_eval_tsensor.h**

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

- b -

- BUTTON_DOWN : **stm3210e_eval.h**
- BUTTON_KEY : **stm3210e_eval.h**
- BUTTON_LEFT : **stm3210e_eval.h**
- BUTTON_MODE_EVT : **stm3210e_eval.h**
- BUTTON_MODE_EXTI : **stm3210e_eval.h**
- BUTTON_MODE_GPIO : **stm3210e_eval.h**
- BUTTON_RIGHT : **stm3210e_eval.h**
- BUTTON_SEL : **stm3210e_eval.h**
- BUTTON_TAMPER : **stm3210e_eval.h**
- BUTTON_UP : **stm3210e_eval.h**
- BUTTON_WAKEUP : **stm3210e_eval.h**

- c -

- CENTER_MODE : **stm3210e_eval_lcd.h**
- COM1 : **stm3210e_eval.h**
- COM2 : **stm3210e_eval.h**

- j -

- JOY_DOWN : **stm3210e_eval.h**
- JOY_LEFT : **stm3210e_eval.h**
- JOY_MODE_EXTI : **stm3210e_eval.h**
- JOY_MODE_GPIO : **stm3210e_eval.h**
- JOY_NONE : **stm3210e_eval.h**
- JOY_RIGHT : **stm3210e_eval.h**
- JOY_SEL : **stm3210e_eval.h**
- JOY_UP : **stm3210e_eval.h**

- l -

- LED1 : **stm3210e_eval.h**
- LED2 : **stm3210e_eval.h**
- LED3 : **stm3210e_eval.h**
- LED4 : **stm3210e_eval.h**
- LED_BLUE : **stm3210e_eval.h**
- LED_GREEN : **stm3210e_eval.h**
- LED_ORANGE : **stm3210e_eval.h**
- LED_RED : **stm3210e_eval.h**
- LEFT_MODE : **stm3210e_eval_lcd.h**

- r -

- RIGHT_MODE : **stm3210e_eval_lcd.h**

- t -

- TSENSOR_ERROR : **stm3210e_eval_tsensor.h**
- TSENSOR_OK : **stm3210e_eval_tsensor.h**

STM3210E_EVAL BSP User Manual

- _ -

- __DMAx_TxRx_CLK_ENABLE : **stm3210e_eval_sd.h**
- __SD_DETECT_GPIO_CLK_ENABLE : **stm3210e_eval_sd.h**
- __SRAM_DMAx_CLK_ENABLE : **stm3210e_eval_sram.h**
- __STM3210E_EVAL_BSP_VERSION : **stm3210e_eval.c**
- __STM3210E_EVAL_BSP_VERSION_MAIN : **stm3210e_eval.c**
- __STM3210E_EVAL_BSP_VERSION_RC : **stm3210e_eval.c**
- __STM3210E_EVAL_BSP_VERSION_SUB1 : **stm3210e_eval.c**
- __STM3210E_EVAL_BSP_VERSION_SUB2 : **stm3210e_eval.c**
- __STM3210e_EVAL_SRAM_H : **stm3210e_eval_sram.h**

STM3210E_EVAL BSP User Manual

- a -

- ABS : **stm3210e_eval_lcd.c**
- AUDIO_ERROR : **stm3210e_eval_audio.h**
- AUDIO_I2C_ADDRESS : **stm3210e_eval.h**
- AUDIO_OK : **stm3210e_eval_audio.h**
- AUDIO_OUT_IRQ_PREPRIO : **stm3210e_eval_audio.h**
- AUDIO_RESET_GPIO : **stm3210e_eval.h**
- AUDIO_RESET_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- AUDIO_RESET_PIN : **stm3210e_eval.h**
- AUDIO_SAMPLE_SENDDUMMYDATA_SIZE : **stm3210e_eval_audio.c**
- AUDIO_TIMEOUT : **stm3210e_eval_audio.h**

STM3210E_EVAL BSP User Manual

- b -

- BLOCKERASE_TIMEOUT : **stm3210e_eval_nor.h**
- BUTTONn : **stm3210e_eval.h**
- BUTTONx_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- BUTTONx_GPIO_CLK_ENABLE : **stm3210e_eval.h**

STM3210E_EVAL BSP User Manual

- c -

- CHIPERASE_TIMEOUT : **stm3210e_eval_nor.h**
- COMn : **stm3210e_eval.h**
- COMx_CLK_DISABLE : **stm3210e_eval.h**
- COMx_CLK_ENABLE : **stm3210e_eval.h**
- COMx_RX_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- COMx_RX_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- COMx_TX_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- COMx_TX_GPIO_CLK_ENABLE : **stm3210e_eval.h**

STM3210E_EVAL BSP User Manual

- d -

- DMA_MAX : **stm3210e_eval_audio.h**
- DMA_MAX_SZE : **stm3210e_eval_audio.h**
- DOWN_JOY_EXTI_IRQn : **stm3210e_eval.h**
- DOWN_JOY_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- DOWN_JOY_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- DOWN_JOY_GPIO_PORT : **stm3210e_eval.h**
- DOWN_JOY_PIN : **stm3210e_eval.h**

STM3210E_EVAL BSP User Manual

- e -

- EVAL_COM1 : **stm3210e_eval.h**
- EVAL_COM1_CLK_DISABLE : **stm3210e_eval.h**
- EVAL_COM1_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_COM1_IRQn : **stm3210e_eval.h**
- EVAL_COM1_RX_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- EVAL_COM1_RX_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_COM1_RX_GPIO_PORT : **stm3210e_eval.h**
- EVAL_COM1_RX_PIN : **stm3210e_eval.h**
- EVAL_COM1_TX_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- EVAL_COM1_TX_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_COM1_TX_GPIO_PORT : **stm3210e_eval.h**
- EVAL_COM1_TX_PIN : **stm3210e_eval.h**
- EVAL_COM2 : **stm3210e_eval.h**
- EVAL_COM2_CLK_DISABLE : **stm3210e_eval.h**
- EVAL_COM2_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_COM2_IRQn : **stm3210e_eval.h**
- EVAL_COM2_RX_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- EVAL_COM2_RX_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_COM2_RX_GPIO_PORT : **stm3210e_eval.h**
- EVAL_COM2_RX_PIN : **stm3210e_eval.h**

- EVAL_COM2_TX_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- EVAL_COM2_TX_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_COM2_TX_GPIO_PORT : **stm3210e_eval.h**
- EVAL_COM2_TX_PIN : **stm3210e_eval.h**
- EVAL_I2Cx : **stm3210e_eval.h**
- EVAL_I2Cx_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_I2Cx_ER_IRQHandler : **stm3210e_eval.h**
- EVAL_I2Cx_ER_IRQn : **stm3210e_eval.h**
- EVAL_I2Cx_EV_IRQHandler : **stm3210e_eval.h**
- EVAL_I2Cx_EV_IRQn : **stm3210e_eval.h**
- EVAL_I2Cx_FORCE_RESET : **stm3210e_eval.h**
- EVAL_I2Cx_RELEASE_RESET : **stm3210e_eval.h**
- EVAL_I2Cx_SCL_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_I2Cx_SCL_GPIO_PORT : **stm3210e_eval.h**
- EVAL_I2Cx_SCL_PIN : **stm3210e_eval.h**
- EVAL_I2Cx_SDA_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_I2Cx_SDA_GPIO_PORT : **stm3210e_eval.h**
- EVAL_I2Cx_SDA_PIN : **stm3210e_eval.h**
- EVAL_I2Cx_TIMEOUT_MAX : **stm3210e_eval.h**
- EVAL_SPIx : **stm3210e_eval.h**
- EVAL_SPIx_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_SPIx_MISO_MOSI_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_SPIx_MISO_MOSI_GPIO_PORT : **stm3210e_eval.h**
- EVAL_SPIx_MISO_PIN : **stm3210e_eval.h**
- EVAL_SPIx_MOSI_PIN : **stm3210e_eval.h**
- EVAL_SPIx_SCK_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- EVAL_SPIx_SCK_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- EVAL_SPIx_SCK_GPIO_PORT : **stm3210e_eval.h**
- EVAL_SPIx_SCK_PIN : **stm3210e_eval.h**
- EVAL_SPIx_TIMEOUT_MAX : **stm3210e_eval.h**

STM3210E_EVAL BSP User Manual

- f -

- FLASH_ERROR : **stm3210e_eval_serialflash.h**
- FLASH_OK : **stm3210e_eval_serialflash.h**
- FLASH_SPI_CMD_BE : **stm3210e_eval.h**
- FLASH_SPI_CMD_RDID : **stm3210e_eval.h**
- FLASH_SPI_CMD_RDSR : **stm3210e_eval.h**
- FLASH_SPI_CMD_READ : **stm3210e_eval.h**
- FLASH_SPI_CMD_SE : **stm3210e_eval.h**
- FLASH_SPI_CMD_WREN : **stm3210e_eval.h**
- FLASH_SPI_CMD_WRITE : **stm3210e_eval.h**
- FLASH_SPI_CMD_WRSR : **stm3210e_eval.h**
- FLASH_SPI_CS_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- FLASH_SPI_CS_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- FLASH_SPI_CS_GPIO_PORT : **stm3210e_eval.h**
- FLASH_SPI_CS_HIGH : **stm3210e_eval.h**
- FLASH_SPI_CS_LOW : **stm3210e_eval.h**
- FLASH_SPI_CS_PIN : **stm3210e_eval.h**
- FLASH_SPI_DUMMY_BYTE : **stm3210e_eval.h**
- FLASH_SPI_M25P128_ID : **stm3210e_eval.h**
- FLASH_SPI_M25P64_ID : **stm3210e_eval.h**
- FLASH_SPI_PAGESIZE : **stm3210e_eval.h**

- FLASH_SPI_WIP_FLAG : **stm3210e_eval.h**

STM3210E_EVAL BSP User Manual

- i -

- I2SOUT : **stm3210e_eval_audio.h**
- I2SOUT_CLK_ENABLE : **stm3210e_eval_audio.h**
- I2SOUT_DMAx_CHANNEL : **stm3210e_eval_audio.h**
- I2SOUT_DMAx_CLK_ENABLE : **stm3210e_eval_audio.h**
- I2SOUT_DMAx_IRQ : **stm3210e_eval_audio.h**
- I2SOUT_DMAx_MEM_DATA_SIZE : **stm3210e_eval_audio.h**
- I2SOUT_DMAx_PERIPH_DATA_SIZE : **stm3210e_eval_audio.h**
- I2SOUT_IRQHandler : **stm3210e_eval_audio.h**
- I2SOUT_MCK_CLK_ENABLE : **stm3210e_eval_audio.h**
- I2SOUT_MCK_GPIO_PORT : **stm3210e_eval_audio.h**
- I2SOUT_MCK_PIN : **stm3210e_eval_audio.h**
- I2SOUT_SCK_PIN : **stm3210e_eval_audio.h**
- I2SOUT_SCK_SD_CLK_ENABLE : **stm3210e_eval_audio.h**
- I2SOUT_SCK_SD_GPIO_PORT : **stm3210e_eval_audio.h**
- I2SOUT_SD_PIN : **stm3210e_eval_audio.h**
- I2SOUT_WS_CLK_ENABLE : **stm3210e_eval_audio.h**
- I2SOUT_WS_GPIO_PORT : **stm3210e_eval_audio.h**
- I2SOUT_WS_PIN : **stm3210e_eval_audio.h**

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP
User Manual by  1.7.6.1

STM3210E_EVAL BSP User Manual

- j -

- JOYn : **stm3210e_eval.h**
- JOYx_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- JOYx_GPIO_CLK_ENABLE : **stm3210e_eval.h**

STM3210E_EVAL BSP User Manual

- k -

- KEY_BUTTON_EXTI_IRQn : **stm3210e_eval.h**
- KEY_BUTTON_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- KEY_BUTTON_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- KEY_BUTTON_GPIO_PORT : **stm3210e_eval.h**
- KEY_BUTTON_PIN : **stm3210e_eval.h**

STM3210E_EVAL BSP User Manual

- l -

- LCD_COLOR_BLACK : **stm3210e_eval_lcd.h**
- LCD_COLOR_BLUE : **stm3210e_eval_lcd.h**
- LCD_COLOR_BROWN : **stm3210e_eval_lcd.h**
- LCD_COLOR_CYAN : **stm3210e_eval_lcd.h**
- LCD_COLOR_DARKBLUE : **stm3210e_eval_lcd.h**
- LCD_COLOR_DARKCYAN : **stm3210e_eval_lcd.h**
- LCD_COLOR_DARKGRAY : **stm3210e_eval_lcd.h**
- LCD_COLOR_DARKGREEN : **stm3210e_eval_lcd.h**
- LCD_COLOR_DARKMAGENTA : **stm3210e_eval_lcd.h**
- LCD_COLOR_DARKRED : **stm3210e_eval_lcd.h**
- LCD_COLOR_DARKYELLOW : **stm3210e_eval_lcd.h**
- LCD_COLOR_GRAY : **stm3210e_eval_lcd.h**
- LCD_COLOR_GREEN : **stm3210e_eval_lcd.h**
- LCD_COLOR_LIGHTBLUE : **stm3210e_eval_lcd.h**
- LCD_COLOR_LIGHTCYAN : **stm3210e_eval_lcd.h**
- LCD_COLOR_LIGHTGRAY : **stm3210e_eval_lcd.h**
- LCD_COLOR_LIGHTGREEN : **stm3210e_eval_lcd.h**
- LCD_COLOR_LIGHTMAGENTA : **stm3210e_eval_lcd.h**
- LCD_COLOR_LIGHTRED : **stm3210e_eval_lcd.h**
- LCD_COLOR_LIGHTYELLOW : **stm3210e_eval_lcd.h**

- LCD_COLOR_MAGENTA : **stm3210e_eval_lcd.h**
- LCD_COLOR_ORANGE : **stm3210e_eval_lcd.h**
- LCD_COLOR_RED : **stm3210e_eval_lcd.h**
- LCD_COLOR_WHITE : **stm3210e_eval_lcd.h**
- LCD_COLOR_YELLOW : **stm3210e_eval_lcd.h**
- LCD_DEFAULT_FONT : **stm3210e_eval_lcd.h**
- LCD_ERROR : **stm3210e_eval_lcd.h**
- LCD_OK : **stm3210e_eval_lcd.h**
- LCD_TIMEOUT : **stm3210e_eval_lcd.h**
- LED1_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- LED1_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- LED1_GPIO_PORT : **stm3210e_eval.h**
- LED1_PIN : **stm3210e_eval.h**
- LED2_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- LED2_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- LED2_GPIO_PORT : **stm3210e_eval.h**
- LED2_PIN : **stm3210e_eval.h**
- LED3_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- LED3_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- LED3_GPIO_PORT : **stm3210e_eval.h**
- LED3_PIN : **stm3210e_eval.h**
- LED4_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- LED4_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- LED4_GPIO_PORT : **stm3210e_eval.h**
- LED4_PIN : **stm3210e_eval.h**
- LEDn : **stm3210e_eval.h**
- LEDx_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- LEDx_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- LEFT_JOY_EXTI_IRQn : **stm3210e_eval.h**
- LEFT_JOY_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- LEFT_JOY_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- LEFT_JOY_GPIO_PORT : **stm3210e_eval.h**
- LEFT_JOY_PIN : **stm3210e_eval.h**

STM3210E_EVAL BSP User Manual

- m -

- MAX_HEIGHT_FONT : **stm3210e_eval_lcd.c**
- MAX_WIDTH_FONT : **stm3210e_eval_lcd.c**
- MSD_ERROR : **stm3210e_eval_sd.h**
- MSD_OK : **stm3210e_eval_sd.h**

STM3210E_EVAL BSP User Manual

- n -

- NAND_BLOCK_SIZE : **stm3210e_eval_nand.h**
- NAND_DEVICE_ADDR : **stm3210e_eval_nand.h**
- NAND_MAX_ZONE : **stm3210e_eval_nand.h**
- NAND_OK : **stm3210e_eval_nand.h**
- NAND_PAGE_SIZE : **stm3210e_eval_nand.h**
- NAND_SPARE_AREA_SIZE : **stm3210e_eval_nand.h**
- NAND_ZONE_SIZE : **stm3210e_eval_nand.h**
- NOR_BURSTACCESS : **stm3210e_eval_nor.h**
- NOR_BUSY_STATE : **stm3210e_eval_nor.h**
- NOR_DEVICE_ADDR : **stm3210e_eval_nor.h**
- NOR_MEMORY_WIDTH : **stm3210e_eval_nor.h**
- NOR_READY_BUSY_GPIO : **stm3210e_eval_nor.h**
- NOR_READY_BUSY_PIN : **stm3210e_eval_nor.h**
- NOR_READY_STATE : **stm3210e_eval_nor.h**
- NOR_STATUS_ERROR : **stm3210e_eval_nor.h**
- NOR_STATUS_OK : **stm3210e_eval_nor.h**
- NOR_WRITEBURST : **stm3210e_eval_nor.h**

User Manual by  1.7.6.1

STM3210E_EVAL BSP User Manual

- o -

- OFFSET_BITMAP : **stm3210e_eval_lcd.c**

STM3210E_EVAL BSP User Manual

- p -

- POLY_X : **stm3210e_eval_lcd.c**
- POLY_Y : **stm3210e_eval_lcd.c**
- PROGRAM_TIMEOUT : **stm3210e_eval_nor.h**

STM3210E_EVAL BSP User Manual

- r -

- RIGHT_JOY_EXTI_IRQn : **stm3210e_eval.h**
- RIGHT_JOY_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- RIGHT_JOY_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- RIGHT_JOY_GPIO_PORT : **stm3210e_eval.h**
- RIGHT_JOY_PIN : **stm3210e_eval.h**

STM3210E_EVAL BSP User Manual

- s -

- SD_CardInfo : **stm3210e_eval_sd.h**
- SD_DATATIMEOUT : **stm3210e_eval_sd.h**
- SD_DETECT_GPIO_PORT : **stm3210e_eval_sd.h**
- SD_DETECT_IRQn : **stm3210e_eval_sd.h**
- SD_DETECT_PIN : **stm3210e_eval_sd.h**
- SD_DMAx_Rx_INSTANCE : **stm3210e_eval_sd.h**
- SD_DMAx_Rx_IRQHandler : **stm3210e_eval_sd.h**
- SD_DMAx_Rx_IRQn : **stm3210e_eval_sd.h**
- SD_DMAx_Tx_INSTANCE : **stm3210e_eval_sd.h**
- SD_DMAx_Tx_IRQHandler : **stm3210e_eval_sd.h**
- SD_DMAx_Tx_IRQn : **stm3210e_eval_sd.h**
- SD_NOT_PRESENT : **stm3210e_eval_sd.h**
- SD_PRESENT : **stm3210e_eval_sd.h**
- SEL_JOY_EXTI_IRQn : **stm3210e_eval.h**
- SEL_JOY_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- SEL_JOY_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- SEL_JOY_GPIO_PORT : **stm3210e_eval.h**
- SEL_JOY_PIN : **stm3210e_eval.h**
- SRAM_BURSTACCESS : **stm3210e_eval_sram.h**
- SRAM_DEVICE_ADDR : **stm3210e_eval_sram.h**

- SRAM_DEVICE_SIZE : **stm3210e_eval_sram.h**
- SRAM_DMAx_IRQHandler : **stm3210e_eval_sram.h**
- SRAM_DMAx_IRQn : **stm3210e_eval_sram.h**
- SRAM_DMAx_STREAM : **stm3210e_eval_sram.h**
- SRAM_ERROR : **stm3210e_eval_sram.h**
- SRAM_MEMORY_WIDTH : **stm3210e_eval_sram.h**
- SRAM_OK : **stm3210e_eval_sram.h**
- SRAM_WRITEBURST : **stm3210e_eval_sram.h**

STM3210E_EVAL BSP User Manual

- t -

- TAMPER_BUTTON_EXTI_IRQn : **stm3210e_eval.h**
- TAMPER_BUTTON_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- TAMPER_BUTTON_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- TAMPER_BUTTON_GPIO_PORT : **stm3210e_eval.h**
- TAMPER_BUTTON_PIN : **stm3210e_eval.h**
- TFT_LCD : **stm3210e_eval.c**
- TFT_LCD_BASE : **stm3210e_eval.c**
- TSENSOR_I2C_ADDRESS_A01 : **stm3210e_eval_tsensor.h**
- TSENSOR_I2C_ADDRESS_A02 : **stm3210e_eval_tsensor.h**
- TSENSOR_MAX_TRIALS : **stm3210e_eval_tsensor.h**

STM3210E_EVAL BSP User Manual

- u -

- UP_JOY_EXTI_IRQn : **stm3210e_eval.h**
- UP_JOY_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- UP_JOY_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- UP_JOY_GPIO_PORT : **stm3210e_eval.h**
- UP_JOY_PIN : **stm3210e_eval.h**

STM3210E_EVAL BSP User Manual

- w -

- WAKEUP_BUTTON_EXTI_IRQn : **stm3210e_eval.h**
- WAKEUP_BUTTON_GPIO_CLK_DISABLE : **stm3210e_eval.h**
- WAKEUP_BUTTON_GPIO_CLK_ENABLE : **stm3210e_eval.h**
- WAKEUP_BUTTON_GPIO_PORT : **stm3210e_eval.h**
- WAKEUP_BUTTON_PIN : **stm3210e_eval.h**

STM3210E_EVAL BSP User Manual

stm3210e_eval.c File Reference

This file provides a set of firmware functions to manage Leds, push-button and COM ports for STM3210E_EVAL. More...

```
#include "stm3210e_eval.h"
```

Go to the source code of this file.

Data Structures

struct	**LCD_CONTROLLER_TypeDef**

Defines

#define	**__STM3210E_EVAL_BSP_VERSION_MAIN** (0x06) STM3210E EVAL BSP Driver version number V6.0.0.
#define	**__STM3210E_EVAL_BSP_VERSION_SUB1** (0x00)
#define	**__STM3210E_EVAL_BSP_VERSION_SUB2** (0x01)
#define	**__STM3210E_EVAL_BSP_VERSION_RC** (0x00)
#define	**__STM3210E_EVAL_BSP_VERSION**
#define	**TFT_LCD_BASE** FSMC_BANK1_4
#define	**TFT_LCD** ((**LCD_CONTROLLER_TypeDef** *) **TFT_LCD_BASE**)

Functions

static void	**FSMC_BANK1NORSRAM4_WriteData** (uint16_t Data) Writes register value.
static void	**FSMC_BANK1NORSRAM4_WriteReg** (uint8_t Reg) Writes register address.
static uint16_t	**FSMC_BANK1NORSRAM4_ReadData** (uint8_t Reg) Reads register value.
static void	**FSMC_BANK1NORSRAM4_Init** (void) Initializes LCD IO.
static void	**FSMC_BANK1NORSRAM4_MspInit** (void) Initializes FSMC_BANK4 MSP.
void	**LCD_IO_Init** (void) Initializes LCD low level.
void	**LCD_IO_WriteData** (uint16_t RegValue) Writes data on LCD data register.
void	**LCD_IO_WriteMultipleData** (uint8_t *pData, uint32_t Size) Writes multiple data on LCD data register.
void	**LCD_IO_WriteReg** (uint8_t Reg) Writes register on LCD register.
uint16_t	**LCD_IO_ReadData** (uint16_t Reg) Reads data from LCD data register.
void	**LCD_Delay** (uint32_t Delay) Wait for loop in ms.
static void	**I2Cx_Init** (void) Eval I2Cx Bus initialization.
static void	**I2Cx_WriteData** (uint16_t Addr, uint8_t Reg, uint8_t Value)

	Write a value in a register of the device through BUS.
static HAL_StatusTypeDef	**I2Cx_WriteBuffer** (uint16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuffer, uint16_t Length) Write a value in a register of the device through BUS.
static uint8_t	**I2Cx_ReadData** (uint16_t Addr, uint8_t Reg) Read a value in a register of the device through BUS.
static HAL_StatusTypeDef	**I2Cx_ReadBuffer** (uint16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuffer, uint16_t Length) Reads multiple data on the BUS.
static HAL_StatusTypeDef	**I2Cx_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.
static void	**I2Cx_Error** (void) Eval I2Cx error treatment function.
static void	**I2Cx_MspInit** (I2C_HandleTypeDef *hi2c) Eval I2Cx MSP Initialization.
void	**TSENSOR_IO_Init** (void) Initializes peripherals used by the I2C Temperature Sensor driver.
void	**TSENSOR_IO_Write** (uint16_t DevAddress, uint8_t *pBuffer, uint8_t WriteAddr, uint16_t Length) Writes one byte to the TSENSOR.
void	**TSENSOR_IO_Read** (uint16_t DevAddress, uint8_t *pBuffer, uint8_t ReadAddr, uint16_t Length) Reads one byte from the TSENSOR.

uint16_t	**TSENSOR_IO_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if Temperature Sensor is ready for communication.
void	**AUDIO_IO_Init** (void) Initializes Audio low level.
void	**AUDIO_IO_Write** (uint8_t Addr, uint8_t Reg, uint8_t Value) Writes a single data.
uint8_t	**AUDIO_IO_Read** (uint8_t Addr, uint8_t Reg) Reads a single data.
static HAL_StatusTypeDef	**SPIx_Init** (void) Initializes SPI HAL.
static uint8_t	**SPIx_Write** (uint8_t WriteValue) SPI Write a byte to device.
static uint8_t	**SPIx_Read** (void) SPI Read 1 byte from device.
static void	**SPIx_Error** (void) SPI error treatment function.
static void	**SPIx_MspInit** (SPI_HandleTypeDef *hspi) Initializes SPI MSP.
HAL_StatusTypeDef	**FLASH_SPI_IO_Init** (void) Initializes the FLASH SPI and put it into StandBy State (Ready for data transfer).
uint8_t	**FLASH_SPI_IO_WriteByte** (uint8_t Data) Write a byte on the FLASH SPI.
uint8_t	**FLASH_SPI_IO_ReadByte** (void) Read a byte from the FLASH SPI.
HAL_StatusTypeDef	**FLASH_SPI_IO_ReadData** (uint32_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Read data from FLASH SPI driver.

void	**FLASH_SPI_IO_WriteEnable** (void) Select the FLASH SPI and send "Write Enable" instruction.
HAL_StatusTypeDef	**FLASH_SPI_IO_WaitForWriteEnd** (void) Wait response from the FLASH SPI and Deselect the device.
uint32_t	**FLASH_SPI_IO_ReadID** (void) Reads FLASH SPI identification.
uint32_t	**BSP_GetVersion** (void) This method returns the STM3210E EVAL BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LED GPIO.
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** Button_Mode) Configures push button GPIO and EXTI Line.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
uint8_t	**BSP_JOY_Init** (**JOYMode_TypeDef** Joy_Mode) Configures all button of the joystick in GPIO or EXTI modes.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the current joystick status.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *huart)

Configures COM port.

Variables

GPIO_TypeDef *	**LED_PORT** [**LEDn**] LED variables.
const uint16_t	**LED_PIN** [**LEDn**]
GPIO_TypeDef *	**BUTTON_PORT** [**BUTTONn**] BUTTON variables.
const uint16_t	**BUTTON_PIN** [**BUTTONn**]
const uint8_t	**BUTTON_IRQn** [**BUTTONn**]
GPIO_TypeDef *	**JOY_PORT** [**JOYn**] JOYSTICK variables.
const uint16_t	**JOY_PIN** [**JOYn**]
const uint8_t	**JOY_IRQn** [**JOYn**]
USART_TypeDef *	**COM_USART** [**COMn**] = {**EVAL_COM1**, **EVAL_COM2**} COM variables.
GPIO_TypeDef *	**COM_TX_PORT** [**COMn**] = {**EVAL_COM1_TX_GPIO_PORT**, **EVAL_COM2_TX_GPIO_PORT**}
GPIO_TypeDef *	**COM_RX_PORT** [**COMn**] = {**EVAL_COM1_RX_GPIO_PORT**, **EVAL_COM2_RX_GPIO_PORT**}
const uint16_t	**COM_TX_PIN** [**COMn**] = {**EVAL_COM1_TX_PIN**, **EVAL_COM2_TX_PIN**}
const uint16_t	**COM_RX_PIN** [**COMn**] = {**EVAL_COM1_RX_PIN**, **EVAL_COM2_RX_PIN**}
uint32_t	**SpixTimeout** = **EVAL_SPIx_TIMEOUT_MAX** BUS variables.
static SPI_HandleTypeDef	**heval_Spi**
uint32_t	**I2cxTimeout** = **EVAL_I2Cx_TIMEOUT_MAX**

I2C_HandleTypeDef **heval_I2c**

Detailed Description

This file provides a set of firmware functions to manage Leds, push-button and COM ports for STM3210E_EVAL.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

© COPYRIGHT(c) 2015 STMicroelectronics

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

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

Definition in file **stm3210e_eval.c**.

STM3210E_EVAL BSP User Manual

Defines | Enumerations | Functions

stm3210e_eval.h File Reference

This file contains definitions for STM3210E_EVAL's LEDs, push-buttons and COM ports hardware resources. More...

```
#include "stm32f1xx_hal.h"
```

Go to the source code of this file.

Defines

#define **LEDn** 4

#define **LED1_PIN** GPIO_PIN_6 /* PF.06*/

#define **LED1_GPIO_PORT** GPIOF

#define **LED1_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOF_CLK_EI

#define **LED1_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOF_CLK_D

#define **LED2_PIN** GPIO_PIN_7 /* PF.07*/

#define **LED2_GPIO_PORT** GPIOF

#define **LED2_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOF_CLK_EI

#define **LED2_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOF_CLK_D

#define **LED3_PIN** GPIO_PIN_8 /* PF.08*/

#define **LED3_GPIO_PORT** GPIOF

#define **LED3_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOF_CLK_EI

#define **LED3_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOF_CLK_D

#define **LED4_PIN** GPIO_PIN_9 /* PF.09*/

#define **LED4_GPIO_PORT** GPIOF

#define **LED4_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOF_CLK_EI

#define **LED4_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOF_CLK_D

#define **LEDx_GPIO_CLK_ENABLE**(__LED__)

#define **LEDx_GPIO_CLK_DISABLE**(__LED__)

#define **JOYn** 5

#define **BUTTONn** 3 + **JOYn**

#define **TAMPER_BUTTON_PIN** GPIO_PIN_13 /* PC.13*/

Tamper push-button.

#define **TAMPER_BUTTON_GPIO_PORT** GPIOC

#define **TAMPER_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_G

#define **TAMPER_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_C

#define **TAMPER_BUTTON_EXTI_IRQn** EXTI15_10_IRQn

#define **KEY_BUTTON_PIN** GPIO_PIN_8 /* PG.08*/

Key push-button.

#define **KEY_BUTTON_GPIO_PORT** GPIOG

#define **KEY_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOC

#define **KEY_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_GPIO

#define **KEY_BUTTON_EXTI_IRQn** EXTI9_5_IRQn

#define **WAKEUP_BUTTON_PIN** GPIO_PIN_0 /* PA.00*/
Wake-up push-button.

#define **WAKEUP_BUTTON_GPIO_PORT** GPIOA

#define **WAKEUP_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_G

#define **WAKEUP_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_(

#define **WAKEUP_BUTTON_EXTI_IRQn** EXTI0_IRQn

#define **RIGHT_JOY_PIN** GPIO_PIN_13 /* PG.13*/
Joystick Right push-button.

#define **RIGHT_JOY_GPIO_PORT** GPIOG

#define **RIGHT_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOG_

#define **RIGHT_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOG_

#define **RIGHT_JOY_EXTI_IRQn** EXTI15_10_IRQn

#define **LEFT_JOY_PIN** GPIO_PIN_14 /* PG.14*/
Joystick Left push-button.

#define **LEFT_JOY_GPIO_PORT** GPIOG

#define **LEFT_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOG_C

#define **LEFT_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOG_C

#define **LEFT_JOY_EXTI_IRQn** EXTI15_10_IRQn

#define **UP_JOY_PIN** GPIO_PIN_15 /* PG.15*/
Joystick Up push-button.

#define **UP_JOY_GPIO_PORT** GPIOG

#define **UP_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOG_CLK

#define **UP_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOG_CL

#define **UP_JOY_EXTI_IRQn** EXTI15_10_IRQn

#define **DOWN_JOY_PIN** GPIO_PIN_3 /* PD.03*/
Joystick Down push-button.

#define **DOWN_JOY_GPIO_PORT** GPIOD

#define **DOWN_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOD_

#define **DOWN_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOD_

#define **DOWN_JOY_EXTI_IRQn** EXTI3_IRQn

#define **SEL_JOY_PIN** GPIO_PIN_7 /* PG.07*/
Joystick Sel push-button.

#define **SEL_JOY_GPIO_PORT** GPIOG

#define **SEL_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOG_CL

#define **SEL_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOG_CL

#define **SEL_JOY_EXTI_IRQn** EXTI9_5_IRQn

#define **BUTTONx_GPIO_CLK_ENABLE**(__BUTTON__)

#define **BUTTONx_GPIO_CLK_DISABLE**(__BUTTON__)

#define **JOYx_GPIO_CLK_ENABLE**(__JOY__)

#define **JOYx_GPIO_CLK_DISABLE**(__JOY__)

#define **COMn** 2

#define **EVAL_COM1** USART1

Definition for COM port1, connected to USART1.

#define **EVAL_COM1_CLK_ENABLE**() __HAL_RCC_USART1_CLK_

#define **EVAL_COM1_CLK_DISABLE**() __HAL_RCC_USART1_CLK

#define **EVAL_COM1_TX_PIN** GPIO_PIN_9 /* PA.09*/

#define **EVAL_COM1_TX_GPIO_PORT** GPIOA

#define **EVAL_COM1_TX_GPIO_CLK_ENABLE**() __HAL_RCC_GPI

#define **EVAL_COM1_TX_GPIO_CLK_DISABLE**() __HAL_RCC_GP

#define **EVAL_COM1_RX_PIN** GPIO_PIN_10 /* PA.10*/

#define **EVAL_COM1_RX_GPIO_PORT** GPIOA

#define **EVAL_COM1_RX_GPIO_CLK_ENABLE**() __HAL_RCC_GPI

#define **EVAL_COM1_RX_GPIO_CLK_DISABLE**() __HAL_RCC_GP

#define **EVAL_COM1_IRQn** USART1_IRQn

#define **EVAL_COM2** USART2

Definition for COM port2, connected to USART2.

#define **EVAL_COM2_CLK_ENABLE**() __HAL_RCC_USART2_CLK_

#define **EVAL_COM2_CLK_DISABLE**() __HAL_RCC_USART2_CLK

#define **EVAL_COM2_TX_PIN** GPIO_PIN_2 /* PA.02*/

#define **EVAL_COM2_TX_GPIO_PORT** GPIOA

#define **EVAL_COM2_TX_GPIO_CLK_ENABLE**() __HAL_RCC_GPI

#define **EVAL_COM2_TX_GPIO_CLK_DISABLE**() __HAL_RCC_GP

#define **EVAL_COM2_RX_PIN** GPIO_PIN_3 /* PA.03*/

#define **EVAL_COM2_RX_GPIO_PORT** GPIOA

#define **EVAL_COM2_RX_GPIO_CLK_ENABLE**() __HAL_RCC_GPI

#define **EVAL_COM2_RX_GPIO_CLK_DISABLE**() __HAL_RCC_GP

```
#define EVAL_COM2_IRQn USART2_IRQn
#define COMx_CLK_ENABLE(__INDEX__)
#define COMx_CLK_DISABLE(__INDEX__)
#define COMx_TX_GPIO_CLK_ENABLE(__INDEX__)
#define COMx_TX_GPIO_CLK_DISABLE(__INDEX__)
#define COMx_RX_GPIO_CLK_ENABLE(__INDEX__)
#define COMx_RX_GPIO_CLK_DISABLE(__INDEX__)
#define EVAL_I2Cx_SCL_PIN GPIO_PIN_6 /* PB.06*/
#define EVAL_I2Cx_SCL_GPIO_PORT GPIOB
#define EVAL_I2Cx_SDA_PIN GPIO_PIN_7 /* PB.07*/
#define EVAL_I2Cx_SDA_GPIO_PORT GPIOB
#define EVAL_I2Cx I2C1
#define EVAL_I2Cx_CLK_ENABLE() __HAL_RCC_I2C1_CLK_ENAI
#define EVAL_I2Cx_SDA_GPIO_CLK_ENABLE() __HAL_RCC_GPI
#define EVAL_I2Cx_SCL_GPIO_CLK_ENABLE() __HAL_RCC_GPI
#define EVAL_I2Cx_FORCE_RESET() __HAL_RCC_I2C1_FORCE_
#define EVAL_I2Cx_RELEASE_RESET() __HAL_RCC_I2C1_RELE/
#define EVAL_I2Cx_EV_IRQn I2C1_EV_IRQn
#define EVAL_I2Cx_EV_IRQHandler I2C1_EV_IRQHandler
#define EVAL_I2Cx_ER_IRQn I2C1_ER_IRQn
#define EVAL_I2Cx_ER_IRQHandler I2C1_ER_IRQHandler
#define EVAL_I2Cx_TIMEOUT_MAX 3000
#define EVAL_SPIx SPI1
#define EVAL_SPIx_CLK_ENABLE() __HAL_RCC_SPI1_CLK_ENA
#define EVAL_SPIx_SCK_GPIO_PORT GPIOA /* PA.05*/
#define EVAL_SPIx_SCK_PIN GPIO_PIN_5
#define EVAL_SPIx_SCK_GPIO_CLK_ENABLE() __HAL_RCC_GPI
#define EVAL_SPIx_SCK_GPIO_CLK_DISABLE() __HAL_RCC_GP
#define EVAL_SPIx_MISO_MOSI_GPIO_PORT GPIOA
#define EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE() __HAL_R
#define EVAL_SPIx_MISO_MOSI_GPIO_CLK_DISABLE() __HAL_F
#define EVAL_SPIx_MISO_PIN GPIO_PIN_6 /* PA.06*/
#define EVAL_SPIx_MOSI_PIN GPIO_PIN_7 /* PA.07*/
#define EVAL_SPIx_TIMEOUT_MAX 1000
```

#define	**FLASH_SPI_CS_LOW**() HAL_GPIO_WritePin(**FLASH_SPI_** **FLASH_SPI_CS_PIN**, GPIO_PIN_RESET) M25P FLASH SPI Chip Select macro definition.
#define	**FLASH_SPI_CS_HIGH**() HAL_GPIO_WritePin(**FLASH_SPI_** **FLASH_SPI_CS_PIN**, GPIO_PIN_SET)
#define	**FLASH_SPI_CS_PIN** GPIO_PIN_2 /* PB.02*/ M25P FLASH SPI Control Interface pins.
#define	**FLASH_SPI_CS_GPIO_PORT** GPIOB
#define	**FLASH_SPI_CS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIO
#define	**FLASH_SPI_CS_GPIO_CLK_DISABLE**() __HAL_RCC_GPI
#define	**FLASH_SPI_CMD_WRITE** 0x02 M25P FLASH SPI supported commands.
#define	**FLASH_SPI_CMD_WRSR** 0x01
#define	**FLASH_SPI_CMD_WREN** 0x06
#define	**FLASH_SPI_CMD_READ** 0x03
#define	**FLASH_SPI_CMD_RDSR** 0x05
#define	**FLASH_SPI_CMD_RDID** 0x9F
#define	**FLASH_SPI_CMD_SE** 0xD8
#define	**FLASH_SPI_CMD_BE** 0xC7
#define	**FLASH_SPI_WIP_FLAG** 0x01
#define	**FLASH_SPI_DUMMY_BYTE** 0xA5
#define	**FLASH_SPI_PAGESIZE** 0x100
#define	**FLASH_SPI_M25P128_ID** 0x202018
#define	**FLASH_SPI_M25P64_ID** 0x202017
#define	**AUDIO_I2C_ADDRESS** 0x27 AUDIO I2C Interface pins.
#define	**AUDIO_RESET_GPIO_CLK_ENABLE**() __HAL_RCC_GPIO
#define	**AUDIO_RESET_PIN** GPIO_PIN_11
#define	**AUDIO_RESET_GPIO** GPIOG

Enumerations

enum	**Led_TypeDef** {
	LED1 = 0, **LED2** = 1, **LED3** = 2, **LED4** = 3,
	LED_GREEN = LED1, **LED_ORANGE** = LED2, **LED_RED** = LED3, **LED_BLUE** = LED4
	}
	LED Types Definition. More...

enum	**Button_TypeDef** {
	BUTTON_WAKEUP = 0, **BUTTON_TAMPER** = 1,
	BUTTON_KEY = 2, **BUTTON_SEL** = 3,
	BUTTON_LEFT = 4, **BUTTON_RIGHT** = 5,
	BUTTON_DOWN = 6, **BUTTON_UP** = 7
	}
	BUTTON Types Definition. More...

enum	**ButtonMode_TypeDef** { **BUTTON_MODE_GPIO** = 0, **BUTTON_MODE_EXTI** = 1, **BUTTON_MODE_EVT** = 2 }

enum	**JOYState_TypeDef** {
	JOY_SEL = 0, **JOY_LEFT** = 1, **JOY_RIGHT** = 2,
	JOY_DOWN = 3,
	JOY_UP = 4, **JOY_NONE** = 5
	}
	JOYSTICK Types Definition. More...

enum	**JOYMode_TypeDef** { **JOY_MODE_GPIO** = 0, **JOY_MODE_EXTI** = 1 }
enum	**COM_TypeDef** { **COM1** = 0, **COM2** = 1 } COM Types Definition. More...

Functions

uint32_t	**BSP_GetVersion** (void) This method returns the STM3210E EVAL BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LED GPIO.
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** Button_Mode) Configures push button GPIO and EXTI Line.
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.
uint8_t	**BSP_JOY_Init** (**JOYMode_TypeDef** Joy_Mode) Configures all button of the joystick in GPIO or EXTI modes.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the current joystick status.

Detailed Description

This file contains definitions for STM3210E_EVAL's LEDs, push-buttons and COM ports hardware resources.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

© COPYRIGHT(c) 2015 STMicroelectronics

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

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

Definition in file **stm3210e_eval.h**.

STM3210E_EVAL BSP User Manual

stm3210e_eval_audio.c File Reference

This file provides the Audio driver for the STM3210E-Eval board.
More...

```
#include "stm32f1xx_hal.h" #include "stm3210e_eval_audio.h"
```

Go to the source code of this file.

Defines

```
#define AUDIO_SAMPLE_SENDDUMMYDATA_SIZE 8
```

Functions

static void	**I2SOUT_MspInit** (void) AUDIO OUT I2S MSP Init.
static void	**I2SOUT_Init** (uint32_t AudioFreq) Initializes the Audio Codec audio interface (I2S)
static uint8_t	**I2SOUT_SendDummyData_Start** (uint16_t *pBuffer, uint32_t Size) Starts sending dummy data as audio stream: function used to generate I2S master clock MCK for audio codec device.
static uint8_t	**I2SOUT_SendDummyData_Stop** (void) Stops sending dummy data as audio stream: function used to stop to generate I2S master clock MCK for audio codec device.
uint8_t	**BSP_AUDIO_OUT_Init** (uint16_t OutputDevice, uint8_t Volume, uint32_t AudioFreq) Configure the audio peripherals.
uint8_t	**BSP_AUDIO_OUT_Play** (uint16_t *pBuffer, uint32_t Size) Starts playing audio stream from a data buffer for a determined size.
void	**BSP_AUDIO_OUT_ChangeBuffer** (uint16_t *pData, uint16_t Size) Sends n-Bytes on the I2S interface.
uint8_t	**BSP_AUDIO_OUT_Pause** (void) This function Pauses the audio file stream.
uint8_t	**BSP_AUDIO_OUT_Resume** (void) This function Resumes the audio file stream.
uint8_t	**BSP_AUDIO_OUT_Stop** (uint32_t Option) Stops audio playing and Power down the Audio Codec.
uint8_t	**BSP_AUDIO_OUT_SetVolume** (uint8_t Volume) Controls the current audio volume level.

uint8_t	**BSP_AUDIO_OUT_SetMute** (uint32_t Cmd) Enables or disables the MUTE mode by software.
uint8_t	**BSP_AUDIO_OUT_SetOutputMode** (uint8_t Output) Switch dynamically (while audio file is played) the output target (speaker or headphone).
void	**BSP_AUDIO_OUT_SetFrequency** (uint32_t AudioFreq) Update the audio frequency.
void	**HAL_I2S_TxCpltCallback** (I2S_HandleTypeDef *hi2s) Tx Transfer completed callbacks.
void	**HAL_I2S_TxHalfCpltCallback** (I2S_HandleTypeDef *hi2s) Tx Transfer Half completed callbacks.
void	**HAL_I2S_ErrorCallback** (I2S_HandleTypeDef *hi2s) I2S error callbacks.
__weak void	**BSP_AUDIO_OUT_TransferComplete_CallBack** (void) Manages the DMA full Transfer complete event.
__weak void	**BSP_AUDIO_OUT_HalfTransfer_CallBack** (void) Manages the DMA Half Transfer complete event.
__weak void	**BSP_AUDIO_OUT_Error_CallBack** (void) Manages the DMA FIFO error event.

Variables

const uint16_t	**audio_sample_SendDummyData** []
static __IO uint32_t	**SendDummyData** = DISABLE
static AUDIO_DrvTypeDef *	**pAudioDrv**
I2S_HandleTypeDef	**hAudioOutI2s**

Detailed Description

This file provides the Audio driver for the STM3210E-Eval board.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

```
  ==========================================
==================================
                     ##### How to use this dri
ver #####
  ==========================================
==================================
   [..]
  (#) This driver supports STM32F103xE devices
 on STM3210E-Eval Kit:
       (++) to play an audio file (all functio
ns names start by BSP_AUDIO_OUT_xxx)

  [..]
   (#) PLAY A FILE:
       (++) Call the function BSP_AUDIO_OUT_In
it(
              OutputDevice: physical output mo
de (OUTPUT_DEVICE_SPEAKER,
                            OUTPUT_DEVICE_HEADP
HONE, OUTPUT_DEVICE_BOTH)
              Volume: initial volume to be set
 (0 is min (mute), 100 is max (100%)
              AudioFreq: Audio frequency in Hz
 (8000, 16000, 22500, 32000 ...)
```

```
              this parameter is relative to th
e audio file/stream type.
               )
           This function configures all the ha
rdware required for the audio application
           (codec, I2C, I2S, GPIOs, DMA and in
terrupt if needed). This function returns 0
           if configuration is OK.
           If the returned value is different
from 0 or the function is stuck then the
           communication with the codec (try t
o un-plug the power or reset device in this ca
se).
              (+++) OUTPUT_DEVICE_SPEAKER: onl
y speaker will be set as output for the
                    audio stream.
              (+++) OUTPUT_DEVICE_HEADPHONE: o
nly headphones will be set as output for
                    the audio stream.
              (+++) OUTPUT_DEVICE_BOTH: both S
peaker and Headphone are used as outputs
                    for the audio stream at th
e same time.
       (++) Call the function BSP_AUDIO_OUT_Pl
ay(
                pBuffer: pointer to the audio
data file address
                Size: size of the buffer to be
 sent in Bytes
               )
              to start playing (for the first
time) from the audio file/stream.
       (++) Call the function BSP_AUDIO_OUT_Pa
use() to pause playing
       (++) Call the function BSP_AUDIO_OUT_Re
sume() to resume playing.
            Note. After calling BSP_AUDIO_OUT_
```

```
Pause() function for pause,
            only BSP_AUDIO_OUT_Resume() should
 be called for resume
            (it is not allowed to call BSP_AUD
IO_OUT_Play() in this case).
            Note. This function should be call
ed only when the audio file is played
            or paused (not stopped).
       (++) For each mode, you may need to imp
lement the relative callback functions
            into your code.
            The Callback functions are named B
SP_AUDIO_OUT_XXXCallBack() and only
            their prototypes are declared in t
he stm3210e_eval_audio.h file.
            (refer to the example for more det
ails on the callbacks implementations)
       (++) To Stop playing, to modify the vol
ume level, the frequency or to mute,
            use the functions BSP_AUDIO_OUT_St
op(), BSP_AUDIO_OUT_SetVolume(),
            AUDIO_OUT_SetFrequency() BSP_AUDIO
_OUT_SetOutputMode and BSP_AUDIO_OUT_SetMute()
.
       (++) The driver API and the callback fu
nctions are at the end of the
            stm3210e_eval_audio.h file.

       (++) This driver provide the High Audio
 Layer: consists of the function API
            exported in the stm3210e_eval_audi
o.h file (BSP_AUDIO_OUT_Init(),
            BSP_AUDIO_OUT_Play() ...)
       (++) This driver provide also the Media
 Access Layer (MAL): which consists
            of functions allowing to access th
e media containing/providing the
```

```
            audio file/stream. These functions
 are also included as local functions into
            the stm3210e_eval_audio.c file (I2
SOUT_Init()...)

  [..]
                      ##### Known Limitations #
####
  ==========================================
==================================
   (#) When using the Speaker, if the audio fi
le quality is not high enough, the
       speaker output may produce high and unc
omfortable noise level. To avoid
       this issue, to use speaker output prope
rly, try to increase audio file
       sampling rate (typically higher than 48
KHz).
       This operation will lead to larger file
 size.

   (#) Communication with the audio codec (thr
ough I2C) may be corrupted if it
       is interrupted by some user interrupt r
outines (in this case, interrupts
       could be disabled just before the start
 of communication then re-enabled
       when it is over). Note that this commun
ication is only done at the
       configuration phase (BSP_AUDIO_OUT_Init
() or BSP_AUDIO_OUT_Stop())
       and when Volume control modification is
 performed (BSP_AUDIO_OUT_SetVolume()
       or BSP_AUDIO_OUT_SetMute()or BSP_AUDIO_
OUT_SetOutputMode()).
       When the audio data is played, no commu
nication is required with the audio codec.
```

```
   (#) Parsing of audio file is not implemente
d (in order to determine audio file
       properties: Mono/Stereo, Data size, Fil
e size, Audio Frequency, Audio Data
       header size ...). The configuration is
fixed for the given audio file.

   (#) Mono audio streaming is not supported (
in order to play mono audio streams,
       each data should be sent twice on the I
2S or should be duplicated on the
       source buffer. Or convert the stream in
 stereo before playing).

   (#) Supports only 16-bit audio data size.
```

Attention:

© COPYRIGHT(c) 2015 STMicroelectronics

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

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

Definition in file **stm3210e_eval_audio.c**.

STM3210E_EVAL BSP User Manual

stm3210e_eval_audio.h File Reference

This file contains the common defines and functions prototypes for **stm3210e_eval_audio.c** driver. More...

```
#include "../Components/ak4343/ak4343.h" #include
"stm3210e_eval.h"
```

Go to the source code of this file.

Defines

#define	**I2SOUT** SPI2
#define	**I2SOUT_CLK_ENABLE**() __HAL_RCC_SPI2_CLK_ENABLE
#define	**I2SOUT_SCK_SD_CLK_ENABLE**() __HAL_RCC_GPIOB_C
#define	**I2SOUT_MCK_CLK_ENABLE**() __HAL_RCC_GPIOC_CLK_
#define	**I2SOUT_WS_CLK_ENABLE**() __HAL_RCC_GPIOB_CLK_E
#define	**I2SOUT_WS_PIN** GPIO_PIN_12 /* PB.12*/
#define	**I2SOUT_SCK_PIN** GPIO_PIN_13 /* PB.13*/
#define	**I2SOUT_SD_PIN** GPIO_PIN_15 /* PB.15*/
#define	**I2SOUT_MCK_PIN** GPIO_PIN_6 /* PC.06*/
#define	**I2SOUT_SCK_SD_GPIO_PORT** GPIOB
#define	**I2SOUT_WS_GPIO_PORT** GPIOB
#define	**I2SOUT_MCK_GPIO_PORT** GPIOC
#define	**I2SOUT_DMAx_CLK_ENABLE**() __HAL_RCC_DMA1_CLK_
#define	**I2SOUT_DMAx_CHANNEL** DMA1_Channel5
#define	**I2SOUT_DMAx_IRQ** DMA1_Channel5_IRQn
#define	**I2SOUT_DMAx_PERIPH_DATA_SIZE** DMA_PDATAALIGN_
#define	**I2SOUT_DMAx_MEM_DATA_SIZE** DMA_MDATAALIGN_HA
#define	**DMA_MAX_SZE** 0xFFFF
#define	**I2SOUT_IRQHandler** DMA1_Channel5_IRQHandler
#define	**AUDIO_OUT_IRQ_PREPRIO** 5 /* Select the preemption prio the highest) */
#define	**AUDIO_OK** 0
#define	**AUDIO_ERROR** 1
#define	**AUDIO_TIMEOUT** 2
#define	**DMA_MAX**(_X_) (((_X_) <= **DMA_MAX_SZE**)? (_X_):**DMA_**

Functions

uint8_t	**BSP_AUDIO_OUT_Init** (uint16_t OutputDevice, uint8_t Volume, uint32_t AudioFreq) Configure the audio peripherals.
uint8_t	**BSP_AUDIO_OUT_Play** (uint16_t *pBuffer, uint32_t Size) Starts playing audio stream from a data buffer for a determined size.
void	**BSP_AUDIO_OUT_ChangeBuffer** (uint16_t *pData, uint16_t Size) Sends n-Bytes on the I2S interface.
uint8_t	**BSP_AUDIO_OUT_Pause** (void) This function Pauses the audio file stream.
uint8_t	**BSP_AUDIO_OUT_Resume** (void) This function Resumes the audio file stream.
uint8_t	**BSP_AUDIO_OUT_Stop** (uint32_t Option) Stops audio playing and Power down the Audio Codec.
uint8_t	**BSP_AUDIO_OUT_SetVolume** (uint8_t Volume) Controls the current audio volume level.
void	**BSP_AUDIO_OUT_SetFrequency** (uint32_t AudioFreq) Update the audio frequency.
uint8_t	**BSP_AUDIO_OUT_SetMute** (uint32_t Cmd) Enables or disables the MUTE mode by software.
uint8_t	**BSP_AUDIO_OUT_SetOutputMode** (uint8_t Output) Switch dynamically (while audio file is played) the output target (speaker or headphone).
__weak void	**BSP_AUDIO_OUT_TransferComplete_CallBack** (void) Manages the DMA full Transfer complete event.
__weak void	**BSP_AUDIO_OUT_HalfTransfer_CallBack** (void) Manages the DMA Half Transfer complete event.

__weak void **BSP_AUDIO_OUT_Error_CallBack** (void)
Manages the DMA FIFO error event.

Detailed Description

This file contains the common defines and functions prototypes for **stm3210e_eval_audio.c** driver.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

© COPYRIGHT(c) 2015 STMicroelectronics

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

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

Definition in file **stm3210e_eval_audio.h**.

STM3210E_EVAL BSP User Manual

Defines | Functions | Variables

stm3210e_eval_lcd.c File Reference

This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM3210E-EVAL evaluation board. More...

```
#include "stm3210e_eval_lcd.h" #include
"../../../Utilities/Fonts/fonts.h"
#include "../../../Utilities/Fonts/font24.c"
#include "../../../Utilities/Fonts/font20.c"
#include "../../../Utilities/Fonts/font16.c"
#include "../../../Utilities/Fonts/font12.c"
#include "../../../Utilities/Fonts/font8.c"
```

Go to the source code of this file.

Defines

```
#define POLY_X(Z)   ((int32_t)((pPoints + (Z))->X))
#define POLY_Y(Z)   ((int32_t)((pPoints + (Z))->Y))
#define MAX_HEIGHT_FONT   17
#define MAX_WIDTH_FONT   24
#define OFFSET_BITMAP   54
#define ABS(X)   ((X) > 0 ? (X) : -(X))
```

Functions

static void	**LCD_DrawPixel** (uint16_t Xpos, uint16_t Ypos, uint16_t RGBCode) Draws a pixel on LCD.
static void	**LCD_DrawChar** (uint16_t Xpos, uint16_t Ypos, const uint8_t *pChar) Draws a character on LCD.
static void	**LCD_SetDisplayWindow** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Sets display window.
uint8_t	**BSP_LCD_Init** (void) Initializes the LCD.
uint32_t	**BSP_LCD_GetXSize** (void) Gets the LCD X size.
uint32_t	**BSP_LCD_GetYSize** (void) Gets the LCD Y size.
uint16_t	**BSP_LCD_GetTextColor** (void) Gets the LCD text color.
uint16_t	**BSP_LCD_GetBackColor** (void) Gets the LCD background color.
void	**BSP_LCD_SetTextColor** (uint16_t Color) Sets the LCD text color.
void	**BSP_LCD_SetBackColor** (uint16_t Color) Sets the LCD background color.
void	**BSP_LCD_SetFont** (sFONT *pFonts) Sets the LCD text font.
sFONT *	**BSP_LCD_GetFont** (void) Gets the LCD text font.
void	**BSP_LCD_Clear** (uint16_t Color) Clears the hole LCD.
void	**BSP_LCD_ClearStringLine** (uint16_t Line) Clears the selected line.

void	**BSP_LCD_DisplayChar** (uint16_t Xpos, uint16_t Ypos, uint8_t Ascii) Displays one character.
void	**BSP_LCD_DisplayStringAt** (uint16_t Xpos, uint16_t Ypos, uint8_t *pText, **Line_ModeTypdef** Mode) Displays characters on the LCD.
void	**BSP_LCD_DisplayStringAtLine** (uint16_t Line, uint8_t *pText) Displays a character on the LCD.
uint16_t	**BSP_LCD_ReadPixel** (uint16_t Xpos, uint16_t Ypos) Reads an LCD pixel.
void	**BSP_LCD_DrawHLine** (uint16_t Xpos, uint16_t Ypos, uint16_t Length) Draws an horizontal line.
void	**BSP_LCD_DrawVLine** (uint16_t Xpos, uint16_t Ypos, uint16_t Length) Draws a vertical line.
void	**BSP_LCD_DrawLine** (uint16_t X1, uint16_t Y1, uint16_t X2, uint16_t Y2) Draws an uni-line (between two points).
void	**BSP_LCD_DrawRect** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Draws a rectangle.
void	**BSP_LCD_DrawCircle** (uint16_t Xpos, uint16_t Ypos, uint16_t Radius) Draws a circle.
void	**BSP_LCD_DrawPolygon** (**pPoint** pPoints, uint16_t PointCount) Draws an poly-line (between many points).
void	**BSP_LCD_DrawEllipse** (int Xpos, int Ypos, int XRadius, int YRadius) Draws an ellipse on LCD.
void	**BSP_LCD_DrawBitmap** (uint16_t Xpos, uint16_t Ypos, uint8_t *pBmp)

	Draws a bitmap picture loaded in the internal Flash (32 bpp).
void	**BSP_LCD_FillRect** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Draws a full rectangle.
void	**BSP_LCD_FillCircle** (uint16_t Xpos, uint16_t Ypos, uint16_t Radius) Draws a full circle.
void	**BSP_LCD_FillTriangle** (uint16_t X1, uint16_t X2, uint16_t X3, uint16_t Y1, uint16_t Y2, uint16_t Y3) Fill triangle.
void	**BSP_LCD_FillPolygon** (**pPoint** pPoints, uint16_t PointCount) Displays a full poly-line (between many points).
void	**BSP_LCD_FillEllipse** (int Xpos, int Ypos, int XRadius, int YRadius) Draws a full ellipse.
void	**BSP_LCD_DisplayOn** (void) Enables the display.
void	**BSP_LCD_DisplayOff** (void) Disables the display.

Variables

LCD_DrawPropTypeDef	**DrawProp**
static LCD_DrvTypeDef *	**lcd_drv**
static uint8_t	**bitmap** [**MAX_HEIGHT_FONT** ***MAX_WIDTH_FONT** *2+**OFFSET_BITMAP**] = {0}
static uint32_t	**LCD_SwapXY** = 0

Detailed Description

This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM3210E-EVAL evaluation board.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

```
  ==========================================
==================================
                     ##### How to use this dri
ver #####
  ==========================================
==================================
  [..]
   (#) This driver is used to drive indirectly
 an LCD TFT.

   (#) This driver supports the AM-240320L8TNQ
W00H (ILI9320), AM-240320LDTNQW-05H (ILI9325)
       AM-240320LDTNQW-02H (SPFD5408B) and AM2
40320LGTNQW-01H (HX8347D) LCD
       mounted on MB895 daughter board

   (#) The ILI9320, ILI9325, SPFD5408B and HX8
347D components driver MUST be included with t
his driver.

   (#) Initialization steps:
       (++) Initialize the LCD using the BSP_L
CD_Init() function.
```

```
   (#) Display on LCD
       (++) Clear the hole LCD using yhe BSP_L
CD_Clear() function or only one specified
            string line using the BSP_LCD_Clea
rStringLine() function.
       (++) Display a character on the specifi
ed line and column using the BSP_LCD_DisplayCh
ar()
            function or a complete string line
 using the BSP_LCD_DisplayStringAtLine() funct
ion.
       (++) Display a string line on the speci
fied position (x,y in pixel) and align mode
            using the BSP_LCD_DisplayStringAtL
ine() function.
       (++) Draw and fill a basic shapes (dot,
 line, rectangle, circle, ellipse, .. bitmap,
raw picture)
            on LCD using a set of functions.
```

Attention:

© COPYRIGHT(c) 2015 STMicroelectronics

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

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

Definition in file **stm3210e_eval_lcd.c**.

STM3210E_EVAL BSP User Manual

Data Structures | Defines | Typedefs | Enumerations | Functions

stm3210e_eval_lcd.h File Reference

This file contains the common defines and functions prototypes for the **stm3210e_eval_lcd.c** driver. More...

```
#include "stm3210e_eval.h" #include
"../Components/hx8347d/hx8347d.h"
#include "../Components/spfd5408/spfd5408.h"
#include "../Components/ili9320/ili9320.h"
#include "../../../Utilities/Fonts/fonts.h"
```

Go to the source code of this file.

Data Structures

struct	**LCD_DrawPropTypeDef**
struct	**Point**

Defines

#define	**LCD_OK** 0x00 LCD status structure definition.
#define	**LCD_ERROR** 0x01
#define	**LCD_TIMEOUT** 0x02
#define	**LCD_COLOR_BLUE** (uint16_t)0x001F LCD color.
#define	**LCD_COLOR_GREEN** (uint16_t)0x07E0
#define	**LCD_COLOR_RED** (uint16_t)0xF800
#define	**LCD_COLOR_CYAN** (uint16_t)0x07FF
#define	**LCD_COLOR_MAGENTA** (uint16_t)0xF81F
#define	**LCD_COLOR_YELLOW** (uint16_t)0xFFE0
#define	**LCD_COLOR_LIGHTBLUE** (uint16_t)0x841F
#define	**LCD_COLOR_LIGHTGREEN** (uint16_t)0x87F0
#define	**LCD_COLOR_LIGHTRED** (uint16_t)0xFC10
#define	**LCD_COLOR_LIGHTCYAN** (uint16_t)0x87FF
#define	**LCD_COLOR_LIGHTMAGENTA** (uint16_t)0xFC1F
#define	**LCD_COLOR_LIGHTYELLOW** (uint16_t)0xFFF0
#define	**LCD_COLOR_DARKBLUE** (uint16_t)0x0010
#define	**LCD_COLOR_DARKGREEN** (uint16_t)0x0400
#define	**LCD_COLOR_DARKRED** (uint16_t)0x8000
#define	**LCD_COLOR_DARKCYAN** (uint16_t)0x0410
#define	**LCD_COLOR_DARKMAGENTA** (uint16_t)0x8010
#define	**LCD_COLOR_DARKYELLOW** (uint16_t)0x8400
#define	**LCD_COLOR_WHITE** (uint16_t)0xFFFF
#define	**LCD_COLOR_LIGHTGRAY** (uint16_t)0xD69A
#define	**LCD_COLOR_GRAY** (uint16_t)0x8410
#define	**LCD_COLOR_DARKGRAY** (uint16_t)0x4208
#define	**LCD_COLOR_BLACK** (uint16_t)0x0000
#define	**LCD_COLOR_BROWN** (uint16_t)0xA145
#define	**LCD_COLOR_ORANGE** (uint16_t)0xFD20
#define	**LCD_DEFAULT_FONT** Font24

LCD default font.

Typedefs

typedef struct **Point** * **pPoint**

Enumerations

enum	**Line_ModeTypdef** { **CENTER_MODE** = 0x01, **RIGHT_MODE** = 0x02, **LEFT_MODE** = 0x03 } Line mode structures definition. More...

Functions

uint8_t	**BSP_LCD_Init** (void) Initializes the LCD.
uint32_t	**BSP_LCD_GetXSize** (void) Gets the LCD X size.
uint32_t	**BSP_LCD_GetYSize** (void) Gets the LCD Y size.
uint16_t	**BSP_LCD_GetTextColor** (void) Gets the LCD text color.
uint16_t	**BSP_LCD_GetBackColor** (void) Gets the LCD background color.
void	**BSP_LCD_SetTextColor** (__IO uint16_t Color)
void	**BSP_LCD_SetBackColor** (__IO uint16_t Color)
void	**BSP_LCD_SetFont** (sFONT *pFonts) Sets the LCD text font.
sFONT *	**BSP_LCD_GetFont** (void) Gets the LCD text font.
void	**BSP_LCD_Clear** (uint16_t Color) Clears the hole LCD.
void	**BSP_LCD_ClearStringLine** (uint16_t Line) Clears the selected line.
void	**BSP_LCD_DisplayStringAtLine** (uint16_t Line, uint8_t *pText) Displays a character on the LCD.
void	**BSP_LCD_DisplayStringAt** (uint16_t Xpos, uint16_t Ypos, uint8_t *pText, **Line_ModeTypdef** Mode) Displays characters on the LCD.
void	**BSP_LCD_DisplayChar** (uint16_t Xpos, uint16_t Ypos, uint8_t Ascii) Displays one character.
uint16_t	**BSP_LCD_ReadPixel** (uint16_t Xpos, uint16_t Ypos) Reads an LCD pixel.

void	**BSP_LCD_DrawPixel** (uint16_t Xpos, uint16_t Ypos, uint16_t RGB_Code)
void	**BSP_LCD_DrawHLine** (uint16_t Xpos, uint16_t Ypos, uint16_t Length) Draws an horizontal line.
void	**BSP_LCD_DrawVLine** (uint16_t Xpos, uint16_t Ypos, uint16_t Length) Draws a vertical line.
void	**BSP_LCD_DrawLine** (uint16_t X1, uint16_t Y1, uint16_t X2, uint16_t Y2) Draws an uni-line (between two points).
void	**BSP_LCD_DrawRect** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Draws a rectangle.
void	**BSP_LCD_DrawCircle** (uint16_t Xpos, uint16_t Ypos, uint16_t Radius) Draws a circle.
void	**BSP_LCD_DrawPolygon** (**pPoint** pPoints, uint16_t PointCount) Draws an poly-line (between many points).
void	**BSP_LCD_FillTriangle** (uint16_t X1, uint16_t X2, uint16_t X3, uint16_t Y1, uint16_t Y2, uint16_t Y3) Fill triangle.
void	**BSP_LCD_FillPolygon** (**pPoint** pPoints, uint16_t PointCount) Displays a full poly-line (between many points).
void	**BSP_LCD_DrawEllipse** (int Xpos, int Ypos, int XRadius, int YRadius) Draws an ellipse on LCD.
void	**BSP_LCD_DrawBitmap** (uint16_t Xpos, uint16_t Ypos, uint8_t *pBmp) Draws a bitmap picture loaded in the internal Flash (32 bpp).
	BSP_LCD_DrawRGBImage (uint16_t Xpos, uint16_t Ypos,

void	uint16_t Xsize, uint16_t Ysize, uint8_t *pbmp)
void	**BSP_LCD_FillRect** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Draws a full rectangle.
void	**BSP_LCD_FillCircle** (uint16_t Xpos, uint16_t Ypos, uint16_t Radius) Draws a full circle.
void	**BSP_LCD_FillEllipse** (int Xpos, int Ypos, int XRadius, int YRadius) Draws a full ellipse.
void	**BSP_LCD_DisplayOff** (void) Disables the display.
void	**BSP_LCD_DisplayOn** (void) Enables the display.

Detailed Description

This file contains the common defines and functions prototypes for the **stm3210e_eval_lcd.c** driver.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

© COPYRIGHT(c) 2015 STMicroelectronics

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

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

Definition in file **stm3210e_eval_lcd.h**.

STM3210E_EVAL BSP User Manual

Functions | Variables

stm3210e_eval_nand.c File Reference

This file includes a standard driver for the NAND512W3A2CN6E NAND flash memory device mounted on STM3210E-EVAL evaluation board. More...

```
#include "stm3210e_eval_nand.h"
```

Go to the source code of this file.

Functions

static void	**NAND_MspInit** (void) Initializes the NAND MSP.
uint8_t	**BSP_NAND_Init** (void) Initializes the NAND device.
uint8_t	**BSP_NAND_ReadData** (NAND_AddressTypeDef BlockAddress, uint8_t *pData, uint32_t uwNumPage) Reads an amount of data from the NAND device.
uint8_t	**BSP_NAND_WriteData** (NAND_AddressTypeDef BlockAddress, uint8_t *pData, uint32_t uwNumPage) Writes an amount of data to the NAND device.
uint8_t	**BSP_NAND_Erase_Block** (NAND_AddressTypeDef BlockAddress) Erases the specified block of the NAND device.
uint8_t	**BSP_NAND_Read_ID** (NAND_IDTypeDef *pNAND_ID) Reads NAND flash IDs.

Variables

static NAND_HandleTypeDef	**nandHandle**
static FSMC_NAND_PCC_TimingTypeDef	**Timing**

Detailed Description

This file includes a standard driver for the NAND512W3A2CN6E NAND flash memory device mounted on STM3210E-EVAL evaluation board.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

© COPYRIGHT(c) 2015 STMicroelectronics

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

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

Definition in file **stm3210e_eval_nand.c**.

STM3210E_EVAL BSP User Manual

stm3210e_eval_nand.h File Reference

This file contains the common defines and functions prototypes for the **stm3210e_eval_nand.c** driver. More...

```
#include "stm32f1xx_hal.h"
```

Go to the source code of this file.

Defines

#define	**NAND_OK** 0x00 NAND status structure definition.
#define	**NAND_DEVICE_ADDR** ((uint32_t)NAND_DEVICE1)
#define	**NAND_PAGE_SIZE** ((uint16_t)0x0200) /* 512 bytes per page w/o Spare Area */ FSMC NAND memory parameters.
#define	**NAND_BLOCK_SIZE** ((uint16_t)0x0020) /* 32x512 bytes pages per block */
#define	**NAND_ZONE_SIZE** ((uint16_t)0x0400) /* 1024 Block per zone */
#define	**NAND_SPARE_AREA_SIZE** ((uint16_t)0x0010) /* last 16 bytes as spare area */
#define	**NAND_MAX_ZONE** ((uint16_t)0x0004) /* 4 zones of 1024 block */

Functions

uint8_t	**BSP_NAND_Init** (void) Initializes the NAND device.
uint8_t	**BSP_NAND_ReadData** (NAND_AddressTypeDef BlockAddress, uint8_t *pData, uint32_t uwNumPage) Reads an amount of data from the NAND device.
uint8_t	**BSP_NAND_WriteData** (NAND_AddressTypeDef BlockAddress, uint8_t *pData, uint32_t uwNumPage) Writes an amount of data to the NAND device.
uint8_t	**BSP_NAND_Erase_Block** (NAND_AddressTypeDef BlockAddress) Erases the specified block of the NAND device.
uint8_t	**BSP_NAND_Erase_Chip** (void)
uint8_t	**BSP_NAND_Read_ID** (NAND_IDTypeDef *pNAND_ID) Reads NAND flash IDs.

Detailed Description

This file contains the common defines and functions prototypes for the **stm3210e_eval_nand.c** driver.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

© COPYRIGHT(c) 2015 STMicroelectronics

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

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

Definition in file **stm3210e_eval_nand.h**.

STM3210E_EVAL BSP User Manual

stm3210e_eval_nor.c File Reference

This file includes a standard driver for the M29W128GL70ZA6E NOR flash memory device mounted on STM3210E-EVAL evaluation board. More...

```
#include "stm3210e_eval_nor.h"
```

Go to the source code of this file.

Functions

static void	**NOR_MspInit** (void) Initializes the NOR MSP.
uint8_t	**BSP_NOR_Init** (void) Initializes the NOR device.
uint8_t	**BSP_NOR_ReadData** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Reads an amount of data from the NOR device.
void	**BSP_NOR_ReturnToReadMode** (void) Returns the NOR memory to read mode.
uint8_t	**BSP_NOR_WriteData** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Writes an amount of data to the NOR device.
uint8_t	**BSP_NOR_ProgramData** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Programs an amount of data to the NOR device.
uint8_t	**BSP_NOR_Erase_Block** (uint32_t BlockAddress) Erases the specified block of the NOR device.
uint8_t	**BSP_NOR_Erase_Chip** (void) Erases the entire NOR chip.
uint8_t	**BSP_NOR_Read_ID** (NOR_IDTypeDef *pNOR_ID) Reads NOR flash IDs.
void	**HAL_NOR_MspWait** (NOR_HandleTypeDef *hnor, uint32_t Timeout) NOR BSP Wait for Ready/Busy signal.

Variables

static NOR_HandleTypeDef	**norHandle**
static FSMC_NORSRAM_TimingTypeDef	**Timing**

Detailed Description

This file includes a standard driver for the M29W128GL70ZA6E NOR flash memory device mounted on STM3210E-EVAL evaluation board.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

© COPYRIGHT(c) 2015 STMicroelectronics

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

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

Definition in file **stm3210e_eval_nor.c**.

STM3210E_EVAL BSP User Manual

stm3210e_eval_nor.h File Reference

This file contains the common defines and functions prototypes for the **stm3210e_eval_nor.c** driver. More...

```
#include "stm32f1xx_hal.h"
```

Go to the source code of this file.

Defines

#define	**NOR_STATUS_OK** 0x00 NOR status structure definition.
#define	**NOR_STATUS_ERROR** 0x01
#define	**NOR_DEVICE_ADDR** ((uint32_t)0x64000000)
#define	**NOR_MEMORY_WIDTH** FSMC_NORSRAM_MEM_BUS_WI
#define	**NOR_BURSTACCESS** FSMC_BURST_ACCESS_MODE_DI
#define	**NOR_WRITEBURST** FSMC_WRITE_BURST_DISABLE
#define	**BLOCKERASE_TIMEOUT** ((uint32_t)0x00A00000) /* NOR b erase timeout */
#define	**CHIPERASE_TIMEOUT** ((uint32_t)0x30000000) /* NOR chip timeout */
#define	**PROGRAM_TIMEOUT** ((uint32_t)0x00004400) /* NOR progra timeout */
#define	**NOR_READY_BUSY_PIN** GPIO_PIN_6
#define	**NOR_READY_BUSY_GPIO** GPIOD
#define	**NOR_READY_STATE** GPIO_PIN_SET
#define	**NOR_BUSY_STATE** GPIO_PIN_RESET

Functions

uint8_t	**BSP_NOR_Init** (void) Initializes the NOR device.
uint8_t	**BSP_NOR_ReadData** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Reads an amount of data from the NOR device.
uint8_t	**BSP_NOR_WriteData** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Writes an amount of data to the NOR device.
uint8_t	**BSP_NOR_ProgramData** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Programs an amount of data to the NOR device.
uint8_t	**BSP_NOR_Erase_Block** (uint32_t BlockAddress) Erases the specified block of the NOR device.
uint8_t	**BSP_NOR_Erase_Chip** (void) Erases the entire NOR chip.
uint8_t	**BSP_NOR_Read_ID** (NOR_IDTypeDef *pNOR_ID) Reads NOR flash IDs.
void	**BSP_NOR_ReturnToReadMode** (void) Returns the NOR memory to read mode.

Detailed Description

This file contains the common defines and functions prototypes for the **stm3210e_eval_nor.c** driver.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

© COPYRIGHT(c) 2015 STMicroelectronics

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

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

Definition in file **stm3210e_eval_nor.h**.

STM3210E_EVAL BSP User Manual

stm3210e_eval_sd.c File Reference

This file includes the uSD card driver. More...

```
#include "stm3210e_eval_sd.h"
```

Go to the source code of this file.

Functions

static void	**SD_MspInit** (void) Initializes the SD MSP.
HAL_SD_ErrorTypedef	**SD_DMAConfigRx** (SD_HandleTypeDef *hsd) SD_DMAConfigRx.
HAL_SD_ErrorTypedef	**SD_DMAConfigTx** (SD_HandleTypeDef *hsd) SD_DMAConfigTx.
uint8_t	**BSP_SD_Init** (void) Initializes the SD card device.
uint8_t	**BSP_SD_ITConfig** (void) Configures Interrupt mode for SD detection pin.
uint8_t	**BSP_SD_IsDetected** (void) Detects if SD card is correctly plugged in the memory slot or not.
void	**BSP_SD_DetectIT** (void) SD detect IT treatment.
__weak void	**BSP_SD_DetectCallback** (void) SD detect IT detection callback.
uint8_t	**BSP_SD_ReadBlocks** (uint32_t *pData, uint64_t ReadAddr, uint32_t BlockSize, uint32_t NumOfBlocks) Reads block(s) from a specified address in an SD card, in polling mode.
uint8_t	**BSP_SD_WriteBlocks** (uint32_t *pData, uint64_t WriteAddr, uint32_t BlockSize, uint32_t NumOfBlocks) Writes block(s) to a specified address in an SD card, in polling mode.

uint8_t	**BSP_SD_ReadBlocks_DMA** (uint32_t *pData, uint64_t ReadAddr, uint32_t BlockSize, uint32_t NumOfBlocks) Reads block(s) from a specified address in an SD card, in DMA mode.
uint8_t	**BSP_SD_WriteBlocks_DMA** (uint32_t *pData, uint64_t WriteAddr, uint32_t BlockSize, uint32_t NumOfBlocks) Writes block(s) to a specified address in an SD card, in DMA mode.
uint8_t	**BSP_SD_Erase** (uint64_t StartAddr, uint64_t EndAddr) Erases the specified memory area of the given SD card.
void	**BSP_SD_IRQHandler** (void) Handles SD card interrupt request.
void	**BSP_SD_DMA_Tx_IRQHandler** (void) Handles SD DMA Tx transfer interrupt request.
void	**BSP_SD_DMA_Rx_IRQHandler** (void) Handles SD DMA Rx transfer interrupt request.
HAL_SD_TransferStateTypedef	**BSP_SD_GetStatus** (void) Gets the current SD card data status.
void	**BSP_SD_GetCardInfo** (HAL_SD_CardInfoTypedef *CardInfo) Get SD information about specific

SD card.

Variables

SD_HandleTypeDef	**uSdHandle**
static **SD_CardInfo**	**uSdCardInfo**

Detailed Description

This file includes the uSD card driver.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

```
  ==========================================
==================================
                     ##### How to use this dri
ver #####
  ==========================================
==================================
  (#) This driver is used to drive the micro S
D external card mounted on STM3210E-EVAL
     evaluation board.

  (#) This driver does not need a specific com
ponent driver for the micro SD device
     to be included with.

  (#) Initialization steps:
       (++) Initialize the micro SD card using
 the BSP_SD_Init() function. This
            function includes the MSP layer ha
rdware resources initialization and the
            SDIO interface configuration to in
terface with the external micro SD. It
            also includes the micro SD initial
ization sequence.
       (++) To check the SD card presence you
```

```
can use the function BSP_SD_IsDetected() which

            returns the detection status
       (++) If SD presence detection interrupt
 mode is desired, you must configure the
            SD detection interrupt mode by cal
ling the function BSP_SD_ITConfig(). The inter
rupt
            is generated as an external interr
upt whenever the micro SD card is
            plugged/unplugged in/from the eval
uation board. The SD detection interrupt
            is handeled by calling the functio
n BSP_SD_DetectIT() which is called in the IRQ
            handler file, the user callback is
 implemented in the function BSP_SD_DetectCall
back().
       (++) The function BSP_SD_GetCardInfo()
is used to get the micro SD card information
            which is stored in the structure "
HAL_SD_CardInfoTypedef".

  (#) Micro SD card operations
       (++) The micro SD card can be accessed
with read/write block(s) operations once
            it is reay for access. The access
cand be performed whether using the polling
            mode by calling the functions BSP_
SD_ReadBlocks()/BSP_SD_WriteBlocks(), or by DM
A
            transfer using the functions BSP_S
D_ReadBlocks_DMA()/BSP_SD_WriteBlocks_DMA()
       (++) The DMA transfer complete is used
with interrupt mode. Once the SD transfer
            is complete, the SD interrupt is h
andeled using the function BSP_SD_IRQHandler()
,
```

```
              the DMA Tx/Rx transfer complete ar
e handeled using the functions
              BSP_SD_DMA_Tx_IRQHandler()/BSP_SD_
DMA_Rx_IRQHandler(). The corresponding user ca
llbacks
              are implemented by the user at app
lication level.
       (++) The SD erase block(s) is performed
 using the function BSP_SD_Erase() with specif
ying
              the number of blocks to erase.
       (++) The SD runtime status is returned
when calling the function BSP_SD_GetStatus().
   [..]
```

Attention:

© COPYRIGHT(c) 2015 STMicroelectronics

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

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

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

© COPYRIGHT(c) 2015 STMicroelectronics

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

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

Definition in file **stm3210e_eval_sd.c**.

STM3210E_EVAL BSP User Manual

Defines | Functions

stm3210e_eval_sd.h File Reference

```
#include "stm32f1xx_hal.h"
```

Go to the source code of this file.

Defines

#define	**SD_CardInfo** HAL_SD_CardInfoTypedef SD Card information structure.
#define	**MSD_OK** 0x00 SD status structure definition.
#define	**MSD_ERROR** 0x01
#define	**SD_DETECT_PIN** GPIO_PIN_11
#define	**SD_DETECT_GPIO_PORT** GPIOF
#define	**__SD_DETECT_GPIO_CLK_ENABLE**() __HAL_RCC_GPIO
#define	**SD_DETECT_IRQn** EXTI15_10_IRQn
#define	**SD_DATATIMEOUT** ((uint32_t)100000000)
#define	**SD_PRESENT** ((uint8_t)0x01)
#define	**SD_NOT_PRESENT** ((uint8_t)0x00)
#define	**__DMAx_TxRx_CLK_ENABLE** __HAL_RCC_DMA2_CLK_E
#define	**SD_DMAx_Tx_INSTANCE** DMA2_Channel4
#define	**SD_DMAx_Rx_INSTANCE** DMA2_Channel4
#define	**SD_DMAx_Tx_IRQn** DMA2_Channel4_5_IRQn
#define	**SD_DMAx_Rx_IRQn** DMA2_Channel4_5_IRQn
#define	**SD_DMAx_Tx_IRQHandler** DMA2_Channel4_5_IRQHandle
#define	**SD_DMAx_Rx_IRQHandler** DMA2_Channel4_5_IRQHandle

Functions

uint8_t	**BSP_SD_Init** (void) Initializes the SD card device.
uint8_t	**BSP_SD_ITConfig** (void) Configures Interrupt mode for SD detection pin.
void	**BSP_SD_DetectIT** (void) SD detect IT treatment.
void	**BSP_SD_DetectCallback** (void) SD detect IT detection callback.
uint8_t	**BSP_SD_ReadBlocks** (uint32_t *pData, uint64_t ReadAddr, uint32_t BlockSize, uint32_t NumOfBlocks) Reads block(s) from a specified address in an SD card, in polling mode.
uint8_t	**BSP_SD_WriteBlocks** (uint32_t *pData, uint64_t WriteAddr, uint32_t BlockSize, uint32_t NumOfBlocks) Writes block(s) to a specified address in an SD card, in polling mode.
uint8_t	**BSP_SD_ReadBlocks_DMA** (uint32_t *pData, uint64_t ReadAddr, uint32_t BlockSize, uint32_t NumOfBlocks) Reads block(s) from a specified address in an SD card, in DMA mode.
uint8_t	**BSP_SD_WriteBlocks_DMA** (uint32_t *pData, uint64_t WriteAddr, uint32_t BlockSize, uint32_t NumOfBlocks)

	Writes block(s) to a specified address in an SD card, in DMA mode.
uint8_t	**BSP_SD_Erase** (uint64_t StartAddr, uint64_t EndAddr) Erases the specified memory area of the given SD card.
void	**BSP_SD_IRQHandler** (void) Handles SD card interrupt request.
void	**BSP_SD_DMA_Tx_IRQHandler** (void) Handles SD DMA Tx transfer interrupt request.
void	**BSP_SD_DMA_Rx_IRQHandler** (void) Handles SD DMA Rx transfer interrupt request.
HAL_SD_TransferStateTypedef	**BSP_SD_GetStatus** (void) Gets the current SD card data status.
void	**BSP_SD_GetCardInfo** (HAL_SD_CardInfoTypedef *CardInfo) Get SD information about specific SD card.
uint8_t	**BSP_SD_IsDetected** (void) Detects if SD card is correctly plugged in the memory slot or not.

STM3210E_EVAL BSP User Manual

stm3210e_eval_serialflash.c File Reference

This file provides a set of functions needed to manage the SPI M25Pxx FLASH memory mounted on STM3210E_EVAL board. It implements a high level communication layer for read and write from/to this memory. The needed STM32 hardware resources (SPI and GPIO) are defined in **stm3210e_eval.h** file, and the initialization is performed in **FLASH_SPI_IO_Init()** function declared in **stm3210e_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **FLASH_SPI_IO_Init()** function. More...

```
#include "stm3210e_eval_serialflash.h"
```

Go to the source code of this file.

Functions

uint8_t	**BSP_SERIAL_FLASH_Init** (void) Initializes peripherals used by the Serial FLASH device.
uint8_t	**BSP_SERIAL_FLASH_EraseSector** (uint32_t SectorAddr) Erases the specified FLASH sector.
uint8_t	**BSP_SERIAL_FLASH_EraseBulk** (void) Erases the entire FLASH.
uint8_t	**BSP_SERIAL_FLASH_WritePage** (uint32_t uwStartAddress, uint8_t *pData, uint32_t uwDataSize) Writes more than one byte to the FLASH with a single WRITE cycle (Page WRITE sequence).
uint8_t	**BSP_SERIAL_FLASH_WriteData** (uint32_t uwStartAddress, uint8_t *pData, uint32_t uwDataSize) Writes block of data to the FLASH.
uint8_t	**BSP_SERIAL_FLASH_ReadData** (uint32_t uwStartAddress, uint8_t *pData, uint32_t uwDataSize) Reads a block of data from the FLASH.
uint32_t	**BSP_SERIAL_FLASH_ReadID** (void) Reads FLASH identification.

Detailed Description

This file provides a set of functions needed to manage the SPI M25Pxx FLASH memory mounted on STM3210E_EVAL board. It implements a high level communication layer for read and write from/to this memory. The needed STM32 hardware resources (SPI and GPIO) are defined in **stm3210e_eval.h** file, and the initialization is performed in **FLASH_SPI_IO_Init()** function declared in **stm3210e_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **FLASH_SPI_IO_Init()** function.

Author:

MCD Application Team

Version:

V6.0.1

Date:

18-December-2015 +----------+ | Pin assignment | +----------+----------+----------+ | STM32 SPI Pins | BSP_FLASH | Pin | +----------+----------+----------+ | BSP_SERIAL_FLASH_CS_PIN | ChipSelect(/S)| 1 | | BSP_SERIAL_FLASH_SPI_MISO_PIN / MISO | DataOut(Q) | 2 | | | VCC | 3 (3.3 V)| | | GND | 4 (0 V) | | BSP_SERIAL_FLASH_SPI_MOSI_PIN / MOSI | DataIn(D) | 5 | | BSP_SERIAL_FLASH_SPI_SCK_PIN / SCLK | Clock(C) | 6 | | | VCC | 7 (3.3 V)| | | VCC | 8 (3.3 V)| +----------+----------+----------+

Attention:

© COPYRIGHT(c) 2015 STMicroelectronics

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

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

Definition in file **stm3210e_eval_serialflash.c**.

STM3210E_EVAL BSP User Manual

stm3210e_eval_serialflash.h File Reference

This file contains the common defines and functions prototypes for the stm3210e_eval_flash driver. More...

```
#include "stm3210e_eval.h"
```

Go to the source code of this file.

Defines

```
#define FLASH_OK 0x00
#define FLASH_ERROR 0x01
```

Functions

uint8_t	**BSP_SERIAL_FLASH_Init** (void) Initializes peripherals used by the Serial FLASH device.
uint8_t	**BSP_SERIAL_FLASH_EraseSector** (uint32_t SectorAddr) Erases the specified FLASH sector.
uint8_t	**BSP_SERIAL_FLASH_EraseBulk** (void) Erases the entire FLASH.
uint8_t	**BSP_SERIAL_FLASH_WritePage** (uint32_t uwStartAddress, uint8_t *pData, uint32_t uwDataSize) Writes more than one byte to the FLASH with a single WRITE cycle (Page WRITE sequence).
uint8_t	**BSP_SERIAL_FLASH_WriteData** (uint32_t uwStartAddress, uint8_t *pData, uint32_t uwDataSize) Writes block of data to the FLASH.
uint8_t	**BSP_SERIAL_FLASH_ReadData** (uint32_t uwStartAddress, uint8_t *pData, uint32_t uwDataSize) Reads a block of data from the FLASH.
uint32_t	**BSP_SERIAL_FLASH_ReadID** (void) Reads FLASH identification.
HAL_StatusTypeDef	**FLASH_SPI_IO_Init** (void) Initializes the FLASH SPI and put it into StandBy State (Ready for data transfer).
uint8_t	**FLASH_SPI_IO_WriteByte** (uint8_t Data) Write a byte on the FLASH SPI.
uint8_t	**FLASH_SPI_IO_ReadByte** (void) Read a byte from the FLASH SPI.
	FLASH_SPI_IO_ReadData (uint32_t

HAL_StatusTypeDef	MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Read data from FLASH SPI driver.
void	**FLASH_SPI_IO_WriteEnable** (void) Select the FLASH SPI and send "Write Enable" instruction.
HAL_StatusTypeDef	**FLASH_SPI_IO_WaitForWriteEnd** (void) Wait response from the FLASH SPI and Deselect the device.
uint32_t	**FLASH_SPI_IO_ReadID** (void) Reads FLASH SPI identification.

Detailed Description

This file contains the common defines and functions prototypes for the stm3210e_eval_flash driver.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

© COPYRIGHT(c) 2015 STMicroelectronics

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

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

Definition in file **stm3210e_eval_serialflash.h**.

STM3210E_EVAL BSP User Manual

stm3210e_eval_sram.c File Reference

This file includes the SRAM driver for the IS61WV51216BLL-10M memory device mounted on STM3210E-EVAL evaluation board. More...

```
#include "stm3210e_eval_sram.h"
```

Go to the source code of this file.

Functions

static void	**SRAM_MspInit** (void) Initializes SRAM MSP.
uint8_t	**BSP_SRAM_Init** (void) Initializes the SRAM device.
uint8_t	**BSP_SRAM_ReadData** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Reads an amount of data from the SRAM device in polling mode.
uint8_t	**BSP_SRAM_ReadData_DMA** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Reads an amount of data from the SRAM device in DMA mode.
uint8_t	**BSP_SRAM_WriteData** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Writes an amount of data from the SRAM device in polling mode.
uint8_t	**BSP_SRAM_WriteData_DMA** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Writes an amount of data from the SRAM device in DMA mode.
void	**BSP_SRAM_DMA_IRQHandler** (void) Handles SRAM DMA transfer interrupt request.

Variables

SRAM_HandleTypeDef	**sramHandle**
static FSMC_NORSRAM_TimingTypeDef	**Timing**

Detailed Description

This file includes the SRAM driver for the IS61WV51216BLL-10M memory device mounted on STM3210E-EVAL evaluation board.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

© COPYRIGHT(c) 2015 STMicroelectronics

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

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

Definition in file **stm3210e_eval_sram.c**.

STM3210E_EVAL BSP User Manual

stm3210e_eval_sram.h File Reference

This file contains the common defines and functions prototypes for the **stm3210e_eval_sram.c** driver. More...

```
#include "stm32f1xx_hal.h"
```

Go to the source code of this file.

Defines

#define	**__STM3210e_EVAL_SRAM_H**
#define	**SRAM_OK** 0x00 SRAM status structure definition.
#define	**SRAM_ERROR** 0x01
#define	**SRAM_DEVICE_ADDR** ((uint32_t)0x68000000)
#define	**SRAM_DEVICE_SIZE** ((uint32_t)0x200000) /* SRAM device MBytes */
#define	**SRAM_MEMORY_WIDTH** FSMC_NORSRAM_MEM_BUS_W
#define	**SRAM_BURSTACCESS** FSMC_BURST_ACCESS_MODE_D
#define	**SRAM_WRITEBURST** FSMC_WRITE_BURST_DISABLE
#define	**__SRAM_DMAx_CLK_ENABLE** __HAL_RCC_DMA2_CLK_
#define	**SRAM_DMAx_STREAM** DMA2_Channel1
#define	**SRAM_DMAx_IRQn** DMA2_Channel1_IRQn
#define	**SRAM_DMAx_IRQHandler** DMA2_Channel1_IRQHandler

Functions

uint8_t	**BSP_SRAM_Init** (void) Initializes the SRAM device.
uint8_t	**BSP_SRAM_ReadData** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Reads an amount of data from the SRAM device in polling mode.
uint8_t	**BSP_SRAM_ReadData_DMA** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Reads an amount of data from the SRAM device in DMA mode.
uint8_t	**BSP_SRAM_WriteData** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Writes an amount of data from the SRAM device in polling mode.
uint8_t	**BSP_SRAM_WriteData_DMA** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Writes an amount of data from the SRAM device in DMA mode.
void	**BSP_SRAM_DMA_IRQHandler** (void) Handles SRAM DMA transfer interrupt request.

Detailed Description

This file contains the common defines and functions prototypes for the **stm3210e_eval_sram.c** driver.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

© COPYRIGHT(c) 2015 STMicroelectronics

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

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

Definition in file **stm3210e_eval_sram.h**.

Define Documentation

#define __STM3210e_EVAL_SRAM_H

Definition at line **41** of file **stm3210e_eval_sram.h**.

STM3210E_EVAL BSP User Manual

stm3210e_eval_tsensor.c File Reference

This file provides a set of functions needed to manage the I2C STLM75 temperature sensor mounted on STM3210E_EVAL board . It implements a high level communication layer for read and write from/to this sensor. The needed STM32L152xD hardware resources (I2C and GPIO) are defined in **stm3210e_eval.h** file, and the initialization is performed in **TSENSOR_IO_Init()** function declared in **stm3210e_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **TSENSOR_IO_Init()** function. More...

```
#include "stm3210e_eval_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.
uint8_t	**BSP_TSENSOR_ReadStatus** (void) Returns the Temperature Sensor status.
uint16_t	**BSP_TSENSOR_ReadTemp** (void) Read Temperature register of STLM75.

Variables

static TSENSOR_DrvTypeDef *	**tsensor_drv**
__IO uint16_t	**TSENSORAddress** = 0

Detailed Description

This file provides a set of functions needed to manage the I2C STLM75 temperature sensor mounted on STM3210E_EVAL board . It implements a high level communication layer for read and write from/to this sensor. The needed STM32L152xD hardware resources (I2C and GPIO) are defined in **stm3210e_eval.h** file, and the initialization is performed in **TSENSOR_IO_Init()** function declared in **stm3210e_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **TSENSOR_IO_Init()** function.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015 +----------+ | Pin assignment | +----------+----------+----------+ | STM3210E I2C Pins | STLM75 | Pin | +----------+----------+----------+ | EVAL_I2C2_SDA_PIN | SDA | 1 | | EVAL_I2C2_SCL_PIN | SCL | 2 | | EVAL_I2C2_SMBUS_PIN | OS/INT | 3 | | . | GND | 4 (0V) | | . | A2 | 5 | | . | A1 | 6 | | . | A0 | 7 | | . | VDD | 8 (3.3V)| +----------+----------+----------+

Attention:

© COPYRIGHT(c) 2015 STMicroelectronics

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

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

Definition in file **stm3210e_eval_tsensor.c**.

STM3210E_EVAL BSP User Manual

stm3210e_eval_tsensor.h File Reference

This file contains all the functions prototypes for the stm3210e_tsensor.c firmware driver. More...

```
#include "stm3210e_eval.h" #include
"../Components/stlm75/stlm75.h"
```

Go to the source code of this file.

Defines

#define	**TSENSOR_I2C_ADDRESS_A01**	0x90
#define	**TSENSOR_I2C_ADDRESS_A02**	0x92
#define	**TSENSOR_MAX_TRIALS**	50

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.
uint8_t	**BSP_TSENSOR_ReadStatus** (void) Returns the Temperature Sensor status.
uint16_t	**BSP_TSENSOR_ReadTemp** (void) Read Temperature register of STLM75.

Detailed Description

This file contains all the functions prototypes for the stm3210e_tsensor.c firmware driver.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

© COPYRIGHT(c) 2015 STMicroelectronics

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

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

Definition in file **stm3210e_eval_tsensor.h**.

STM3210E_EVAL BSP User Manual

Modules

Here is a list of all modules:

- BSP
  - STM3210E-EVAL
    - STM3210E-EVAL Common
      - Private Types Definitions
      - Private Defines
      - Private Variables
      - Exported Functions
      - Bus Operations Functions
      - Link Operations Functions
      - Exported Types
      - Exported Constants
        - STM3210E_EVAL_LED
        - STM3210E_EVAL_BUTTON
        - STM3210E_EVAL_COM
        - STM3210E_EVAL_BUS
        - STM3210E_EVAL_COMPONENT
    - STM3210E-EVAL LCD
      - Private Defines
      - Private Macros
      - Private Variables
      - Private Functions
      - Exported Functions
      - Exported Types
      - Exported Constants
    - STM3210E_EVAL NAND

    - Private_Types_Definitions
    - Private_Defines
    - Private_Macros
    - Private_Variables
    - Private_Function_Prototypes
    - Exported_Functions
    - Exported_Types
    - Exported_Constants
    - Exported_Macro
- STM3210E_EVAL NOR
    - Private_Types_Definitions
    - Private_Defines
    - Private_Macros
    - Private_Variables
    - Private_Function_Prototypes
    - Exported_Functions
    - Exported_Types
    - Exported_Constants
    - Exported_Macro
- STM3210E-EVAL SD
    - Private Variables
    - Private Functions
    - Exported_Functions
    - Exported_Types
    - Exported_Constants
    - STM3210E_EVAL_SD_Exported_Functions
- STM3210E_EVAL Serial FLASH
    - Exported_Functions
    - Exported_Types
    - Exported_Constants
    - Exported_Macros
- STM3210E_EVAL SRAM
    - Private_Types_Definitions
    - Private_Defines
    - Private_Macros
    - Private_Variables
    - Private_Function_Prototypes

    - Exported_Functions
    - Exported_Types
    - Exported_Constants
    - Exported_Macro
- STM3210E_EVAL_AUDIO
    - AUDIO_Private_Types
    - AUDIO_Private_Defines
    - AUDIO Private_Macros
    - AUDIO_Private_Variables
    - AUDIO_Private_Function_Prototypes
    - AUDIO_OUT_Exported_Functions
    - AUDIO_Exported_Types
    - AUDIO_OUT_Exported_Constants
    - AUDIO_Exported_Macros
- STM3210E_EVAL TSENSOR
    - Private Variables
    - Exported Functions
    - Exported_Types
    - Exported_Constants

STM3210E_EVAL BSP User Manual

Data Structures

Here are the data structures with brief descriptions:

LCD_CONTROLLER_TypeDef	
LCD_DrawPropTypeDef	
Point	

STM3210E_EVAL BSP User Manual

File List

Here is a list of all files with brief descriptions:

stm3210e_eval.c [code]	This file provides a set of firmware functions to manage Leds, push-button and COM ports for STM3210E_EVAL
stm3210e_eval.h [code]	This file contains definitions for STM3210E_EVAL's LEDs, push-buttons and COM ports hardware resources
stm3210e_eval_audio.c [code]	This file provides the Audio driver for the STM3210E-Eval board
stm3210e_eval_audio.h [code]	This file contains the common defines and functions prototypes for **stm3210e_eval_audio.c** driver
stm3210e_eval_lcd.c [code]	This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM3210E-EVAL evaluation board

stm3210e_eval_lcd.h [code]	This file contains the common defines and functions prototypes for the **stm3210e_eval_lcd.c** driver
stm3210e_eval_nand.c [code]	This file includes a standard driver for the NAND512W3A2CN6E NAND flash memory device mounted on STM3210E-EVAL evaluation board
stm3210e_eval_nand.h [code]	This file contains the common defines and functions prototypes for the **stm3210e_eval_nand.c** driver
stm3210e_eval_nor.c [code]	This file includes a standard driver for the M29W128GL70ZA6E NOR flash memory device mounted on STM3210E-EVAL evaluation board
stm3210e_eval_nor.h [code]	This file contains the common defines and functions prototypes for the **stm3210e_eval_nor.c** driver
stm3210e_eval_sd.c [code]	This file includes the uSD card driver
stm3210e_eval_sd.h [code]	
stm3210e_eval_serialflash.c [code]	This file provides a set of functions needed to manage the SPI M25Pxx FLASH memory mounted on STM3210E_EVAL board. It

	implements a high level communication layer for read and write from/to this memory. The needed STM32 hardware resources (SPI and GPIO) are defined in **stm3210e_eval.h** file, and the initialization is performed in **FLASH_SPI_IO_Init()** function declared in **stm3210e_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **FLASH_SPI_IO_Init()** function
stm3210e_eval_serialflash.h [code]	This file contains the common defines and functions prototypes for the stm3210e_eval_flash driver
stm3210e_eval_sram.c [code]	This file includes the SRAM driver for the IS61WV51216BLL-10M memory device mounted on STM3210E-EVAL evaluation board
stm3210e_eval_sram.h [code]	This file contains the common defines and functions prototypes for the **stm3210e_eval_sram.c** driver
stm3210e_eval_tsensor.c [code]	This file provides a set of

	functions needed to manage the I2C STLM75 temperature sensor mounted on STM3210E_EVAL board . It implements a high level communication layer for read and write from/to this sensor. The needed STM32L152xD hardware resources (I2C and GPIO) are defined in **stm3210e_eval.h** file, and the initialization is performed in **TSENSOR_IO_Init()** function declared in **stm3210e_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **TSENSOR_IO_Init()** function
stm3210e_eval_tsensor.h [code]	This file contains all the functions prototypes for the stm3210e_tsensor.c firmware driver

STM3210E_EVAL BSP User Manual

Directories

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

- Drivers
  - BSP
    - STM3210E_EVAL

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Data Structure Index

L | P

L

LCD_CONTROLLER_TypeDef

LCD_DrawPropTypeDef

P

Point

L | P

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Private Types Definitions

STM3210E-EVAL Common

Data Structures

struct	**LCD_CONTROLLER_TypeDef**

STM3210E_EVAL BSP User Manual

stm3210e_eval.c

Go to the documentation of this file.

```
00001 /**
00002   ********************************************************************************
00003   * @file    stm3210e_eval.c
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file provides a set of firmware functions to manage Leds,
00008   *          push-button and COM ports for STM3210E_EVAL
00009   ********************************************************************************
00010   * @attention
00011   *
00012   * <h2><center>&copy; COPYRIGHT(c) 2015 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 /* Includes ------------------------------------------------------------------*/
00040 #include "stm3210e_eval.h"
00041
00042 /** @addtogroup BSP
00043   * @{
00044   */
00045
00046 /** @defgroup STM3210E_EVAL STM3210E-EVAL
00047   * @{
00048   */
00049
00050 /** @defgroup STM3210E_EVAL_Common STM3210E-EVAL Common
00051   * @{
00052   */
00053
00054 /** @defgroup STM3210E_EVAL_Private_TypesDefinitions Private Types Definitions
00055   * @{
00056   */
00057
00058 typedef struct
00059 {
00060   __IO uint16_t REG;
00061   __IO uint16_t RAM;
00062
00063 }LCD_CONTROLLER_TypeDef;
00064
00065 /**
00066   * @}
00067   */
```

```
00068
00069 /** @defgroup STM3210E_EVAL_Private_Defines
Private Defines
00070   * @{
00071   */
00072
00073 /**
00074  * @brief STM3210E EVAL BSP Driver version n
umber V6.0.0
00075    */
00076 #define __STM3210E_EVAL_BSP_VERSION_MAIN
   (0x06) /*!< [31:24] main version */
00077 #define __STM3210E_EVAL_BSP_VERSION_SUB1
   (0x00) /*!< [23:16] sub1 version */
00078 #define __STM3210E_EVAL_BSP_VERSION_SUB2
   (0x01) /*!< [15:8]  sub2 version */
00079 #define __STM3210E_EVAL_BSP_VERSION_RC
   (0x00) /*!< [7:0]  release candidate */
00080 #define __STM3210E_EVAL_BSP_VERSION
   ((__STM3210E_EVAL_BSP_VERSION_MAIN << 24)\
00081
   |(__STM3210E_EVAL_BSP_VERSION_SUB1 << 16)\
00082
   |(__STM3210E_EVAL_BSP_VERSION_SUB2 << 8 )\
00083
   |(__STM3210E_EVAL_BSP_VERSION_RC))
00084
00085
00086 /* Note: LCD /CS is CE4 - Bank 4 of NOR/SRAM
 Bank 1~4 */
00087 #define TFT_LCD_BASE            FSMC_BANK1_4
00088 #define TFT_LCD                 ((LCD_CONTROL
LER_TypeDef *) TFT_LCD_BASE)
00089
00090 /**
00091   * @}
00092   */
```

```
00093
00094
00095 /** @defgroup STM3210E_EVAL_Private_Variable
s Private Variables
00096   * @{
00097   */
00098 /**
00099  * @brief LED variables
00100  */
00101 GPIO_TypeDef* LED_PORT[LEDn] = {LED1_GPIO_PO
RT,
00102                                 LED2_GPIO_PO
RT,
00103                                 LED3_GPIO_PO
RT,
00104                                 LED4_GPIO_PO
RT};
00105
00106 const uint16_t LED_PIN[LEDn] = {LED1_PIN,
00107                                 LED2_PIN,
00108                                 LED3_PIN,
00109                                 LED4_PIN};
00110
00111 /**
00112  * @brief BUTTON variables
00113  */
00114 GPIO_TypeDef* BUTTON_PORT[BUTTONn] = {WAKEUP
_BUTTON_GPIO_PORT,
00115                                       TAMPER
_BUTTON_GPIO_PORT,
00116                                       KEY_BU
TTON_GPIO_PORT,
00117                                       SEL_JO
Y_GPIO_PORT,
00118                                       LEFT_J
OY_GPIO_PORT,
00119                                       RIGHT_
```

```
JOY_GPIO_PORT,
00120                                                 DOWN_J
OY_GPIO_PORT,
00121                                                 UP_JOY
_GPIO_PORT};
00122
00123 const uint16_t BUTTON_PIN[BUTTONn] = {WAKEUP
_BUTTON_PIN,
00124                                                 TAMPER
_BUTTON_PIN,
00125                                                 KEY_BU
TTON_PIN,
00126                                                 SEL_JO
Y_PIN,
00127                                                 LEFT_J
OY_PIN,
00128                                                 RIGHT_
JOY_PIN,
00129                                                 DOWN_J
OY_PIN,
00130                                                 UP_JOY
_PIN};
00131
00132 const uint8_t BUTTON_IRQn[BUTTONn] = {WAKEUP
_BUTTON_EXTI_IRQn,
00133                                                 TAMPER
_BUTTON_EXTI_IRQn,
00134                                                 KEY_BU
TTON_EXTI_IRQn,
00135                                                 SEL_JO
Y_EXTI_IRQn,
00136                                                 LEFT_J
OY_EXTI_IRQn,
00137                                                 RIGHT_
JOY_EXTI_IRQn,
00138                                                 DOWN_J
OY_EXTI_IRQn,
```

```
00139                                                   UP_JOY
_EXTI_IRQn};
00140
00141 /**
00142  * @brief JOYSTICK variables
00143  */
00144 GPIO_TypeDef* JOY_PORT[JOYn] = {SEL_JOY_GPIO
_PORT,
00145                                 LEFT_JOY_GPI
O_PORT,
00146                                 RIGHT_JOY_GP
IO_PORT,
00147                                 DOWN_JOY_GPI
O_PORT,
00148                                 UP_JOY_GPIO_
PORT};
00149
00150 const uint16_t JOY_PIN[JOYn] = {SEL_JOY_PIN,

00151                                 LEFT_JOY_PIN
,
00152                                 RIGHT_JOY_PIN
,
00153                                 DOWN_JOY_PIN
,
00154                                 UP_JOY_PIN};

00155
00156 const uint8_t JOY_IRQn[JOYn] = {SEL_JOY_EXTI
_IRQn,
00157                                 LEFT_JOY_EXT
I_IRQn,
00158                                 RIGHT_JOY_EX
TI_IRQn,
00159                                 DOWN_JOY_EXT
I_IRQn,
00160                                 UP_JOY_EXTI_
```

```
IRQn};
00161
00162 /**
00163  * @brief COM variables
00164  */
00165 USART_TypeDef* COM_USART[COMn]   = {EVAL_COM1
, EVAL_COM2};
00166
00167 GPIO_TypeDef* COM_TX_PORT[COMn]   = {EVAL_CO
M1_TX_GPIO_PORT, EVAL_COM2_TX_GPIO_PORT};
00168
00169 GPIO_TypeDef* COM_RX_PORT[COMn]   = {EVAL_CO
M1_RX_GPIO_PORT, EVAL_COM2_RX_GPIO_PORT};
00170
00171 const uint16_t COM_TX_PIN[COMn]   = {EVAL_CO
M1_TX_PIN, EVAL_COM2_TX_PIN};
00172
00173 const uint16_t COM_RX_PIN[COMn]   = {EVAL_CO
M1_RX_PIN, EVAL_COM2_RX_PIN};
00174
00175 /**
00176  * @brief BUS variables
00177  */
00178 #ifdef HAL_SPI_MODULE_ENABLED
00179 uint32_t SpixTimeout = EVAL_SPIx_TIMEOUT_MAX
;        /*<! Value of Timeout when SPI communicat
ion fails */
00180 static SPI_HandleTypeDef heval_Spi;
00181 #endif /* HAL_SPI_MODULE_ENABLED */
00182
00183 #ifdef HAL_I2C_MODULE_ENABLED
00184 uint32_t I2cxTimeout = EVAL_I2Cx_TIMEOUT_MAX
;   /*<! Value of Timeout when I2C communication f
ails */
00185 I2C_HandleTypeDef heval_I2c;
00186 #endif /* HAL_I2C_MODULE_ENABLED */
00187
```

```
00188 /**
00189   * @}
00190   */
00191
00192 #if defined(HAL_SRAM_MODULE_ENABLED)
00193
00194 static void     FSMC_BANK1NORSRAM4_WriteData
(uint16_t Data);
00195 static void     FSMC_BANK1NORSRAM4_WriteReg(
uint8_t Reg);
00196 static uint16_t FSMC_BANK1NORSRAM4_ReadData(
uint8_t Reg);
00197 static void     FSMC_BANK1NORSRAM4_Init(void
);
00198 static void     FSMC_BANK1NORSRAM4_MspInit(v
oid);
00199
00200 /* LCD IO functions */
00201 void            LCD_IO_Init(void);
00202 void            LCD_IO_WriteData(uint16_t Re
gValue);
00203 void            LCD_IO_WriteMultipleData(uin
t8_t *pData, uint32_t Size);
00204 void            LCD_IO_WriteReg(uint8_t Reg)
;
00205 uint16_t        LCD_IO_ReadData(uint16_t Reg
);
00206 void            LCD_Delay (uint32_t delay);
00207 #endif /*HAL_SRAM_MODULE_ENABLED*/
00208
00209 /* I2Cx bus function */
00210 #ifdef HAL_I2C_MODULE_ENABLED
00211 /* Link function for I2C EEPROM peripheral */

00212 static void               I2Cx_Init(void);
00213 static void               I2Cx_WriteData(uin
t16_t Addr, uint8_t Reg, uint8_t Value);
```

```
00214 static HAL_StatusTypeDef  I2Cx_WriteBuffer(u
int16_t Addr, uint8_t Reg, uint16_t RegSize, uint8
_t *pBuffer, uint16_t Length);
00215 static uint8_t             I2Cx_ReadData(uint
16_t Addr, uint8_t Reg);
00216 static HAL_StatusTypeDef  I2Cx_ReadBuffer(ui
nt16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_
t *pBuffer, uint16_t Length);
00217 static HAL_StatusTypeDef  I2Cx_IsDeviceReady
(uint16_t DevAddress, uint32_t Trials);
00218 static void               I2Cx_Error (void);
00219 static void               I2Cx_MspInit(I2C_H
andleTypeDef *hi2c);
00220
00221 /* Link functions for Temperature Sensor per
ipheral */
00222 void                      TSENSOR_IO_Init(vo
id);
00223 void                      TSENSOR_IO_Write(u
int16_t DevAddress, uint8_t* pBuffer, uint8_t Writ
eAddr, uint16_t Length);
00224 void                      TSENSOR_IO_Read(ui
nt16_t DevAddress, uint8_t* pBuffer, uint8_t ReadA
ddr, uint16_t Length);
00225 uint16_t                  TSENSOR_IO_IsDevic
eReady(uint16_t DevAddress, uint32_t Trials);
00226
00227 /* Link function for Audio peripheral */
00228 void                      AUDIO_IO_Init(void
);
00229 void                      AUDIO_IO_Write(uin
t8_t Addr, uint8_t Reg, uint8_t Value);
00230 uint8_t                   AUDIO_IO_Read(uint
8_t Addr, uint8_t Reg);
00231
00232 #endif /* HAL_I2C_MODULE_ENABLED */
00233
```

```
00234 #ifdef HAL_SPI_MODULE_ENABLED
00235 /* SPIx bus function */
00236 static HAL_StatusTypeDef  SPIx_Init(void);
00237 static uint8_t            SPIx_Write(uint8_t
 Value);
00238 static uint8_t            SPIx_Read(void);
00239 static void               SPIx_Error (void);
00240 static void               SPIx_MspInit(SPI_H
andleTypeDef *hspi);
00241
00242 /* Link function for EEPROM peripheral over
SPI */
00243 HAL_StatusTypeDef         FLASH_SPI_IO_Init(
void);
00244 uint8_t                   FLASH_SPI_IO_Write
Byte(uint8_t Data);
00245 uint8_t                   FLASH_SPI_IO_ReadB
yte(void);
00246 HAL_StatusTypeDef         FLASH_SPI_IO_ReadD
ata(uint32_t MemAddress, uint8_t* pBuffer, uint32_
t BufferSize);
00247 void                      FLASH_SPI_IO_Write
Enable(void);
00248 HAL_StatusTypeDef         FLASH_SPI_IO_WaitF
orWriteEnd(void);
00249 uint32_t                  FLASH_SPI_IO_ReadID
(void);
00250 #endif /* HAL_SPI_MODULE_ENABLED */
00251
00252 /** @defgroup STM3210E_EVAL_Exported_Functio
ns Exported Functions
00253   * @{
00254   */
00255
00256 /**
00257   * @brief  This method returns the STM3210E
 EVAL BSP Driver revision
```

```
00258   * @retval version : 0xXYZR (8bits for each
 decimal, R for RC)
00259   */
00260 uint32_t BSP_GetVersion(void)
00261 {
00262   return __STM3210E_EVAL_BSP_VERSION;
00263 }
00264
00265 /**
00266   * @brief  Configures LED GPIO.
00267   * @param  Led: Specifies the Led to be con
figured.
00268   *   This parameter can be one of following
 parameters:
00269   *     @arg LED1
00270   *     @arg LED2
00271   *     @arg LED3
00272   *     @arg LED4
00273   * @retval None
00274   */
00275 void BSP_LED_Init(Led_TypeDef Led)
00276 {
00277   GPIO_InitTypeDef  gpioinitstruct = {0};
00278
00279   /* Enable the GPIO_LED clock */
00280   LEDx_GPIO_CLK_ENABLE(Led);
00281
00282   /* Configure the GPIO_LED pin */
00283   gpioinitstruct.Pin    = LED_PIN[Led];
00284   gpioinitstruct.Mode   = GPIO_MODE_OUTPUT_P
P;
00285   gpioinitstruct.Pull   = GPIO_NOPULL;
00286   gpioinitstruct.Speed  = GPIO_SPEED_FREQ_HI
GH;
00287
00288   HAL_GPIO_Init(LED_PORT[Led], &gpioinitstru
ct);
```

```
00289
00290   HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[L
ed], GPIO_PIN_RESET);
00291 }
00292
00293 /**
00294   * @brief  Turns selected LED On.
00295   * @param  Led: Specifies the Led to be set
 on.
00296   *   This parameter can be one of following
 parameters:
00297   *     @arg LED1
00298   *     @arg LED2
00299   *     @arg LED3
00300   *     @arg LED4
00301   * @retval None
00302   */
00303 void BSP_LED_On(Led_TypeDef Led)
00304 {
00305   HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[L
ed], GPIO_PIN_SET);
00306 }
00307
00308 /**
00309   * @brief  Turns selected LED Off.
00310   * @param  Led: Specifies the Led to be set
 off.
00311   *   This parameter can be one of following
 parameters:
00312   *     @arg LED1
00313   *     @arg LED2
00314   *     @arg LED3
00315   *     @arg LED4
00316   * @retval None
00317   */
00318 void BSP_LED_Off(Led_TypeDef Led)
00319 {
```

```
00320   HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[L
ed], GPIO_PIN_RESET);
00321 }
00322
00323 /**
00324   * @brief  Toggles the selected LED.
00325   * @param  Led: Specifies the Led to be tog
gled.
00326   *   This parameter can be one of following
 parameters:
00327   *     @arg LED1
00328   *     @arg LED2
00329   *     @arg LED3
00330   *     @arg LED4
00331   * @retval None
00332   */
00333 void BSP_LED_Toggle(Led_TypeDef Led)
00334 {
00335   HAL_GPIO_TogglePin(LED_PORT[Led], LED_PIN[
Led]);
00336 }
00337
00338 /**
00339   * @brief  Configures push button GPIO and
EXTI Line.
00340   * @param  Button: Button to be configured.
00341   *   This parameter can be one of the follo
wing values:
00342   *     @arg BUTTON_TAMPER: Key/Tamper Push
Button
00343   *     @arg BUTTON_SEL   : Sel Push Button
on Joystick
00344   *     @arg BUTTON_LEFT  : Left Push Button
 on Joystick
00345   *     @arg BUTTON_RIGHT : Right Push Butto
n on Joystick
00346   *     @arg BUTTON_DOWN  : Down Push Button
```

```
 on Joystick
00347   *     @arg BUTTON_UP    : Up Push Button o
n Joystick
00348   * @param  Button_Mode: Button mode request
ed.
00349   *   This parameter can be one of the follo
wing values:
00350   *     @arg BUTTON_MODE_GPIO: Button will b
e used as simple IO
00351   *     @arg BUTTON_MODE_EVT : Button will b
e connected to EXTI line
00352   *                            with event ge
neration capability
00353   *     @arg BUTTON_MODE_EXTI: Button will b
e connected to EXTI line
00354   *                            with interrup
t generation capability
00355   * @retval None
00356   */
00357 void BSP_PB_Init(Button_TypeDef Button, Butt
onMode_TypeDef Button_Mode)
00358 {
00359   GPIO_InitTypeDef gpioinitstruct = {0};
00360
00361   /* Enable the corresponding Push Button cl
ock */
00362   BUTTONx_GPIO_CLK_ENABLE(Button);
00363
00364   /* Configure Push Button pin as input */
00365   gpioinitstruct.Pin    = BUTTON_PIN[Button]
;
00366   gpioinitstruct.Pull   = GPIO_PULLDOWN;
00367   gpioinitstruct.Speed  = GPIO_SPEED_FREQ_HI
GH;
00368
00369   if (Button_Mode == BUTTON_MODE_GPIO)
00370   {
```

```
00371     /* Configure Button pin as input */
00372     gpioinitstruct.Mode = GPIO_MODE_INPUT;
00373     HAL_GPIO_Init(BUTTON_PORT[Button], &gpio
initstruct);
00374   }
00375   else if (Button_Mode == BUTTON_MODE_EXTI)
00376   {
00377     if(Button != BUTTON_WAKEUP)
00378     {
00379       /* Configure Joystick Button pin as in
put with External interrupt, falling edge */
00380       gpioinitstruct.Mode = GPIO_MODE_IT_FAL
LING;
00381     }
00382     else
00383     {
00384       /* Configure Key Push Button pin as in
put with External interrupt, rising edge */
00385       gpioinitstruct.Mode = GPIO_MODE_IT_RIS
ING;
00386     }
00387     HAL_GPIO_Init(BUTTON_PORT[Button], &gpio
initstruct);
00388
00389     /* Enable and set Button EXTI Interrupt
to the lowest priority */
00390     HAL_NVIC_SetPriority((IRQn_Type)(BUTTON_
IRQn[Button]), 0x0F, 0);
00391     HAL_NVIC_EnableIRQ((IRQn_Type)(BUTTON_IR
Qn[Button]));
00392   }
00393   else if (Button_Mode == BUTTON_MODE_EVT)
00394   {
00395     if(Button != BUTTON_WAKEUP)
00396     {
00397       /* Configure Joystick Button pin as in
put with External interrupt, falling edge */
```

```
00398       gpioinitstruct.Mode = GPIO_MODE_EVT_FA
LLING;
00399     }
00400     else
00401     {
00402       /* Configure Key Push Button pin as in
put with External interrupt, rising edge */
00403       gpioinitstruct.Mode = GPIO_MODE_EVT_RI
SING;
00404     }
00405     HAL_GPIO_Init(BUTTON_PORT[Button], &gpio
initstruct);
00406   }
00407 }
00408
00409 /**
00410   * @brief  Returns the selected button stat
e.
00411   * @param  Button: Button to be checked.
00412   *   This parameter can be one of the follo
wing values:
00413   *     @arg BUTTON_TAMPER: Key/Tamper Push
Button
00414   * @retval Button state
00415   */
00416 uint32_t BSP_PB_GetState(Button_TypeDef Butt
on)
00417 {
00418   return HAL_GPIO_ReadPin(BUTTON_PORT[Button
], BUTTON_PIN[Button]);
00419 }
00420
00421 /**
00422   * @brief  Configures all button of the joy
stick in GPIO or EXTI modes.
00423   * @param  Joy_Mode: Joystick mode.
00424   *    This parameter can be one of the foll
```

```
owing values:
00425   *     @arg  JOY_MODE_GPIO: Joystick pins w
ill be used as simple IOs
00426   *     @arg  JOY_MODE_EXTI: Joystick pins w
ill be connected to EXTI line
00427   *                                  with int
errupt generation capability
00428   * @retval HAL_OK: if all initializations a
re OK. Other value if error.
00429   */
00430 uint8_t BSP_JOY_Init(JOYMode_TypeDef Joy_Mod
e)
00431 {
00432   JOYState_TypeDef joykey = JOY_NONE;
00433   GPIO_InitTypeDef gpioinitstruct = {0};
00434
00435   /* Initialized the Joystick. */
00436   for(joykey = JOY_SEL; joykey < (JOY_SEL+JO
Yn) ; joykey++)
00437   {
00438     /* Enable the JOY clock */
00439     JOYx_GPIO_CLK_ENABLE(joykey);
00440
00441     gpioinitstruct.Pin    = JOY_PIN[joykey];
00442     gpioinitstruct.Pull   = GPIO_NOPULL;
00443     gpioinitstruct.Speed  = GPIO_SPEED_FREQ_
HIGH;
00444
00445     if (Joy_Mode == JOY_MODE_GPIO)
00446     {
00447       /* Configure Joy pin as input */
00448       gpioinitstruct.Mode = GPIO_MODE_INPUT;
00449       HAL_GPIO_Init(JOY_PORT[joykey], &gpioi
nitstruct);
00450     }
00451
00452     if (Joy_Mode == JOY_MODE_EXTI)
```

```
00453     {
00454       /* Configure Joy pin as input with Ext
ernal interrupt */
00455       gpioinitstruct.Mode = GPIO_MODE_IT_FAL
LING;
00456       HAL_GPIO_Init(JOY_PORT[joykey], &gpioi
nitstruct);
00457
00458       /* Enable and set Joy EXTI Interrupt t
o the lowest priority */
00459       HAL_NVIC_SetPriority((IRQn_Type)(JOY_I
RQn[joykey]), 0x0F, 0);
00460       HAL_NVIC_EnableIRQ((IRQn_Type)(JOY_IRQn
[joykey]));
00461     }
00462   }
00463
00464   return HAL_OK;
00465 }
00466
00467 /**
00468   * @brief  Returns the current joystick sta
tus.
00469   * @retval Code of the joystick key pressed
00470   *          This code can be one of the fol
lowing values:
00471   *            @arg  JOY_SEL
00472   *            @arg  JOY_DOWN
00473   *            @arg  JOY_LEFT
00474   *            @arg  JOY_RIGHT
00475   *            @arg  JOY_UP
00476   *            @arg  JOY_NONE
00477   */
00478 JOYState_TypeDef BSP_JOY_GetState(void)
00479 {
00480   JOYState_TypeDef joykey = JOY_NONE;
00481
```

```
00482   for(joykey = JOY_SEL; joykey < (JOY_SEL+JO
Yn) ; joykey++)
00483   {
00484     if(HAL_GPIO_ReadPin(JOY_PORT[joykey], JO
Y_PIN[joykey]) == GPIO_PIN_RESET)
00485     {
00486       /* Return Code Joystick key pressed */
00487       return joykey;
00488     }
00489   }
00490
00491   /* No Joystick key pressed */
00492   return JOY_NONE;
00493 }
00494
00495 #ifdef HAL_UART_MODULE_ENABLED
00496 /**
00497   * @brief  Configures COM port.
00498   * @param  COM: Specifies the COM port to b
e configured.
00499   *   This parameter can be one of following
 parameters:
00500   *     @arg COM1
00501   * @param  huart: pointer to a UART_HandleT
ypeDef structure that
00502   *   contains the configuration information
 for the specified UART peripheral.
00503   * @retval None
00504   */
00505 void BSP_COM_Init(COM_TypeDef COM, UART_Hand
leTypeDef* huart)
00506 {
00507   GPIO_InitTypeDef gpioinitstruct = {0};
00508
00509   /* Enable GPIO clock */
00510   COMx_TX_GPIO_CLK_ENABLE(COM);
00511   COMx_RX_GPIO_CLK_ENABLE(COM);
```

```
00512
00513   /* Enable USART clock */
00514   COMx_CLK_ENABLE(COM);
00515
00516   /* Configure USART Tx as alternate functio
n push-pull */
00517   gpioinitstruct.Pin        = COM_TX_PIN[COM
];
00518   gpioinitstruct.Mode       = GPIO_MODE_AF_P
P;
00519   gpioinitstruct.Speed      = GPIO_SPEED_FRE
Q_HIGH;
00520   gpioinitstruct.Pull       = GPIO_PULLUP;
00521   HAL_GPIO_Init(COM_TX_PORT[COM], &gpioinits
truct);
00522
00523   /* Configure USART Rx as alternate functio
n push-pull */
00524   gpioinitstruct.Mode       = GPIO_MODE_INPU
T;
00525   gpioinitstruct.Pin        = COM_RX_PIN[COM
];
00526   HAL_GPIO_Init(COM_RX_PORT[COM], &gpioinits
truct);
00527
00528   /* USART configuration */
00529   huart->Instance = COM_USART[COM];
00530   HAL_UART_Init(huart);
00531 }
00532 #endif /* HAL_UART_MODULE_ENABLED */
00533
00534 /**
00535   * @}
00536   */
00537
00538 /** @defgroup STM3210E_EVAL_BusOperations_Fu
nctions Bus Operations Functions
```

```
  * @{
  */

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

/************************** FSMC Routines *
**********************************/
#if defined(HAL_SRAM_MODULE_ENABLED)
/**
  * @brief  Initializes FSMC_BANK4 MSP.
  * @retval None
  */
static void FSMC_BANK1NORSRAM4_MspInit(void)
{
  GPIO_InitTypeDef gpioinitstruct = {0};

  /* Enable FMC clock */
  __HAL_RCC_FSMC_CLK_ENABLE();

  /* Enable GPIOs clock */
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOE_CLK_ENABLE();
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOG_CLK_ENABLE();
  __HAL_RCC_AFIO_CLK_ENABLE();

  /* Common GPIO configuration */
  gpioinitstruct.Mode      = GPIO_MODE_AF_PP
;
  gpioinitstruct.Speed     = GPIO_SPEED_FREQ
_HIGH;

  /* Set PD.00(D2), PD.01(D3), PD.04(NOE), P
```

```
D.05(NWE), PD.08(D13), PD.09(D14),
00571      PD.10(D15), PD.14(D0), PD.15(D1) as alt
ernate function push pull */
00572   gpioinitstruct.Pin = GPIO_PIN_0 | GPIO_PIN
_1 | GPIO_PIN_4 | GPIO_PIN_5 |
00573                                       GPIO_PIN_8 |
 GPIO_PIN_9 | GPIO_PIN_10 | GPIO_PIN_14 |
00574                                       GPIO_PIN_15;
00575   HAL_GPIO_Init(GPIOD, &gpioinitstruct);
00576
00577   /* Set PE.07(D4), PE.08(D5), PE.09(D6), PE
.10(D7), PE.11(D8), PE.12(D9), PE.13(D10),
00578      PE.14(D11), PE.15(D12) as alternate fun
ction push pull */
00579   gpioinitstruct.Pin = GPIO_PIN_7 | GPIO_PIN
_8 | GPIO_PIN_9 | GPIO_PIN_10 |
00580                                       GPIO_PIN_11
| GPIO_PIN_12 | GPIO_PIN_13 | GPIO_PIN_14 |
00581                                       GPIO_PIN_15;
00582   HAL_GPIO_Init(GPIOE, &gpioinitstruct);
00583
00584   /* Set PF.00(A0 (RS)) as alternate functio
n push pull */
00585   gpioinitstruct.Pin = GPIO_PIN_0;
00586   HAL_GPIO_Init(GPIOF, &gpioinitstruct);
00587
00588   /* Set PG.12(NE4 (LCD/CS)) as alternate fu
nction push pull - CE3(LCD /CS) */
00589   gpioinitstruct.Pin = GPIO_PIN_12;
00590   HAL_GPIO_Init(GPIOG, &gpioinitstruct);
00591 }
00592
00593 /**
00594   * @brief  Initializes LCD IO.
00595   * @retval None
00596   */
00597 static void FSMC_BANK1NORSRAM4_Init(void)
```

```
00598 {
00599   SRAM_HandleTypeDef           hsram;
00600   FSMC_NORSRAM_TimingTypeDef  sramtiming = {
0};
00601
00602   /*** Configure the SRAM Bank 4 ***/
00603   /* Configure IPs */
00604   hsram.Instance  = FSMC_NORSRAM_DEVICE;
00605   hsram.Extended  = FSMC_NORSRAM_EXTENDED_DE
VICE;
00606
00607   sramtiming.AddressSetupTime       = 1;
00608   sramtiming.AddressHoldTime        = 1;
00609   sramtiming.DataSetupTime          = 2;
00610   sramtiming.BusTurnAroundDuration  = 1;
00611   sramtiming.CLKDivision            = 2;
00612   sramtiming.DataLatency            = 2;
00613   sramtiming.AccessMode             = FSMC_A
CCESS_MODE_A;
00614
00615   /* Color LCD configuration
00616      LCD configured as follow:
00617         - Data/Address MUX = Disable
00618         - Memory Type = SRAM
00619         - Data Width = 16bit
00620         - Write Operation = Enable
00621         - Extended Mode = Enable
00622         - Asynchronous Wait = Disable */
00623   hsram.Init.NSBank             = FSMC_NORSR
AM_BANK4;
00624   hsram.Init.DataAddressMux     = FSMC_DATA_
ADDRESS_MUX_DISABLE;
00625   hsram.Init.MemoryType         = FSMC_MEMOR
Y_TYPE_SRAM;
00626   hsram.Init.MemoryDataWidth    = FSMC_NORSR
AM_MEM_BUS_WIDTH_16;
00627   hsram.Init.BurstAccessMode    = FSMC_BURST
```

```
_ACCESS_MODE_DISABLE;
00628   hsram.Init.WaitSignalPolarity = FSMC_WAIT_
SIGNAL_POLARITY_LOW;
00629   hsram.Init.WrapMode           = FSMC_WRAP_
MODE_DISABLE;
00630   hsram.Init.WaitSignalActive   = FSMC_WAIT_
TIMING_BEFORE_WS;
00631   hsram.Init.WriteOperation     = FSMC_WRITE
_OPERATION_ENABLE;
00632   hsram.Init.WaitSignal         = FSMC_WAIT_
SIGNAL_DISABLE;
00633   hsram.Init.ExtendedMode       = FSMC_EXTEN
DED_MODE_DISABLE;
00634   hsram.Init.AsynchronousWait   = FSMC_ASYNC
HRONOUS_WAIT_DISABLE;
00635   hsram.Init.WriteBurst         = FSMC_WRITE
_BURST_DISABLE;
00636
00637   /* Initialize the SRAM controller */
00638   FSMC_BANK1NORSRAM4_MspInit();
00639   HAL_SRAM_Init(&hsram, &sramtiming, &sramti
ming);
00640 }
00641
00642 /**
00643   * @brief  Writes register value.
00644   * @param  Data:
00645   * @retval None
00646   */
00647 static void FSMC_BANK1NORSRAM4_WriteData(uin
t16_t Data)
00648 {
00649   /* Write 16-bit Data */
00650   TFT_LCD->RAM = Data;
00651 }
00652
00653 /**
```

```
00654   * @brief  Writes register address.
00655   * @param  Reg:
00656   * @retval None
00657   */
00658 static void FSMC_BANK1NORSRAM4_WriteReg(uint
8_t Reg)
00659 {
00660   /* Write 16-bit Index, then Write Reg */
00661   TFT_LCD->REG = Reg;
00662 }
00663
00664 /**
00665   * @brief  Reads register value.
00666   * @retval Read value
00667   */
00668 static uint16_t FSMC_BANK1NORSRAM4_ReadData(
uint8_t Reg)
00669 {
00670   /* Write 16-bit Index (then Read Reg) */
00671   TFT_LCD->REG = Reg;
00672   /* Read 16-bit Reg */
00673   return (TFT_LCD->RAM);
00674 }
00675 #endif /*HAL_SRAM_MODULE_ENABLED*/
00676
00677 #ifdef HAL_I2C_MODULE_ENABLED
00678 /****************************** I2C Routine
s*****************************/
00679
00680 /**
00681   * @brief Eval I2Cx MSP Initialization
00682   * @param hi2c: I2C handle
00683   * @retval None
00684   */
00685 static void I2Cx_MspInit(I2C_HandleTypeDef *
hi2c)
00686 {
```

```
00687   GPIO_InitTypeDef  gpioinitstruct = {0};
00688
00689   if (hi2c->Instance == EVAL_I2Cx)
00690   {
00691     /*## Configure the GPIOs ###############
#####################################*/
00692
00693     /* Enable GPIO clock */
00694     EVAL_I2Cx_SDA_GPIO_CLK_ENABLE();
00695     EVAL_I2Cx_SCL_GPIO_CLK_ENABLE();
00696
00697     /* Configure I2C Tx as alternate functio
n  */
00698     gpioinitstruct.Pin       = EVAL_I2Cx_SCL
_PIN;
00699     gpioinitstruct.Mode      = GPIO_MODE_AF_
OD;
00700     gpioinitstruct.Pull      = GPIO_NOPULL;
00701     gpioinitstruct.Speed     = GPIO_SPEED_FR
EQ_HIGH;
00702     HAL_GPIO_Init(EVAL_I2Cx_SCL_GPIO_PORT, &
gpioinitstruct);
00703
00704     /* Configure I2C Rx as alternate functio
n  */
00705     gpioinitstruct.Pin = EVAL_I2Cx_SDA_PIN;
00706     HAL_GPIO_Init(EVAL_I2Cx_SDA_GPIO_PORT, &
gpioinitstruct);
00707
00708
00709     /*## Configure the Eval I2Cx peripheral
#####################################*/
00710     /* Enable Eval_I2Cx clock */
00711     EVAL_I2Cx_CLK_ENABLE();
00712
00713     /* Add delay related to RCC workaround */
```

```
00714     while (READ_BIT(RCC->APB1ENR, RCC_APB1EN
R_I2C1EN) != RCC_APB1ENR_I2C1EN) {};
00715
00716     /* Force the I2C Periheral Clock Reset */

00717     EVAL_I2Cx_FORCE_RESET();
00718
00719     /* Release the I2C Periheral Clock Reset
 */
00720     EVAL_I2Cx_RELEASE_RESET();
00721
00722     /* Enable and set Eval I2Cx Interrupt to
 the highest priority */
00723     HAL_NVIC_SetPriority(EVAL_I2Cx_EV_IRQn,
5, 0);
00724     HAL_NVIC_EnableIRQ(EVAL_I2Cx_EV_IRQn);
00725
00726     /* Enable and set Eval I2Cx Interrupt to
 the highest priority */
00727     HAL_NVIC_SetPriority(EVAL_I2Cx_ER_IRQn,
5, 0);
00728     HAL_NVIC_EnableIRQ(EVAL_I2Cx_ER_IRQn);
00729   }
00730 }
00731
00732 /**
00733   * @brief Eval I2Cx Bus initialization
00734   * @retval None
00735   */
00736 static void I2Cx_Init(void)
00737 {
00738   if(HAL_I2C_GetState(&heval_I2c) == HAL_I2C
_STATE_RESET)
00739   {
00740     heval_I2c.Instance              = EVAL_I
2Cx;
00741     heval_I2c.Init.ClockSpeed       = BSP_I2
```

```
C_SPEED;
00742     heval_I2c.Init.DutyCycle        = I2C_DU
TYCYCLE_2;
00743     heval_I2c.Init.OwnAddress1      = 0;
00744     heval_I2c.Init.AddressingMode   = I2C_AD
DRESSINGMODE_7BIT;
00745     heval_I2c.Init.DualAddressMode  = I2C_DU
ALADDRESS_DISABLE;
00746     heval_I2c.Init.OwnAddress2      = 0;
00747     heval_I2c.Init.GeneralCallMode  = I2C_GE
NERALCALL_DISABLE;
00748     heval_I2c.Init.NoStretchMode    = I2C_NO
STRETCH_DISABLE;
00749
00750     /* Init the I2C */
00751     I2Cx_MspInit(&heval_I2c);
00752     HAL_I2C_Init(&heval_I2c);
00753   }
00754 }
00755
00756 /**
00757   * @brief  Write a value in a register of t
he device through BUS.
00758   * @param  Addr: Device address on BUS Bus.

00759   * @param  Reg: The target register address
 to write
00760   * @param  Value: The target register value
 to be written
00761   * @retval  None
00762   */
00763 static void I2Cx_WriteData(uint16_t Addr, ui
nt8_t Reg, uint8_t Value)
00764 {
00765   HAL_StatusTypeDef status = HAL_OK;
00766
00767   status = HAL_I2C_Mem_Write(&heval_I2c, Add
```

```
r, (uint16_t)Reg, I2C_MEMADD_SIZE_8BIT, &Value, 1,
I2cxTimeout);
00768
00769   /* Check the communication status */
00770   if(status != HAL_OK)
00771   {
00772     /* Execute user timeout callback */
00773     I2Cx_Error();
00774   }
00775 }
00776
00777 /**
00778   * @brief  Write a value in a register of t
he device through BUS.
00779   * @param  Addr: Device address on BUS Bus.

00780   * @param  Reg: The target register address
 to write
00781   * @param  RegSize: The target register siz
e (can be 8BIT or 16BIT)
00782   * @param  pBuffer: The target register val
ue to be written
00783   * @param  Length: buffer size to be written

00784   * @retval None
00785   */
00786 static HAL_StatusTypeDef I2Cx_WriteBuffer(ui
nt16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_
t *pBuffer, uint16_t Length)
00787 {
00788   HAL_StatusTypeDef status = HAL_OK;
00789
00790   status = HAL_I2C_Mem_Write(&heval_I2c, Add
r, (uint16_t)Reg, RegSize, pBuffer, Length, I2cxTi
meout);
00791
00792 /* Check the communication status */
```

```
00793   if(status != HAL_OK)
00794   {
00795     /* Re-Initiaize the BUS */
00796     I2Cx_Error();
00797   }
00798   return status;
00799 }
00800
00801 /**
00802   * @brief  Read a value in a register of th
e device through BUS.
00803   * @param  Addr: Device address on BUS Bus.

00804   * @param  Reg: The target register address
 to write
00805   * @retval Data read at register @
00806   */
00807 static uint8_t I2Cx_ReadData(uint16_t Addr,
uint8_t Reg)
00808 {
00809   HAL_StatusTypeDef status = HAL_OK;
00810   uint8_t value = 0;
00811
00812   status = HAL_I2C_Mem_Read(&heval_I2c, Addr
, Reg, I2C_MEMADD_SIZE_8BIT, &value, 1, I2cxTimeout
);
00813
00814   /* Check the communication status */
00815   if(status != HAL_OK)
00816   {
00817     /* Execute user timeout callback */
00818     I2Cx_Error();
00819
00820   }
00821   return value;
00822 }
00823
```

```
00824 /**
00825   * @brief  Reads multiple data on the BUS.
00826   * @param  Addr: I2C Address
00827   * @param  Reg: Reg Address
00828   * @param  RegSize : The target register si
ze (can be 8BIT or 16BIT)
00829   * @param  pBuffer: pointer to read data bu
ffer
00830   * @param  Length: length of the data
00831   * @retval 0 if no problems to read multipl
e data
00832   */
00833 static HAL_StatusTypeDef I2Cx_ReadBuffer(uin
t16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t
 *pBuffer, uint16_t Length)
00834 {
00835   HAL_StatusTypeDef status = HAL_OK;
00836
00837   status = HAL_I2C_Mem_Read(&heval_I2c, Addr
, (uint16_t)Reg, RegSize, pBuffer, Length, I2cxTim
eout);
00838
00839   /* Check the communication status */
00840   if(status != HAL_OK)
00841   {
00842     /* Re-Initiaize the BUS */
00843     I2Cx_Error();
00844   }
00845   return status;
00846 }
00847
00848 /**
00849 * @brief  Checks if target device is ready f
or communication.
00850 * @note   This function is used with Memory
devices
00851 * @param  DevAddress: Target device address
```

```
00852 * @param  Trials: Number of trials
00853 * @retval HAL status
00854 */
00855 static HAL_StatusTypeDef I2Cx_IsDeviceReady(
uint16_t DevAddress, uint32_t Trials)
00856 {
00857   return (HAL_I2C_IsDeviceReady(&heval_I2c,
DevAddress, Trials, I2cxTimeout));
00858 }
00859
00860
00861 /**
00862   * @brief Eval I2Cx error treatment function

00863   * @retval None
00864   */
00865 static void I2Cx_Error (void)
00866 {
00867   /* De-initialize the I2C communication BUS
 */
00868   HAL_I2C_DeInit(&heval_I2c);
00869
00870   /* Re- Initiaize the I2C communication BUS
 */
00871   I2Cx_Init();
00872 }
00873
00874 #endif /* HAL_I2C_MODULE_ENABLED */
00875
00876 /****************************** SPI Routine
s******************************/
00877 #ifdef HAL_SPI_MODULE_ENABLED
00878 /**
00879   * @brief  Initializes SPI MSP.
00880   * @retval None
00881   */
00882 static void SPIx_MspInit(SPI_HandleTypeDef *
```

```
hspi)
00883 {
00884   GPIO_InitTypeDef  gpioinitstruct = {0};
00885
00886   /*** Configure the GPIOs ***/
00887   /* Enable GPIO clock */
00888   EVAL_SPIx_SCK_GPIO_CLK_ENABLE();
00889   EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE();
00890
00891   /* configure SPI SCK */
00892   gpioinitstruct.Pin        = EVAL_SPIx_SCK_
PIN;
00893   gpioinitstruct.Mode       = GPIO_MODE_AF_P
P;
00894   gpioinitstruct.Pull       = GPIO_NOPULL;
00895   gpioinitstruct.Speed      = GPIO_SPEED_FRE
Q_HIGH;
00896   HAL_GPIO_Init(EVAL_SPIx_SCK_GPIO_PORT, &gp
ioinitstruct);
00897
00898   /* configure SPI MISO and MOSI */
00899   gpioinitstruct.Pin        = (EVAL_SPIx_MIS
O_PIN | EVAL_SPIx_MOSI_PIN);
00900   gpioinitstruct.Mode       = GPIO_MODE_AF_P
P;
00901   gpioinitstruct.Pull       = GPIO_NOPULL;
00902   gpioinitstruct.Speed      = GPIO_SPEED_FRE
Q_HIGH;
00903   HAL_GPIO_Init(EVAL_SPIx_MISO_MOSI_GPIO_PORT
, &gpioinitstruct);
00904
00905   /*** Configure the SPI peripheral ***/
00906   /* Enable SPI clock */
00907   EVAL_SPIx_CLK_ENABLE();
00908 }
00909
00910 /**
```

```
00911   * @brief  Initializes SPI HAL.
00912   * @retval None
00913   */
00914 HAL_StatusTypeDef SPIx_Init(void)
00915 {
00916   /* DeInitializes the SPI peripheral */
00917   heval_Spi.Instance = EVAL_SPIx;
00918   HAL_SPI_DeInit(&heval_Spi);
00919
00920   /* SPI Config */
00921   /* SPI baudrate is set to 36 MHz (PCLK2/SP
I_BaudRatePrescaler = 72/2 = 36 MHz) */
00922   heval_Spi.Init.BaudRatePrescaler  = SPI_BA
UDRATEPRESCALER_2;
00923   heval_Spi.Init.Direction          = SPI_DI
RECTION_2LINES;
00924   heval_Spi.Init.CLKPhase           = SPI_PH
ASE_1EDGE;
00925   heval_Spi.Init.CLKPolarity        = SPI_PO
LARITY_LOW;
00926   heval_Spi.Init.CRCCalculation     = SPI_CR
CCALCULATION_DISABLE;
00927   heval_Spi.Init.CRCPolynomial      = 7;
00928   heval_Spi.Init.DataSize           = SPI_DA
TASIZE_8BIT;
00929   heval_Spi.Init.FirstBit           = SPI_FI
RSTBIT_MSB;
00930   heval_Spi.Init.NSS                = SPI_NS
S_SOFT;
00931   heval_Spi.Init.TIMode             = SPI_TI
MODE_DISABLE;
00932   heval_Spi.Init.Mode               = SPI_MO
DE_MASTER;
00933
00934   SPIx_MspInit(&heval_Spi);
00935
00936   return (HAL_SPI_Init(&heval_Spi));
```

```
00937 }
00938
00939
00940 /**
00941   * @brief  SPI Write a byte to device
00942   * @param  WriteValue to be written
00943   * @retval The value of the received byte.
00944   */
00945 static uint8_t SPIx_Write(uint8_t WriteValue
)
00946 {
00947   HAL_StatusTypeDef status = HAL_OK;
00948   uint8_t ReadValue = 0;
00949
00950   status = HAL_SPI_TransmitReceive(&heval_Spi
, (uint8_t*) &WriteValue, (uint8_t*) &ReadValue, 1
, SpixTimeout);
00951
00952   /* Check the communication status */
00953   if(status != HAL_OK)
00954   {
00955     /* Execute user timeout callback */
00956     SPIx_Error();
00957   }
00958
00959    return ReadValue;
00960 }
00961
00962
00963 /**
00964   * @brief SPI Read 1 byte from device
00965   * @retval Read data
00966 */
00967 static uint8_t SPIx_Read(void)
00968 {
00969   return (SPIx_Write(FLASH_SPI_DUMMY_BYTE));
00970 }
```

```
/**
  * @brief SPI error treatment function
  * @retval None
  */
static void SPIx_Error (void)
{
  /* De-initialize the SPI communication BUS */
  HAL_SPI_DeInit(&heval_Spi);

  /* Re- Initiaize the SPI communication BUS */
  SPIx_Init();
}
#endif /* HAL_SPI_MODULE_ENABLED */

/**
  * @}
  */

/** @defgroup STM3210E_EVAL_LinkOperations_Functions Link Operations Functions
  * @{
  */

/*****************************************************************************
                                LINK OPERATIONS
*****************************************************************************/

#if defined(HAL_SRAM_MODULE_ENABLED)
/******************************** LINK LCD ********************************/

```

```
01002 /**
01003   * @brief  Initializes LCD low level.
01004   * @retval None
01005   */
01006 void LCD_IO_Init(void)
01007 {
01008   FSMC_BANK1NORSRAM4_Init();
01009 }
01010
01011 /**
01012   * @brief  Writes data on LCD data register.

01013   * @param  RegValue: Data to be written
01014   * @retval None
01015   */
01016 void  LCD_IO_WriteData(uint16_t RegValue)
01017 {
01018     FSMC_BANK1NORSRAM4_WriteData(RegValue);
01019 }
01020
01021 /**
01022   * @brief  Writes multiple data on LCD data
 register.
01023   * @param  pData: Data to be written
01024   * @param  Size: number of data to write
01025   * @retval None
01026   */
01027 void LCD_IO_WriteMultipleData(uint8_t *pData
, uint32_t Size)
01028 {
01029   uint32_t counter = 0;
01030   uint16_t regvalue;
01031
01032   regvalue = *pData | (*(pData+1) << 8);
01033
01034   for (counter = Size; counter != 0; counter
--)
```

```
01035   {
01036     /* Write 16-bit Reg */
01037     FSMC_BANK1NORSRAM4_WriteData(regvalue);
01038     counter--;
01039     pData += 2;
01040     regvalue = *pData | (*(pData+1) << 8);
01041   }
01042 }
01043
01044 /**
01045   * @brief  Writes register on LCD register.
01046   * @param  Reg: Register to be written
01047   * @retval None
01048   */
01049 void LCD_IO_WriteReg(uint8_t Reg)
01050 {
01051   FSMC_BANK1NORSRAM4_WriteReg(Reg);
01052 }
01053
01054 /**
01055   * @brief  Reads data from LCD data registe
r.
01056   * @param  Reg: Register to be read
01057   * @retval Read data.
01058   */
01059 uint16_t LCD_IO_ReadData(uint16_t Reg)
01060 {
01061   /* Read 16-bit Reg */
01062   return (FSMC_BANK1NORSRAM4_ReadData(Reg));
01063 }
01064
01065 /**
01066   * @brief  Wait for loop in ms.
01067   * @param  Delay in ms.
01068   * @retval None
01069   */
01070 void LCD_Delay (uint32_t Delay)
```

```
01071 {
01072   HAL_Delay(Delay);
01073 }
01074
01075 #endif /*HAL_SRAM_MODULE_ENABLED*/
01076
01077 #ifdef HAL_SPI_MODULE_ENABLED
01078 /******************************** LINK FLASH
 SPI ********************************/
01079 /**
01080   * @brief  Initializes the FLASH SPI and pu
t it into StandBy State (Ready for
01081   *         data transfer).
01082   * @retval None
01083   */
01084 HAL_StatusTypeDef FLASH_SPI_IO_Init(void)
01085 {
01086   HAL_StatusTypeDef Status = HAL_OK;
01087
01088   GPIO_InitTypeDef  gpioinitstruct = {0};
01089
01090   /* EEPROM_CS_GPIO Periph clock enable */
01091   FLASH_SPI_CS_GPIO_CLK_ENABLE();
01092
01093   /* Configure EEPROM_CS_PIN pin: EEPROM SPI
 CS pin */
01094   gpioinitstruct.Pin    = FLASH_SPI_CS_PIN;
01095   gpioinitstruct.Mode   = GPIO_MODE_OUTPUT_P
P;
01096   gpioinitstruct.Pull   = GPIO_PULLUP;
01097   gpioinitstruct.Speed  = GPIO_SPEED_FREQ_HI
GH;
01098   HAL_GPIO_Init(FLASH_SPI_CS_GPIO_PORT, &gpi
oinitstruct);
01099
01100   /* SPI FLASH Config */
01101   Status = SPIx_Init();
```

```
01102
01103   /* EEPROM chip select high */
01104   FLASH_SPI_CS_HIGH();
01105
01106   return Status;
01107 }
01108
01109 /**
01110   * @brief  Write a byte on the FLASH SPI.
01111   * @param  Data: byte to send.
01112   * @retval None
01113   */
01114 uint8_t FLASH_SPI_IO_WriteByte(uint8_t Data)
01115 {
01116   /* Send the byte */
01117   return (SPIx_Write(Data));
01118 }
01119
01120 /**
01121   * @brief  Read a byte from the FLASH SPI.
01122   * @retval uint8_t (The received byte).
01123   */
01124 uint8_t FLASH_SPI_IO_ReadByte(void)
01125 {
01126   uint8_t data = 0;
01127
01128   /* Get the received data */
01129   data = SPIx_Read();
01130
01131   /* Return the shifted data */
01132   return data;
01133 }
01134
01135 /**
01136   * @brief  Read data from FLASH SPI driver
01137   * @param  MemAddress: Internal memory addr
ess
```

```
01138   * @param  pBuffer: Pointer to data buffer
01139   * @param  BufferSize: Amount of data to be
 read
01140   * @retval HAL_StatusTypeDef HAL Status
01141   */
01142 HAL_StatusTypeDef FLASH_SPI_IO_ReadData(uint
32_t MemAddress, uint8_t* pBuffer, uint32_t Buffer
Size)
01143 {
01144   /*!< Select the FLASH: Chip Select low */
01145   FLASH_SPI_CS_LOW();
01146
01147   /*!< Send "Read from Memory " instruction
*/
01148   SPIx_Write(FLASH_SPI_CMD_READ);
01149
01150   /*!< Send ReadAddr high nibble address byt
e to read from */
01151   SPIx_Write((MemAddress & 0xFF0000) >> 16);
01152   /*!< Send ReadAddr medium nibble address b
yte to read from */
01153   SPIx_Write((MemAddress& 0xFF00) >> 8);
01154   /*!< Send ReadAddr low nibble address byte
 to read from */
01155   SPIx_Write(MemAddress & 0xFF);
01156
01157   while (BufferSize--) /*!< while there is d
ata to be read */
01158   {
01159     /*!< Read a byte from the FLASH */
01160     *pBuffer = SPIx_Write(FLASH_SPI_DUMMY_BY
TE);
01161     /*!< Point to the next location where th
e byte read will be saved */
01162     pBuffer++;
01163   }
01164
```

```
01165   /*!< Deselect the FLASH: Chip Select high
*/
01166   FLASH_SPI_CS_HIGH();
01167
01168   return HAL_OK;
01169 }
01170
01171 /**
01172   * @brief  Select the FLASH SPI and send "W
rite Enable" instruction
01173   * @retval None
01174   */
01175 void FLASH_SPI_IO_WriteEnable(void)
01176 {
01177   /*!< Select the FLASH: Chip Select low */
01178   FLASH_SPI_CS_LOW();
01179
01180   /*!< Send "Write Enable" instruction */
01181   SPIx_Write(FLASH_SPI_CMD_WREN);
01182
01183   /*!< Select the FLASH: Chip Select low */
01184   FLASH_SPI_CS_HIGH();
01185
01186     /*!< Select the FLASH: Chip Select low */

01187   FLASH_SPI_CS_LOW();
01188 }
01189
01190 /**
01191   * @brief  Wait response from the FLASH SPI
 and Deselect the device
01192   * @retval HAL_StatusTypeDef HAL Status
01193   */
01194 HAL_StatusTypeDef FLASH_SPI_IO_WaitForWriteE
nd(void)
01195 {
01196   /*!< Select the FLASH: Chip Select low */
```

```
01197   FLASH_SPI_CS_HIGH();
01198
01199   /*!< Select the FLASH: Chip Select low */
01200   FLASH_SPI_CS_LOW();
01201
01202   uint8_t flashstatus = 0;
01203
01204   /*!< Send "Read Status Register" instructi
on */
01205   SPIx_Write(FLASH_SPI_CMD_RDSR);
01206
01207   /*!< Loop as long as the memory is busy wi
th a write cycle */
01208   do
01209   {
01210     /*!< Send a dummy byte to generate the c
lock needed by the FLASH
01211     and put the value of the status register
 in FLASH_Status variable */
01212     flashstatus = SPIx_Write(FLASH_SPI_DUMMY
_BYTE);
01213
01214   }
01215   while ((flashstatus & FLASH_SPI_WIP_FLAG)
== SET); /* Write in progress */
01216
01217   /*!< Deselect the FLASH: Chip Select high
*/
01218   FLASH_SPI_CS_HIGH();
01219
01220   return HAL_OK;
01221 }
01222
01223 /**
01224   * @brief  Reads FLASH SPI identification.
01225   * @retval FLASH identification
01226   */
```

```
01227 uint32_t FLASH_SPI_IO_ReadID(void)
01228 {
01229   uint32_t Temp = 0, Temp0 = 0, Temp1 = 0, T
emp2 = 0;
01230
01231   /*!< Select the FLASH: Chip Select low */
01232   FLASH_SPI_CS_LOW();
01233
01234   /*!< Send "RDID " instruction */
01235   SPIx_Write(0x9F);
01236
01237   /*!< Read a byte from the FLASH */
01238   Temp0 = SPIx_Write(FLASH_SPI_DUMMY_BYTE);
01239
01240   /*!< Read a byte from the FLASH */
01241   Temp1 = SPIx_Write(FLASH_SPI_DUMMY_BYTE);
01242
01243   /*!< Read a byte from the FLASH */
01244   Temp2 = SPIx_Write(FLASH_SPI_DUMMY_BYTE);
01245
01246   /*!< Deselect the FLASH: Chip Select high
*/
01247   FLASH_SPI_CS_HIGH();
01248
01249   Temp = (Temp0 << 16) | (Temp1 << 8) | Temp
2;
01250
01251   return Temp;
01252 }
01253 #endif /* HAL_SPI_MODULE_ENABLED */
01254
01255 #ifdef HAL_I2C_MODULE_ENABLED
01256 /****************************** LINK I2C
TEMPERATURE SENSOR *****************************/
01257 /**
01258   * @brief  Initializes peripherals used by
the I2C Temperature Sensor driver.
```

```
01259   * @retval None
01260   */
01261 void TSENSOR_IO_Init(void)
01262 {
01263   I2Cx_Init();
01264 }
01265
01266 /**
01267   * @brief  Writes one byte to the TSENSOR.
01268   * @param  DevAddress: Target device address

01269   * @param  pBuffer: Pointer to data buffer
01270   * @param  WriteAddr: TSENSOR's internal ad
dress to write to.
01271   * @param  Length: Number of data to write
01272   * @retval None
01273   */
01274 void TSENSOR_IO_Write(uint16_t DevAddress, u
int8_t* pBuffer, uint8_t WriteAddr, uint16_t Lengt
h)
01275 {
01276   I2Cx_WriteBuffer(DevAddress, WriteAddr, I2
C_MEMADD_SIZE_8BIT, pBuffer, Length);
01277 }
01278
01279 /**
01280   * @brief  Reads one byte from the TSENSOR.
01281   * @param  DevAddress: Target device address

01282   * @param  pBuffer : pointer to the buffer
that receives the data read from the TSENSOR.
01283   * @param  ReadAddr : TSENSOR's internal ad
dress to read from.
01284   * @param  Length: Number of data to read
01285   * @retval None
01286   */
01287 void TSENSOR_IO_Read(uint16_t DevAddress, ui
```

```
nt8_t* pBuffer, uint8_t ReadAddr, uint16_t Length)
01288 {
01289   I2Cx_ReadBuffer(DevAddress, ReadAddr, I2C_
MEMADD_SIZE_8BIT, pBuffer, Length);
01290 }
01291
01292 /**
01293 * @brief  Checks if Temperature Sensor is re
ady for communication.
01294   * @param  DevAddress: Target device address

01295   * @param  Trials: Number of trials
01296 * @retval HAL status
01297 */
01298 uint16_t TSENSOR_IO_IsDeviceReady(uint16_t D
evAddress, uint32_t Trials)
01299 {
01300   return (I2Cx_IsDeviceReady(DevAddress, Tri
als));
01301 }
01302
01303 /********************************** LINK AUDI
O ***********************************/
01304
01305 /**
01306   * @brief  Initializes Audio low level.
01307   * @retval None
01308   */
01309 void AUDIO_IO_Init (void)
01310 {
01311   GPIO_InitTypeDef  gpioinitstruct = {0};
01312
01313   /* Enable Reset GPIO Clock */
01314   AUDIO_RESET_GPIO_CLK_ENABLE();
01315
01316   /* Audio reset pin configuration ---------
---------------------------------------*/
```

```
01317   gpioinitstruct.Pin   = AUDIO_RESET_PIN;
01318   gpioinitstruct.Mode  = GPIO_MODE_OUTPUT_PP
;
01319   gpioinitstruct.Speed = GPIO_SPEED_FREQ_MED
IUM;
01320   gpioinitstruct.Pull  = GPIO_NOPULL;
01321   HAL_GPIO_Init(AUDIO_RESET_GPIO, &gpioinits
truct);
01322
01323   I2Cx_Init();
01324
01325   /* Power Down the codec */
01326   HAL_GPIO_WritePin(AUDIO_RESET_GPIO, AUDIO_
RESET_PIN, GPIO_PIN_RESET);
01327
01328   /* wait for a delay to insure registers er
asing */
01329   HAL_Delay(5);
01330
01331   /* Power on the codec */
01332   HAL_GPIO_WritePin(AUDIO_RESET_GPIO, AUDIO_
RESET_PIN, GPIO_PIN_SET);
01333
01334   /* wait for a delay to insure registers er
asing */
01335   HAL_Delay(5);
01336 }
01337
01338 /**
01339   * @brief  Writes a single data.
01340   * @param  Addr: I2C address
01341   * @param  Reg: Reg address
01342   * @param  Value: Data to be written
01343   * @retval None
01344   */
01345 void AUDIO_IO_Write (uint8_t Addr, uint8_t R
eg, uint8_t Value)
```

```
01346 {
01347   I2Cx_WriteData(Addr, Reg, Value);
01348 }
01349
01350 /**
01351   * @brief  Reads a single data.
01352   * @param  Addr: I2C address
01353   * @param  Reg: Reg address
01354   * @retval Data to be read
01355   */
01356 uint8_t AUDIO_IO_Read (uint8_t Addr, uint8_t
 Reg)
01357 {
01358   return I2Cx_ReadData(Addr, Reg);
01359 }
01360
01361 #endif /* HAL_I2C_MODULE_ENABLED */
01362
01363 /**
01364   * @}
01365   */
01366
01367 /**
01368   * @}
01369   */
01370
01371 /**
01372   * @}
01373   */
01374
01375 /**
01376   * @}
01377   */
01378
01379 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
01380
```



Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP
User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

Exported Types

STM3210E-EVAL LCD

Data Structures

struct	**LCD_DrawPropTypeDef**

STM3210E_EVAL BSP User Manual

stm3210e_eval_lcd.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210e_eval_lcd.h
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm3210e_eval_lcd.c driver.
00009   ******************************************************************************
00010   * @attention
00011   *
00012   * <h2><center>&copy; COPYRIGHT(c) 2015 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 /** @addtogroup BSP
00040   * @{
00041   */
00042
00043 /** @addtogroup STM3210E_EVAL
00044   * @{
00045   */
00046
00047 /* Define to prevent recursive inclusion -------------------------------------*/
00048 #ifndef __STM3210E_EVAL_LCD_H
00049 #define __STM3210E_EVAL_LCD_H
00050
00051 #ifdef __cplusplus
00052  extern "C" {
00053 #endif
00054
00055 /* Includes ------------------------------------------------------------------*/
00056 #include "stm3210e_eval.h"
00057 #include "../Components/hx8347d/hx8347d.h"
00058 #include "../Components/spfd5408/spfd5408.h"
00059 #include "../Components/ili9320/ili9320.h"
00060 #include "../../../Utilities/Fonts/fonts.h"
00061
00062 /** @addtogroup STM3210E_EVAL_LCD
00063   * @{
00064   */
00065
00066
00067 /** @defgroup STM3210E_EVAL_LCD_Exported_Types Exported Types
00068   * @{
```

```
00069   */
00070 typedef struct
00071 {
00072   uint32_t TextColor;
00073   uint32_t BackColor;
00074   sFONT    *pFont;
00075
00076 }LCD_DrawPropTypeDef;
00077 /**
00078   * @}
00079   */
00080
00081 /** @defgroup STM3210E_EVAL_LCD_Exported_Con
stants Exported Constants
00082   * @{
00083   */
00084 /**
00085   * @brief  LCD status structure definition

00086   */
00087 #define LCD_OK         0x00
00088 #define LCD_ERROR      0x01
00089 #define LCD_TIMEOUT    0x02
00090
00091 typedef struct
00092 {
00093   int16_t X;
00094   int16_t Y;
00095
00096 }Point, * pPoint;
00097
00098 /**
00099   * @brief  Line mode structures definition
00100   */
00101 typedef enum
00102 {
00103   CENTER_MODE             = 0x01,    /*!< Ce
```

```
nter mode */
00104   RIGHT_MODE                = 0x02,    /*!< Ri
ght mode  */
00105   LEFT_MODE                 = 0x03     /*!< Le
ft mode   */
00106
00107 }Line_ModeTypdef;
00108
00109 /**
00110   * @brief  LCD color
00111   */
00112 #define LCD_COLOR_BLUE          (uint16_t)0x
001F
00113 #define LCD_COLOR_GREEN         (uint16_t)0x
07E0
00114 #define LCD_COLOR_RED           (uint16_t)0x
F800
00115 #define LCD_COLOR_CYAN          (uint16_t)0x
07FF
00116 #define LCD_COLOR_MAGENTA       (uint16_t)0x
F81F
00117 #define LCD_COLOR_YELLOW        (uint16_t)0x
FFE0
00118 #define LCD_COLOR_LIGHTBLUE     (uint16_t)0x
841F
00119 #define LCD_COLOR_LIGHTGREEN    (uint16_t)0x
87F0
00120 #define LCD_COLOR_LIGHTRED      (uint16_t)0x
FC10
00121 #define LCD_COLOR_LIGHTCYAN     (uint16_t)0x
87FF
00122 #define LCD_COLOR_LIGHTMAGENTA  (uint16_t)0x
FC1F
00123 #define LCD_COLOR_LIGHTYELLOW   (uint16_t)0x
FFF0
00124 #define LCD_COLOR_DARKBLUE      (uint16_t)0x
0010
```

```
00125 #define LCD_COLOR_DARKGREEN     (uint16_t)0x
0400
00126 #define LCD_COLOR_DARKRED       (uint16_t)0x
8000
00127 #define LCD_COLOR_DARKCYAN      (uint16_t)0x
0410
00128 #define LCD_COLOR_DARKMAGENTA   (uint16_t)0x
8010
00129 #define LCD_COLOR_DARKYELLOW    (uint16_t)0x
8400
00130 #define LCD_COLOR_WHITE         (uint16_t)0x
FFFF
00131 #define LCD_COLOR_LIGHTGRAY     (uint16_t)0x
D69A
00132 #define LCD_COLOR_GRAY          (uint16_t)0x
8410
00133 #define LCD_COLOR_DARKGRAY      (uint16_t)0x
4208
00134 #define LCD_COLOR_BLACK         (uint16_t)0x
0000
00135 #define LCD_COLOR_BROWN         (uint16_t)0x
A145
00136 #define LCD_COLOR_ORANGE        (uint16_t)0x
FD20
00137
00138 /**
00139   * @brief LCD default font
00140   */
00141 #define LCD_DEFAULT_FONT         Font24
00142
00143 /**
00144   * @}
00145   */
00146
00147 /** @addtogroup STM3210E_EVAL_LCD_Exported_F
unctions
00148   * @{
```

```
00149   */
00150
00151 uint8_t  BSP_LCD_Init(void);
00152 uint32_t BSP_LCD_GetXSize(void);
00153 uint32_t BSP_LCD_GetYSize(void);
00154
00155 uint16_t BSP_LCD_GetTextColor(void);
00156 uint16_t BSP_LCD_GetBackColor(void);
00157 void     BSP_LCD_SetTextColor(__IO uint16_t
Color);
00158 void     BSP_LCD_SetBackColor(__IO uint16_t
Color);
00159 void     BSP_LCD_SetFont(sFONT *fonts);
00160 sFONT    *BSP_LCD_GetFont(void);
00161
00162 void     BSP_LCD_Clear(uint16_t Color);
00163 void     BSP_LCD_ClearStringLine(uint16_t Li
ne);
00164 void     BSP_LCD_DisplayStringAtLine(uint16_
t Line, uint8_t *ptr);
00165 void     BSP_LCD_DisplayStringAt(uint16_t Xp
os, uint16_t Ypos, uint8_t *Text, Line_ModeTypdef
Mode);
00166 void     BSP_LCD_DisplayChar(uint16_t Xpos,
uint16_t Ypos, uint8_t Ascii);
00167
00168 uint16_t BSP_LCD_ReadPixel(uint16_t Xpos, ui
nt16_t Ypos);
00169 void     BSP_LCD_DrawPixel(uint16_t Xpos, ui
nt16_t Ypos, uint16_t RGB_Code);
00170 void     BSP_LCD_DrawHLine(uint16_t Xpos, ui
nt16_t Ypos, uint16_t Length);
00171 void     BSP_LCD_DrawVLine(uint16_t Xpos, ui
nt16_t Ypos, uint16_t Length);
00172 void     BSP_LCD_DrawLine(uint16_t x1, uint1
6_t y1, uint16_t x2, uint16_t y2);
00173 void     BSP_LCD_DrawRect(uint16_t Xpos, uin
```

```
t16_t Ypos, uint16_t Width, uint16_t Height);
00174 void     BSP_LCD_DrawCircle(uint16_t Xpos, u
int16_t Ypos, uint16_t Radius);
00175 void     BSP_LCD_DrawPolygon(pPoint Points,
uint16_t PointCount);
00176 void     BSP_LCD_FillTriangle(uint16_t X1, u
int16_t X2, uint16_t X3, uint16_t Y1, uint16_t Y2,
 uint16_t Y3);
00177 void     BSP_LCD_FillPolygon(pPoint Points,
uint16_t PointCount);
00178 void     BSP_LCD_DrawEllipse(int Xpos, int Y
pos, int XRadius, int YRadius);
00179 void     BSP_LCD_DrawBitmap(uint16_t Xpos, u
int16_t Ypos, uint8_t *pbmp);
00180 void     BSP_LCD_DrawRGBImage(uint16_t Xpos,
 uint16_t Ypos, uint16_t Xsize, uint16_t Ysize, ui
nt8_t *pbmp);
00181 void     BSP_LCD_FillRect(uint16_t Xpos, uin
t16_t Ypos, uint16_t Width, uint16_t Height);
00182 void     BSP_LCD_FillCircle(uint16_t Xpos, u
int16_t Ypos, uint16_t Radius);
00183 void     BSP_LCD_FillEllipse(int Xpos, int Y
pos, int XRadius, int YRadius);
00184
00185 void     BSP_LCD_DisplayOff(void);
00186 void     BSP_LCD_DisplayOn(void);
00187
00188 /**
00189   * @}
00190   */
00191
00192 /**
00193   * @}
00194   */
00195
00196 #ifdef __cplusplus
00197 }
```

```
00198 #endif
00199
00200 #endif /* __STM3210E_EVAL_LCD_H */
00201
00202 /**
00203   * @}
00204   */
00205
00206 /**
00207   * @}
00208   */
00209
00210 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210E_EVAL BSP User Manual

stm3210e_eval_lcd.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210e_eval_lcd.c
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file includes the driver for Liquid Crystal Display (LCD) module
00008   *          mounted on STM3210E-EVAL evaluation board.
00009   @verbatim
00010   ==============================================================================
00011                         ##### How to use this driver #####
00012   ==============================================================================
00013   [..]
00014    (#) This driver is used to drive indirectly an LCD TFT.
00015
```

```
00016    (#) This driver supports the AM-240320L8T
NQW00H (ILI9320), AM-240320LDTNQW-05H (ILI9325)
00017        AM-240320LDTNQW-02H (SPFD5408B) and A
M240320LGTNQW-01H (HX8347D) LCD
00018        mounted on MB895 daughter board
00019
00020    (#) The ILI9320, ILI9325, SPFD5408B and H
X8347D components driver MUST be included with thi
s driver.
00021
00022    (#) Initialization steps:
00023        (++) Initialize the LCD using the BSP
_LCD_Init() function.
00024
00025    (#) Display on LCD
00026        (++) Clear the hole LCD using yhe BSP
_LCD_Clear() function or only one specified
00027             string line using the BSP_LCD_Cl
earStringLine() function.
00028        (++) Display a character on the speci
fied line and column using the BSP_LCD_DisplayChar
()
00029             function or a complete string li
ne using the BSP_LCD_DisplayStringAtLine() functio
n.
00030        (++) Display a string line on the spe
cified position (x,y in pixel) and align mode
00031             using the BSP_LCD_DisplayStringA
tLine() function.
00032        (++) Draw and fill a basic shapes (do
t, line, rectangle, circle, ellipse, .. bitmap, ra
w picture)
00033             on LCD using a set of functions.

00034   @endverbatim
00035   ******************************************
**********************************
```

```
00036   * @attention
00037   *
00038   * <h2><center>&copy; COPYRIGHT(c) 2015 STM
icroelectronics</center></h2>
00039   *
00040   * Redistribution and use in source and bin
ary forms, with or without modification,
00041   * are permitted provided that the followin
g conditions are met:
00042   *   1. Redistributions of source code must
 retain the above copyright notice,
00043   *      this list of conditions and the fol
lowing disclaimer.
00044   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00045   *      this list of conditions and the fol
lowing disclaimer in the documentation
00046   *      and/or other materials provided wit
h the distribution.
00047   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00048   *      may be used to endorse or promote p
roducts derived from this software
00049   *      without specific prior written perm
ission.
00050   *
00051   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00052   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00053   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00054   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00055   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00056   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
```

```
PROCUREMENT OF SUBSTITUTE GOODS OR
00057   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00058   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00059   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00060   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00061   *
00062   ****************************************
**********************************
00063   */
00064
00065 /* Includes ------------------------------
-----------------------------------*/
00066 #include "stm3210e_eval_lcd.h"
00067 #include "../../../Utilities/Fonts/fonts.h"
00068 #include "../../../Utilities/Fonts/font24.c"
00069 #include "../../../Utilities/Fonts/font20.c"
00070 #include "../../../Utilities/Fonts/font16.c"
00071 #include "../../../Utilities/Fonts/font12.c"
00072 #include "../../../Utilities/Fonts/font8.c"
00073
00074 /** @addtogroup BSP
00075   * @{
00076   */
00077
00078 /** @addtogroup STM3210E_EVAL
00079   * @{
00080   */
00081
00082 /** @defgroup STM3210E_EVAL_LCD STM3210E-EVA
L LCD
00083   * @{
00084   */
00085
```

```
00086
00087 /** @defgroup STM3210E_EVAL_LCD_Private_Defi
nes Private Defines
00088   * @{
00089   */
00090 #define POLY_X(Z)              ((int32_t)((
pPoints + (Z))->X))
00091 #define POLY_Y(Z)              ((int32_t)((
pPoints + (Z))->Y))
00092
00093 #define MAX_HEIGHT_FONT         17
00094 #define MAX_WIDTH_FONT          24
00095 #define OFFSET_BITMAP           54
00096 /**
00097   * @}
00098   */
00099
00100 /** @defgroup STM3210E_EVAL_LCD_Private_Macr
os Private Macros
00101   * @{
00102   */
00103 #define ABS(X)  ((X) > 0 ? (X) : -(X))
00104
00105 /**
00106   * @}
00107   */
00108
00109 /** @defgroup STM3210E_EVAL_LCD_Private_Vari
ables Private Variables
00110   * @{
00111   */
00112 LCD_DrawPropTypeDef DrawProp;
00113
00114 static LCD_DrvTypeDef  *lcd_drv;
00115
00116 /* Max size of bitmap will based on a font24
 (17x24) */
```

```
00117 static uint8_t bitmap[MAX_HEIGHT_FONT*MAX_WI
DTH_FONT*2+OFFSET_BITMAP] = {0};
00118
00119 static uint32_t LCD_SwapXY = 0;
00120 /**
00121   * @}
00122   */
00123
00124 /** @defgroup STM3210E_EVAL_LCD_Private_Func
tions Private Functions
00125   * @{
00126   */
00127 static void LCD_DrawPixel(uint16_t Xpos, uin
t16_t Ypos, uint16_t RGBCode);
00128 static void LCD_DrawChar(uint16_t Xpos, uint
16_t Ypos, const uint8_t *pChar);
00129 static void LCD_SetDisplayWindow(uint16_t Xp
os, uint16_t Ypos, uint16_t Width, uint16_t Height
);
00130 /**
00131   * @}
00132   */
00133
00134
00135 /** @defgroup STM3210E_EVAL_LCD_Exported_Fun
ctions Exported Functions
00136   * @{
00137   */
00138
00139 /**
00140   * @brief  Initializes the LCD.
00141   * @retval LCD state
00142   */
00143 uint8_t BSP_LCD_Init(void)
00144 {
00145   uint8_t ret = LCD_ERROR;
00146
```

```
00147   /* Default value for draw propriety */
00148   DrawProp.BackColor = 0xFFFF;
00149   DrawProp.pFont     = &Font24;
00150   DrawProp.TextColor = 0x0000;
00151
00152   if(hx8347d_drv.ReadID() == HX8347D_ID)
00153   {
00154     lcd_drv = &hx8347d_drv;
00155     ret = LCD_OK;
00156   }
00157   else if(spfd5408_drv.ReadID() == SPFD5408_
ID)
00158   {
00159     lcd_drv = &spfd5408_drv;
00160     ret = LCD_OK;
00161   }
00162   else if(ili9320_drv.ReadID() == ILI9320_ID
)
00163   {
00164     lcd_drv = &ili9320_drv;
00165     LCD_SwapXY = 1;
00166     ret = LCD_OK;
00167   }
00168
00169   if(ret != LCD_ERROR)
00170   {
00171     /* LCD Init */
00172     lcd_drv->Init();
00173
00174     /* Initialize the font */
00175     BSP_LCD_SetFont(&LCD_DEFAULT_FONT);
00176   }
00177
00178   #if defined(AFIO_MAPR2_FSMC_NADV_REMAP)
00179     /* Disconnect FSMC pin NADV.
                                    */
00180     /* FSMC and I2C share the same I/O pin:
```

```
NADV/SDA (PB7). Since this pin    */
00181     /* is not used by FSMC on STM32F10E_EVAL
 board, disconnect it internaly   */
00182     /* from FSMC to let it dedicated to I2C1
 without sharing constraints.     */
00183     __HAL_AFIO_FSMCNADV_DISCONNECTED();
00184   #else
00185     #error "Warning: FSMC pin NADV cannot be
 disconnected by internal remap.   \
00186             Therefore, I2C1 will have a shar
ed pin in conflict with FSMC.      \
00187             Use macro __HAL_RCC_FSMC_CLK_ENA
BLE() and                          \
00188             __HAL_RCC_FSMC_CLK_DISABLE() to
manage the FSMC and I2C1 usage     \
00189             alternatively."
00190   #endif
00191
00192   return ret;
00193 }
00194
00195 /**
00196   * @brief  Gets the LCD X size.
00197   * @retval Used LCD X size
00198   */
00199 uint32_t BSP_LCD_GetXSize(void)
00200 {
00201   return(lcd_drv->GetLcdPixelWidth());
00202 }
00203
00204 /**
00205   * @brief  Gets the LCD Y size.
00206   * @retval Used LCD Y size
00207   */
00208 uint32_t BSP_LCD_GetYSize(void)
00209 {
00210   return(lcd_drv->GetLcdPixelHeight());
```

```
00211 }
00212
00213 /**
00214   * @brief  Gets the LCD text color.
00215   * @retval Used text color.
00216   */
00217 uint16_t BSP_LCD_GetTextColor(void)
00218 {
00219   return DrawProp.TextColor;
00220 }
00221
00222 /**
00223   * @brief  Gets the LCD background color.
00224   * @retval Used background color
00225   */
00226 uint16_t BSP_LCD_GetBackColor(void)
00227 {
00228   return DrawProp.BackColor;
00229 }
00230
00231 /**
00232   * @brief  Sets the LCD text color.
00233   * @param  Color: Text color code RGB(5-6-5)

00234   * @retval None
00235   */
00236 void BSP_LCD_SetTextColor(uint16_t Color)
00237 {
00238   DrawProp.TextColor = Color;
00239 }
00240
00241 /**
00242   * @brief  Sets the LCD background color.
00243   * @param  Color: Background color code RGB
(5-6-5)
00244   * @retval None
00245   */
```

```
00246 void BSP_LCD_SetBackColor(uint16_t Color)
00247 {
00248   DrawProp.BackColor = Color;
00249 }
00250
00251 /**
00252   * @brief  Sets the LCD text font.
00253   * @param  pFonts: Font to be used
00254   * @retval None
00255   */
00256 void BSP_LCD_SetFont(sFONT *pFonts)
00257 {
00258   DrawProp.pFont = pFonts;
00259 }
00260
00261 /**
00262   * @brief  Gets the LCD text font.
00263   * @retval Used font
00264   */
00265 sFONT *BSP_LCD_GetFont(void)
00266 {
00267   return DrawProp.pFont;
00268 }
00269
00270 /**
00271   * @brief  Clears the hole LCD.
00272   * @param  Color: Color of the background
00273   * @retval None
00274   */
00275 void BSP_LCD_Clear(uint16_t Color)
00276 {
00277   uint32_t counter = 0;
00278
00279   uint32_t color_backup = DrawProp.TextColor
;
00280   DrawProp.TextColor = Color;
00281
```

```
00282   for(counter = 0; counter < BSP_LCD_GetYSize
(); counter++)
00283   {
00284     BSP_LCD_DrawHLine(0, counter, BSP_LCD_Ge
tXSize());
00285   }
00286
00287   DrawProp.TextColor = color_backup;
00288   BSP_LCD_SetTextColor(DrawProp.TextColor);
00289 }
00290
00291 /**
00292   * @brief  Clears the selected line.
00293   * @param  Line: Line to be cleared
00294   *          This parameter can be one of th
e following values:
00295   *            @arg  0..9: if the Current fo
nts is Font16x24
00296   *            @arg  0..19: if the Current f
onts is Font12x12 or Font8x12
00297   *            @arg  0..29: if the Current f
onts is Font8x8
00298   * @retval None
00299   */
00300 void BSP_LCD_ClearStringLine(uint16_t Line)
00301 {
00302   uint32_t colorbackup = DrawProp.TextColor;

00303   DrawProp.TextColor = DrawProp.BackColor;;
00304
00305   /* Draw a rectangle with background color
*/
00306   BSP_LCD_FillRect(0, (Line * DrawProp.pFont
->Height), BSP_LCD_GetXSize(), DrawProp.pFont->Hei
ght);
00307
00308   DrawProp.TextColor = colorbackup;
```

```
00309   BSP_LCD_SetTextColor(DrawProp.TextColor);
00310 }
00311
00312 /**
00313   * @brief  Displays one character.
00314   * @param  Xpos: Start column address
00315   * @param  Ypos: Line where to display the
character shape.
00316   * @param  Ascii: Character ascii code
00317   *           This parameter must be a numbe
r between Min_Data = 0x20 and Max_Data = 0x7E
00318   * @retval None
00319   */
00320 void BSP_LCD_DisplayChar(uint16_t Xpos, uint
16_t Ypos, uint8_t Ascii)
00321 {
00322   LCD_DrawChar(Ypos, Xpos, &DrawProp.pFont->
table[(Ascii-' ') *\
00323     DrawProp.pFont->Height * ((DrawProp.pFont
->Width + 7) / 8)]);
00324 }
00325
00326 /**
00327   * @brief  Displays characters on the LCD.
00328   * @param  Xpos: X position (in pixel)
00329   * @param  Ypos: Y position (in pixel)
00330   * @param  pText: Pointer to string to disp
lay on LCD
00331   * @param  Mode: Display mode
00332   *          This parameter can be one of th
e following values:
00333   *            @arg  CENTER_MODE
00334   *            @arg  RIGHT_MODE
00335   *            @arg  LEFT_MODE
00336   * @retval None
00337   */
00338 void BSP_LCD_DisplayStringAt(uint16_t Xpos,
```

```
uint16_t Ypos, uint8_t *pText, Line_ModeTypdef Mod
e)
00339 {
00340   uint16_t refcolumn = 1, counter = 0;
00341   uint32_t size = 0, ysize = 0;
00342   uint8_t  *ptr = pText;
00343
00344   /* Get the text size */
00345   while (*ptr++) size ++ ;
00346
00347   /* Characters number per line */
00348   ysize = (BSP_LCD_GetXSize()/DrawProp.pFont
->Width);
00349
00350   switch (Mode)
00351   {
00352   case CENTER_MODE:
00353     {
00354       refcolumn = Xpos + ((ysize - size)* Dr
awProp.pFont->Width) / 2;
00355       break;
00356     }
00357   case LEFT_MODE:
00358     {
00359       refcolumn = Xpos;
00360       break;
00361     }
00362   case RIGHT_MODE:
00363     {
00364       refcolumn = -Xpos + ((ysize - size)*Dr
awProp.pFont->Width);
00365       break;
00366     }
00367   default:
00368     {
00369       refcolumn = Xpos;
00370       break;
```

```
00371     }
00372   }
00373
00374   /* Send the string character by character
on lCD */
00375   while ((*pText != 0) & (((BSP_LCD_GetXSize
() - (counter*DrawProp.pFont->Width)) & 0xFFFF) >=
 DrawProp.pFont->Width))
00376   {
00377     /* Display one character on LCD */
00378     BSP_LCD_DisplayChar(refcolumn, Ypos, *pT
ext);
00379     /* Decrement the column position by 16 */

00380     refcolumn += DrawProp.pFont->Width;
00381     /* Point on the next character */
00382     pText++;
00383     counter++;
00384   }
00385 }
00386
00387 /**
00388   * @brief  Displays a character on the LCD.
00389   * @param  Line: Line where to display the
character shape
00390   *           This parameter can be one of th
e following values:
00391   *             @arg  0..9: if the Current fo
nts is Font16x24
00392   *             @arg  0..19: if the Current f
onts is Font12x12 or Font8x12
00393   *             @arg  0..29: if the Current f
onts is Font8x8
00394   * @param  pText: Pointer to string to disp
lay on LCD
00395   * @retval None
00396   */
```

```
00397 void BSP_LCD_DisplayStringAtLine(uint16_t Li
ne, uint8_t *pText)
00398 {
00399   BSP_LCD_DisplayStringAt(0, LINE(Line),pTex
t, LEFT_MODE);
00400 }
00401
00402 /**
00403   * @brief  Reads an LCD pixel.
00404   * @param  Xpos: X position
00405   * @param  Ypos: Y position
00406   * @retval RGB pixel color
00407   */
00408 uint16_t BSP_LCD_ReadPixel(uint16_t Xpos, ui
nt16_t Ypos)
00409 {
00410   uint16_t ret = 0;
00411
00412   if(lcd_drv->ReadPixel != NULL)
00413   {
00414     ret = lcd_drv->ReadPixel(Xpos, Ypos);
00415   }
00416
00417   return ret;
00418 }
00419
00420 /**
00421   * @brief  Draws an horizontal line.
00422   * @param  Xpos: X position
00423   * @param  Ypos: Y position
00424   * @param  Length: Line length
00425   * @retval None
00426   */
00427 void BSP_LCD_DrawHLine(uint16_t Xpos, uint16
_t Ypos, uint16_t Length)
00428 {
00429   uint32_t index = 0;
```

```
00430
00431   if(lcd_drv->DrawHLine != NULL)
00432   {
00433     if (LCD_SwapXY)
00434     {
00435       uint16_t tmp = Ypos;
00436       Ypos = Xpos;
00437       Xpos =  tmp;
00438     }
00439
00440     lcd_drv->DrawHLine(DrawProp.TextColor, Y
pos, Xpos, Length);
00441   }
00442   else
00443   {
00444     for(index = 0; index < Length; index++)
00445     {
00446       LCD_DrawPixel((Ypos + index), Xpos, Dr
awProp.TextColor);
00447     }
00448   }
00449 }
00450
00451 /**
00452   * @brief  Draws a vertical line.
00453   * @param  Xpos: X position
00454   * @param  Ypos: Y position
00455   * @param  Length: Line length
00456   * @retval None
00457   */
00458 void BSP_LCD_DrawVLine(uint16_t Xpos, uint16
_t Ypos, uint16_t Length)
00459 {
00460   uint32_t index = 0;
00461
00462   if(lcd_drv->DrawVLine != NULL)
00463   {
```

```
00464     if (LCD_SwapXY)
00465     {
00466       uint16_t tmp = Ypos;
00467       Ypos = Xpos;
00468       Xpos =  tmp;
00469     }
00470
00471     LCD_SetDisplayWindow(Ypos, Xpos, 1, Leng
th);
00472     lcd_drv->DrawVLine(DrawProp.TextColor, Y
pos, Xpos, Length);
00473     LCD_SetDisplayWindow(0, 0, BSP_LCD_GetXS
ize(), BSP_LCD_GetYSize());
00474   }
00475   else
00476   {
00477     for(index = 0; index < Length; index++)
00478     {
00479       LCD_DrawPixel(Ypos, Xpos + index, Draw
Prop.TextColor);
00480     }
00481   }
00482 }
00483
00484 /**
00485   * @brief  Draws an uni-line (between two p
oints).
00486   * @param  X1: Point 1 X position
00487   * @param  Y1: Point 1 Y position
00488   * @param  X2: Point 2 X position
00489   * @param  Y2: Point 2 Y position
00490   * @retval None
00491   */
00492 void BSP_LCD_DrawLine(uint16_t X1, uint16_t
Y1, uint16_t X2, uint16_t Y2)
00493 {
00494   int16_t deltax = 0, deltay = 0, x = 0, y =
```

```
 0, xinc1 = 0, xinc2 = 0,
00495   yinc1 = 0, yinc2 = 0, den = 0, num = 0, nu
madd = 0, numpixels = 0,
00496   curpixel = 0;
00497
00498   deltax = ABS(Y2 - Y1);        /* The diffe
rence between the x's */
00499   deltay = ABS(X2 - X1);        /* The diffe
rence between the y's */
00500   x = Y1;                       /* Start x o
ff at the first pixel */
00501   y = X1;                       /* Start y o
ff at the first pixel */
00502
00503   if (Y2 >= Y1)                 /* The x-val
ues are increasing */
00504   {
00505     xinc1 = 1;
00506     xinc2 = 1;
00507   }
00508   else                          /* The x-val
ues are decreasing */
00509   {
00510     xinc1 = -1;
00511     xinc2 = -1;
00512   }
00513
00514   if (X2 >= X1)                 /* The y-val
ues are increasing */
00515   {
00516     yinc1 = 1;
00517     yinc2 = 1;
00518   }
00519   else                          /* The y-val
ues are decreasing */
00520   {
00521     yinc1 = -1;
```

```
00522     yinc2 = -1;
00523   }
00524
00525   if (deltax >= deltay)         /* There is
at least one x-value for every y-value */
00526   {
00527     xinc1 = 0;                  /* Don't cha
nge the x when numerator >= denominator */
00528     yinc2 = 0;                  /* Don't cha
nge the y for every iteration */
00529     den = deltax;
00530     num = deltax / 2;
00531     numadd = deltay;
00532     numpixels = deltax;         /* There are
 more x-values than y-values */
00533   }
00534   else                          /* There is
at least one y-value for every x-value */
00535   {
00536     xinc2 = 0;                  /* Don't cha
nge the x for every iteration */
00537     yinc1 = 0;                  /* Don't cha
nge the y when numerator >= denominator */
00538     den = deltay;
00539     num = deltay / 2;
00540     numadd = deltax;
00541     numpixels = deltay;         /* There are
 more y-values than x-values */
00542   }
00543
00544   for (curpixel = 0; curpixel <= numpixels;
curpixel++)
00545   {
00546     LCD_DrawPixel(x, y, DrawProp.TextColor);
  /* Draw the current pixel */
00547     num += numadd;
  /* Increase the numerator by the top of the frac
```

```
tion */
00548     if (num >= den)
  /* Check if numerator >= denominator */
00549     {
00550       num -= den;
  /* Calculate the new numerator value */
00551       x += xinc1;
  /* Change the x as appropriate */
00552       y += yinc1;
  /* Change the y as appropriate */
00553     }
00554     x += xinc2;
  /* Change the x as appropriate */
00555     y += yinc2;
  /* Change the y as appropriate */
00556   }
00557 }
00558
00559 /**
00560   * @brief  Draws a rectangle.
00561   * @param  Xpos: X position
00562   * @param  Ypos: Y position
00563   * @param  Width: Rectangle width
00564   * @param  Height: Rectangle height
00565   * @retval None
00566   */
00567 void BSP_LCD_DrawRect(uint16_t Xpos, uint16_
t Ypos, uint16_t Width, uint16_t Height)
00568 {
00569   /* Draw horizontal lines */
00570   BSP_LCD_DrawHLine(Xpos, Ypos, Width);
00571   BSP_LCD_DrawHLine(Xpos, (Ypos+ Height), Wi
dth);
00572
00573   /* Draw vertical lines */
00574   BSP_LCD_DrawVLine(Xpos, Ypos, Height);
00575   BSP_LCD_DrawVLine((Xpos + Width), Ypos, He
```

```
ight);
00576 }
00577
00578 /**
00579   * @brief  Draws a circle.
00580   * @param  Xpos: X position
00581   * @param  Ypos: Y position
00582   * @param  Radius: Circle radius
00583   * @retval None
00584   */
00585 void BSP_LCD_DrawCircle(uint16_t Xpos, uint1
6_t Ypos, uint16_t Radius)
00586 {
00587   int32_t  decision;       /* Decision Varia
ble */
00588   uint32_t  curx;   /* Current X Value */
00589   uint32_t  cury;   /* Current Y Value */
00590
00591   decision = 3 - (Radius << 1);
00592   curx = 0;
00593   cury = Radius;
00594
00595   while (curx <= cury)
00596   {
00597     LCD_DrawPixel((Ypos + curx), (Xpos - cur
y), DrawProp.TextColor);
00598
00599     LCD_DrawPixel((Ypos - curx), (Xpos - cur
y), DrawProp.TextColor);
00600
00601     LCD_DrawPixel((Ypos + cury), (Xpos - cur
x), DrawProp.TextColor);
00602
00603     LCD_DrawPixel((Ypos - cury), (Xpos - cur
x), DrawProp.TextColor);
00604
00605     LCD_DrawPixel((Ypos + curx), (Xpos + cur
```

```
y), DrawProp.TextColor);
00606
00607     LCD_DrawPixel((Ypos - curx), (Xpos + cur
y), DrawProp.TextColor);
00608
00609     LCD_DrawPixel((Ypos + cury), (Xpos + cur
x), DrawProp.TextColor);
00610
00611     LCD_DrawPixel((Ypos - cury), (Xpos + cur
x), DrawProp.TextColor);
00612
00613     /* Initialize the font */
00614     BSP_LCD_SetFont(&LCD_DEFAULT_FONT);
00615
00616     if (decision < 0)
00617     {
00618       decision += (curx << 2) + 6;
00619     }
00620     else
00621     {
00622       decision += ((curx - cury) << 2) + 10;
00623       cury--;
00624     }
00625     curx++;
00626   }
00627 }
00628
00629 /**
00630   * @brief  Draws an poly-line (between many
 points).
00631   * @param  pPoints: Pointer to the points a
rray
00632   * @param  PointCount: Number of points
00633   * @retval None
00634   */
00635 void BSP_LCD_DrawPolygon(pPoint pPoints, uin
t16_t PointCount)
```

```
00636 {
00637   int16_t x = 0, y = 0;
00638 
00639   if(PointCount < 2)
00640   {
00641     return;
00642   }
00643 
00644   BSP_LCD_DrawLine(pPoints->X, pPoints->Y, (
pPoints+PointCount-1)->X, (pPoints+PointCount-1)->
Y);
00645 
00646   while(--PointCount)
00647   {
00648     x = pPoints->X;
00649     y = pPoints->Y;
00650     pPoints++;
00651     BSP_LCD_DrawLine(x, y, pPoints->X, pPoin
ts->Y);
00652   }
00653 
00654 }
00655 
00656 /**
00657   * @brief  Draws an ellipse on LCD.
00658   * @param  Xpos: X position
00659   * @param  Ypos: Y position
00660   * @param  XRadius: Ellipse X radius
00661   * @param  YRadius: Ellipse Y radius
00662   * @retval None
00663   */
00664 void BSP_LCD_DrawEllipse(int Xpos, int Ypos,
int XRadius, int YRadius)
00665 {
00666   int x = 0, y = -XRadius, err = 2-2*YRadius
, e2;
00667   float k = 0, rad1 = 0, rad2 = 0;
```

```
00668
00669   rad1 = YRadius;
00670   rad2 = XRadius;
00671
00672   k = (float)(rad2/rad1);
00673
00674   do {
00675     LCD_DrawPixel((Ypos-(uint16_t)(x/k)), (X
pos+y), DrawProp.TextColor);
00676     LCD_DrawPixel((Ypos+(uint16_t)(x/k)), (X
pos+y), DrawProp.TextColor);
00677     LCD_DrawPixel((Ypos+(uint16_t)(x/k)), (X
pos-y), DrawProp.TextColor);
00678     LCD_DrawPixel((Ypos-(uint16_t)(x/k)), (X
pos-y), DrawProp.TextColor);
00679
00680     e2 = err;
00681     if (e2 <= x) {
00682       err += ++x*2+1;
00683       if (-y == x && e2 <= y) e2 = 0;
00684     }
00685     if (e2 > y) err += ++y*2+1;
00686   }
00687   while (y <= 0);
00688 }
00689
00690 /**
00691   * @brief  Draws a bitmap picture loaded in
 the internal Flash (32 bpp).
00692   * @param  Xpos: Bmp X position in the LCD
00693   * @param  Ypos: Bmp Y position in the LCD
00694   * @param  pBmp: Pointer to Bmp picture add
ress in the internal Flash
00695   * @retval None
00696   */
00697 void BSP_LCD_DrawBitmap(uint16_t Xpos, uint1
6_t Ypos, uint8_t *pBmp)
```

```
00698 {
00699   uint32_t height = 0, width  = 0;
00700
00701   if (LCD_SwapXY)
00702   {
00703     uint16_t tmp = Ypos;
00704     Ypos = Xpos;
00705     Xpos =  tmp;
00706   }
00707
00708   /* Read bitmap width */
00709   width = *(uint16_t *) (pBmp + 18);
00710   width |= (*(uint16_t *) (pBmp + 20)) << 16
;
00711
00712   /* Read bitmap height */
00713   height = *(uint16_t *) (pBmp + 22);
00714   height |= (*(uint16_t *) (pBmp + 24)) << 1
6;
00715
00716   /* Remap Ypos, hx8347d works with inverted
 X in case of bitmap */
00717   /* X = 0, cursor is on Bottom corner */
00718   if(lcd_drv == &hx8347d_drv)
00719   {
00720     Ypos = BSP_LCD_GetYSize() - Ypos - heigh
t;
00721   }
00722
00723   LCD_SetDisplayWindow(Ypos, Xpos, width, he
ight);
00724
00725   if(lcd_drv->DrawBitmap != NULL)
00726   {
00727     lcd_drv->DrawBitmap(Ypos, Xpos, pBmp);
00728   }
00729   LCD_SetDisplayWindow(0, 0, BSP_LCD_GetXSize
```

```
(), BSP_LCD_GetYSize());
00730 }
00731
00732 /**
00733   * @brief  Draws a full rectangle.
00734   * @param  Xpos: X position
00735   * @param  Ypos: Y position
00736   * @param  Width: Rectangle width
00737   * @param  Height: Rectangle height
00738   * @retval None
00739   */
00740 void BSP_LCD_FillRect(uint16_t Xpos, uint16_
t Ypos, uint16_t Width, uint16_t Height)
00741 {
00742   BSP_LCD_SetTextColor(DrawProp.TextColor);
00743   do
00744   {
00745     BSP_LCD_DrawHLine(Xpos, Ypos++, Width);

00746   }
00747   while(Height--);
00748 }
00749
00750 /**
00751   * @brief  Draws a full circle.
00752   * @param  Xpos: X position
00753   * @param  Ypos: Y position
00754   * @param  Radius: Circle radius
00755   * @retval None
00756   */
00757 void BSP_LCD_FillCircle(uint16_t Xpos, uint1
6_t Ypos, uint16_t Radius)
00758 {
00759   int32_t  decision;        /* Decision Vari
able */
00760   uint32_t  curx;    /* Current X Value */
00761   uint32_t  cury;    /* Current Y Value */
```

```
00762
00763   decision = 3 - (Radius << 1);
00764
00765   curx = 0;
00766   cury = Radius;
00767
00768   BSP_LCD_SetTextColor(DrawProp.TextColor);
00769
00770   while (curx <= cury)
00771   {
00772     if(cury > 0)
00773     {
00774       BSP_LCD_DrawVLine(Xpos + curx, Ypos -
cury, 2*cury);
00775       BSP_LCD_DrawVLine(Xpos - curx, Ypos -
cury, 2*cury);
00776     }
00777
00778     if(curx > 0)
00779     {
00780       BSP_LCD_DrawVLine(Xpos - cury, Ypos -
curx, 2*curx);
00781       BSP_LCD_DrawVLine(Xpos + cury, Ypos -
curx, 2*curx);
00782     }
00783     if (decision < 0)
00784     {
00785       decision += (curx << 2) + 6;
00786     }
00787     else
00788     {
00789       decision += ((curx - cury) << 2) + 10;
00790       cury--;
00791     }
00792     curx++;
00793   }
00794
```

```
00795   BSP_LCD_SetTextColor(DrawProp.TextColor);
00796   BSP_LCD_DrawCircle(Xpos, Ypos, Radius);
00797 }
00798
00799 /**
00800   * @brief  Fill triangle.
00801   * @param  X1: specifies the point 1 x posi
tion.
00802   * @param  Y1: specifies the point 1 y posi
tion.
00803   * @param  X2: specifies the point 2 x posi
tion.
00804   * @param  Y2: specifies the point 2 y posi
tion.
00805   * @param  X3: specifies the point 3 x posi
tion.
00806   * @param  Y3: specifies the point 3 y posi
tion.
00807   * @retval None
00808   */
00809 void BSP_LCD_FillTriangle(uint16_t X1, uint1
6_t X2, uint16_t X3, uint16_t Y1, uint16_t Y2, uin
t16_t Y3)
00810 {
00811   int16_t deltax = 0, deltay = 0, x = 0, y =
 0, xinc1 = 0, xinc2 = 0,
00812   yinc1 = 0, yinc2 = 0, den = 0, num = 0, nu
madd = 0, numpixels = 0,
00813   curpixel = 0;
00814
00815   deltax = ABS(X2 - X1);        /* The diffe
rence between the x's */
00816   deltay = ABS(Y2 - Y1);        /* The diffe
rence between the y's */
00817   x = X1;                       /* Start x o
ff at the first pixel */
00818   y = Y1;                       /* Start y o
```

```
ff at the first pixel */
00819
00820   if (X2 >= X1)                   /* The x-val
ues are increasing */
00821   {
00822     xinc1 = 1;
00823     xinc2 = 1;
00824   }
00825   else                            /* The x-val
ues are decreasing */
00826   {
00827     xinc1 = -1;
00828     xinc2 = -1;
00829   }
00830
00831   if (Y2 >= Y1)                   /* The y-val
ues are increasing */
00832   {
00833     yinc1 = 1;
00834     yinc2 = 1;
00835   }
00836   else                            /* The y-val
ues are decreasing */
00837   {
00838     yinc1 = -1;
00839     yinc2 = -1;
00840   }
00841
00842   if (deltax >= deltay)           /* There is
at least one x-value for every y-value */
00843   {
00844     xinc1 = 0;                    /* Don't cha
nge the x when numerator >= denominator */
00845     yinc2 = 0;                    /* Don't cha
nge the y for every iteration */
00846     den = deltax;
00847     num = deltax / 2;
```

```
00848     numadd = deltay;
00849     numpixels = deltax;         /* There are
 more x-values than y-values */
00850   }
00851   else                          /* There is
at least one y-value for every x-value */
00852   {
00853     xinc2 = 0;                  /* Don't cha
nge the x for every iteration */
00854     yinc1 = 0;                  /* Don't cha
nge the y when numerator >= denominator */
00855     den = deltay;
00856     num = deltay / 2;
00857     numadd = deltax;
00858     numpixels = deltay;         /* There are
 more y-values than x-values */
00859   }
00860
00861   for (curpixel = 0; curpixel <= numpixels;
curpixel++)
00862   {
00863     BSP_LCD_DrawLine(x, y, X3, Y3);
00864
00865     num += numadd;              /* Increase
the numerator by the top of the fraction */
00866     if (num >= den)             /* Check if
numerator >= denominator */
00867     {
00868       num -= den;               /* Calculate
 the new numerator value */
00869       x += xinc1;               /* Change th
e x as appropriate */
00870       y += yinc1;               /* Change th
e y as appropriate */
00871     }
00872     x += xinc2;                 /* Change th
e x as appropriate */
```

```
00873     y += yinc2;                    /* Change th
e y as appropriate */
00874   }
00875 }
00876
00877 /**
00878   * @brief  Displays a  full poly-line (betw
een many points).
00879   * @param  pPoints: pointer to the points a
rray.
00880   * @param  PointCount: Number of points.
00881   * @retval None
00882   */
00883 void BSP_LCD_FillPolygon(pPoint pPoints, uin
t16_t PointCount)
00884 {
00885
00886   int16_t x = 0, y = 0, x2 = 0, y2 = 0, xcen
ter = 0, ycenter = 0, xfirst = 0, yfirst = 0, pixe
lx = 0, pixely = 0, counter = 0;
00887   uint16_t  imageleft = 0, imageright = 0, i
magetop = 0, imagebottom = 0;
00888
00889   imageleft = imageright = pPoints->X;
00890   imagetop= imagebottom = pPoints->Y;
00891
00892   for(counter = 1; counter < PointCount; cou
nter++)
00893   {
00894     pixelx = POLY_X(counter);
00895     if(pixelx < imageleft)
00896     {
00897       imageleft = pixelx;
00898     }
00899     if(pixelx > imageright)
00900     {
00901       imageright = pixelx;
```

```
00902     }
00903
00904     pixely = POLY_Y(counter);
00905     if(pixely < imagetop)
00906     {
00907       imagetop = pixely;
00908     }
00909     if(pixely > imagebottom)
00910     {
00911       imagebottom = pixely;
00912     }
00913   }
00914
00915   if(PointCount < 2)
00916   {
00917     return;
00918   }
00919
00920   xcenter = (imageleft + imageright)/2;
00921   ycenter = (imagebottom + imagetop)/2;
00922
00923   xfirst = pPoints->X;
00924   yfirst = pPoints->Y;
00925
00926   while(--PointCount)
00927   {
00928     x = pPoints->X;
00929     y = pPoints->Y;
00930     pPoints++;
00931     x2 = pPoints->X;
00932     y2 = pPoints->Y;
00933
00934     BSP_LCD_FillTriangle(x, x2, xcenter, y,
y2, ycenter);
00935     BSP_LCD_FillTriangle(x, xcenter, x2, y,
ycenter, y2);
00936     BSP_LCD_FillTriangle(xcenter, x2, x, yce
```

```
nter, y2, y);
00937   }
00938
00939   BSP_LCD_FillTriangle(xfirst, x2, xcenter,
yfirst, y2, ycenter);
00940   BSP_LCD_FillTriangle(xfirst, xcenter, x2,
yfirst, ycenter, y2);
00941   BSP_LCD_FillTriangle(xcenter, x2, xfirst,
ycenter, y2, yfirst);
00942 }
00943
00944 /**
00945   * @brief  Draws a full ellipse.
00946   * @param  Xpos: X position
00947   * @param  Ypos: Y position
00948   * @param  XRadius: Ellipse X radius
00949   * @param  YRadius: Ellipse Y radius
00950   * @retval None
00951   */
00952 void BSP_LCD_FillEllipse(int Xpos, int Ypos,
int XRadius, int YRadius)
00953 {
00954   int x = 0, y = -XRadius, err = 2-2*YRadius
, e2;
00955   float k = 0, rad1 = 0, rad2 = 0;
00956
00957   rad1 = YRadius;
00958   rad2 = XRadius;
00959
00960   k = (float)(rad2/rad1);
00961
00962   do
00963   {
00964     BSP_LCD_DrawVLine((Xpos+y), (Ypos-(uint1
6_t)(x/k)), (2*(uint16_t)(x/k) + 1));
00965     BSP_LCD_DrawVLine((Xpos-y), (Ypos-(uint1
6_t)(x/k)), (2*(uint16_t)(x/k) + 1));
```

```
00966
00967     e2 = err;
00968     if (e2 <= x)
00969     {
00970       err += ++x*2+1;
00971       if (-y == x && e2 <= y) e2 = 0;
00972     }
00973     if (e2 > y) err += ++y*2+1;
00974   }
00975   while (y <= 0);
00976 }
00977
00978 /**
00979   * @brief  Enables the display.
00980   * @retval None
00981   */
00982 void BSP_LCD_DisplayOn(void)
00983 {
00984   lcd_drv->DisplayOn();
00985 }
00986
00987 /**
00988   * @brief  Disables the display.
00989   * @retval None
00990   */
00991 void BSP_LCD_DisplayOff(void)
00992 {
00993   lcd_drv->DisplayOff();
00994 }
00995
00996 /**
00997   * @}
00998   */
00999
01000 /** @addtogroup STM3210E_EVAL_LCD_Private_Fu
nctions
01001   * @{
```

```
  */

/*****************************************************************************
                                Static Function
*****************************************************************************/
/**
  * @brief  Draws a pixel on LCD.
  * @param  Xpos: X position
  * @param  Ypos: Y position
  * @param  RGBCode: Pixel color in RGB mode (5-6-5)
  * @retval None
  */
static void LCD_DrawPixel(uint16_t Xpos, uint16_t Ypos, uint16_t RGBCode)
{
  if (LCD_SwapXY)
  {
    uint16_t tmp = Ypos;
    Ypos = Xpos;
    Xpos =  tmp;
  }

  if(lcd_drv->WritePixel != NULL)
  {
    lcd_drv->WritePixel(Xpos, Ypos, RGBCode);
  }
}

/**
  * @brief  Draws a character on LCD.
  * @param  Xpos: Line where to display the character shape
  * @param  Ypos: Start column address
```

```
01033   * @param  pChar: Pointer to the character
data
01034   * @retval None
01035   */
01036 static void LCD_DrawChar(uint16_t Xpos, uint
16_t Ypos, const uint8_t *pChar)
01037 {
01038   uint32_t counterh = 0, counterw = 0, index
 = 0;
01039   uint16_t height = 0, width = 0;
01040   uint8_t offset = 0;
01041   uint8_t *pchar = NULL;
01042   uint32_t line = 0;
01043
01044
01045   height = DrawProp.pFont->Height;
01046   width  = DrawProp.pFont->Width;
01047
01048   /* Fill bitmap header*/
01049   *(uint16_t *) (bitmap + 2) = (uint16_t)(he
ight*width*2+OFFSET_BITMAP);
01050   *(uint16_t *) (bitmap + 4) = (uint16_t)((h
eight*width*2+OFFSET_BITMAP)>>16);
01051   *(uint16_t *) (bitmap + 10) = OFFSET_BITMAP
;
01052   *(uint16_t *) (bitmap + 18) = (uint16_t)(w
idth);
01053   *(uint16_t *) (bitmap + 20) = (uint16_t)((
width)>>16);
01054   *(uint16_t *) (bitmap + 22) = (uint16_t)(h
eight);
01055   *(uint16_t *) (bitmap + 24) = (uint16_t)((
height)>>16);
01056
01057   offset =  8 *((width + 7)/8) -  width ;
01058
01059   for(counterh = 0; counterh < height; count
```

```
erh++)
01060   {
01061     pchar = ((uint8_t *)pChar + (width + 7)/
8 * counterh);
01062
01063     if(((width + 7)/8) == 3)
01064     {
01065       line =  (pchar[0]<< 16) | (pchar[1]<<
8) | pchar[2];
01066     }
01067
01068     if(((width + 7)/8) == 2)
01069     {
01070       line =  (pchar[0]<< 8) | pchar[1];
01071     }
01072
01073     if(((width + 7)/8) == 1)
01074     {
01075       line =  pchar[0];
01076     }
01077
01078     for (counterw = 0; counterw < width; cou
nterw++)
01079     {
01080       /* Image in the bitmap is written from
 the bottom to the top */
01081       /* Need to invert image in the bitmap
*/
01082       index = (((height-counterh-1)*width)+(
counterw))*2+OFFSET_BITMAP;
01083       if(line & (1 << (width- counterw + off
set- 1)))
01084       {
01085         bitmap[index] = (uint8_t)DrawProp.Te
xtColor;
01086         bitmap[index+1] = (uint8_t)(DrawProp.
TextColor >> 8);
```

```
01087       }
01088       else
01089       {
01090         bitmap[index] = (uint8_t)DrawProp.Ba
ckColor;
01091         bitmap[index+1] = (uint8_t)(DrawProp.
BackColor >> 8);
01092       }
01093     }
01094   }
01095
01096   BSP_LCD_DrawBitmap(Ypos, Xpos, bitmap);
01097 }
01098
01099 /**
01100   * @brief  Sets display window.
01101   * @param  Xpos: LCD X position
01102   * @param  Ypos: LCD Y position
01103   * @param  Width: LCD window width
01104   * @param  Height: LCD window height
01105   * @retval None
01106   */
01107 static void LCD_SetDisplayWindow(uint16_t Xp
os, uint16_t Ypos, uint16_t Width, uint16_t Height
)
01108 {
01109   if(lcd_drv->SetDisplayWindow != NULL)
01110   {
01111     lcd_drv->SetDisplayWindow(Xpos, Ypos, Wi
dth, Height);
01112   }
01113 }
01114
01115 /**
01116   * @}
01117   */
01118
```

```
01119 /**
01120   * @}
01121   */
01122
01123 /**
01124   * @}
01125   */
01126
01127 /**
01128   * @}
01129   */
01130
01131 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM3210E_EVAL BSP User Manual

Exported Constants

STM3210E-EVAL LCD

Data Structures

struct	**Point**

Defines

#define	**LCD_OK** 0x00 LCD status structure definition.
#define	**LCD_ERROR** 0x01
#define	**LCD_TIMEOUT** 0x02
#define	**LCD_COLOR_BLUE** (uint16_t)0x001F LCD color.
#define	**LCD_COLOR_GREEN** (uint16_t)0x07E0
#define	**LCD_COLOR_RED** (uint16_t)0xF800
#define	**LCD_COLOR_CYAN** (uint16_t)0x07FF
#define	**LCD_COLOR_MAGENTA** (uint16_t)0xF81F
#define	**LCD_COLOR_YELLOW** (uint16_t)0xFFE0
#define	**LCD_COLOR_LIGHTBLUE** (uint16_t)0x841F
#define	**LCD_COLOR_LIGHTGREEN** (uint16_t)0x87F0
#define	**LCD_COLOR_LIGHTRED** (uint16_t)0xFC10
#define	**LCD_COLOR_LIGHTCYAN** (uint16_t)0x87FF
#define	**LCD_COLOR_LIGHTMAGENTA** (uint16_t)0xFC1F
#define	**LCD_COLOR_LIGHTYELLOW** (uint16_t)0xFFF0
#define	**LCD_COLOR_DARKBLUE** (uint16_t)0x0010
#define	**LCD_COLOR_DARKGREEN** (uint16_t)0x0400
#define	**LCD_COLOR_DARKRED** (uint16_t)0x8000
#define	**LCD_COLOR_DARKCYAN** (uint16_t)0x0410
#define	**LCD_COLOR_DARKMAGENTA** (uint16_t)0x8010
#define	**LCD_COLOR_DARKYELLOW** (uint16_t)0x8400
#define	**LCD_COLOR_WHITE** (uint16_t)0xFFFF
#define	**LCD_COLOR_LIGHTGRAY** (uint16_t)0xD69A
#define	**LCD_COLOR_GRAY** (uint16_t)0x8410
#define	**LCD_COLOR_DARKGRAY** (uint16_t)0x4208
#define	**LCD_COLOR_BLACK** (uint16_t)0x0000
#define	**LCD_COLOR_BROWN** (uint16_t)0xA145
#define	**LCD_COLOR_ORANGE** (uint16_t)0xFD20
#define	**LCD_DEFAULT_FONT** Font24

LCD default font.

Typedefs

typedef struct **Point** * **pPoint**

Enumerations

enum	**Line_ModeTypdef** { **CENTER_MODE** = 0x01, **RIGHT_MODE** = 0x02, **LEFT_MODE** = 0x03 } Line mode structures definition. More...

Define Documentation

#define LCD_COLOR_BLACK (uint16_t)0x0000

Definition at line 134 of file stm3210e_eval_lcd.h.

#define LCD_COLOR_BLUE (uint16_t)0x001F

LCD color.

Definition at line 112 of file stm3210e_eval_lcd.h.

#define LCD_COLOR_BROWN (uint16_t)0xA145

Definition at line 135 of file stm3210e_eval_lcd.h.

#define LCD_COLOR_CYAN (uint16_t)0x07FF

Definition at line 115 of file stm3210e_eval_lcd.h.

#define LCD_COLOR_DARKBLUE (uint16_t)0x0010

Definition at line 124 of file stm3210e_eval_lcd.h.

#define LCD_COLOR_DARKCYAN (uint16_t)0x0410

Definition at line 127 of file stm3210e_eval_lcd.h.

#define LCD_COLOR_DARKGRAY (uint16_t)0x4208

Definition at line **133** of file **stm3210e_eval_lcd.h**.

#define LCD_COLOR_DARKGREEN (uint16_t)0x0400

Definition at line **125** of file **stm3210e_eval_lcd.h**.

#define LCD_COLOR_DARKMAGENTA (uint16_t)0x8010

Definition at line **128** of file **stm3210e_eval_lcd.h**.

#define LCD_COLOR_DARKRED (uint16_t)0x8000

Definition at line **126** of file **stm3210e_eval_lcd.h**.

#define LCD_COLOR_DARKYELLOW (uint16_t)0x8400

Definition at line **129** of file **stm3210e_eval_lcd.h**.

#define LCD_COLOR_GRAY (uint16_t)0x8410

Definition at line **132** of file **stm3210e_eval_lcd.h**.

#define LCD_COLOR_GREEN (uint16_t)0x07E0

Definition at line **113** of file **stm3210e_eval_lcd.h**.

#define LCD_COLOR_LIGHTBLUE (uint16_t)0x841F

Definition at line **118** of file **stm3210e_eval_lcd.h**.

#define LCD_COLOR_LIGHTCYAN (uint16_t)0x87FF

Definition at line 121 of file stm3210e_eval_lcd.h.

#define LCD_COLOR_LIGHTGRAY (uint16_t)0xD69A

Definition at line 131 of file stm3210e_eval_lcd.h.

#define LCD_COLOR_LIGHTGREEN (uint16_t)0x87F0

Definition at line 119 of file stm3210e_eval_lcd.h.

#define LCD_COLOR_LIGHTMAGENTA (uint16_t)0xFC1F

Definition at line 122 of file stm3210e_eval_lcd.h.

#define LCD_COLOR_LIGHTRED (uint16_t)0xFC10

Definition at line 120 of file stm3210e_eval_lcd.h.

#define LCD_COLOR_LIGHTYELLOW (uint16_t)0xFFF0

Definition at line 123 of file stm3210e_eval_lcd.h.

#define LCD_COLOR_MAGENTA (uint16_t)0xF81F

Definition at line 116 of file stm3210e_eval_lcd.h.

#define LCD_COLOR_ORANGE (uint16_t)0xFD20

Definition at line 136 of file stm3210e_eval_lcd.h.

#define LCD_COLOR_RED (uint16_t)0xF800

Definition at line 114 of file stm3210e_eval_lcd.h.

#define LCD_COLOR_WHITE (uint16_t)0xFFFF

Definition at line 130 of file stm3210e_eval_lcd.h.

#define LCD_COLOR_YELLOW (uint16_t)0xFFE0

Definition at line 117 of file stm3210e_eval_lcd.h.

#define LCD_DEFAULT_FONT Font24

LCD default font.

Definition at line 141 of file stm3210e_eval_lcd.h.

Referenced by BSP_LCD_DrawCircle(), and BSP_LCD_Init().

#define LCD_ERROR 0x01

Definition at line 88 of file stm3210e_eval_lcd.h.

Referenced by BSP_LCD_Init().

#define LCD_OK 0x00

LCD status structure definition.

Definition at line **87** of file **stm3210e_eval_lcd.h**.

Referenced by **BSP_LCD_Init()**.

#define LCD_TIMEOUT 0x02

Definition at line **89** of file **stm3210e_eval_lcd.h**.

Typedef Documentation

typedef struct Point * pPoint

Enumeration Type Documentation

enum Line_ModeTypdef

Line mode structures definition.

Enumerator:

CENTER_MODE	Center mode
RIGHT_MODE	Right mode
LEFT_MODE	Left mode

Definition at line **101** of file **stm3210e_eval_lcd.h**.

STM3210E_EVAL BSP User Manual

STM3210E_EVAL NAND

STM3210E-EVAL

Modules

- Private_Types_Definitions
- Private_Defines
- Private_Macros
- Private_Variables
- Private_Function_Prototypes
- Exported_Functions
- Exported_Types
- Exported_Constants
- Exported_Macro

STM3210E_EVAL BSP User Manual

STM3210E_EVAL NOR

STM3210E-EVAL

Modules

- Private_Types_Definitions
- Private_Defines
- Private_Macros
- Private_Variables
- Private_Function_Prototypes
- Exported_Functions
- Exported_Types
- Exported_Constants
- Exported_Macro

STM3210E_EVAL BSP User Manual

STM3210E_EVAL Serial FLASH

STM3210E-EVAL

This file includes the M25Pxxx SPI FLASH driver of STM3210E-EVAL board. More...

Modules

- Exported_Functions
- Exported_Types
- Exported_Constants
- Exported_Macros

Detailed Description

This file includes the M25Pxxx SPI FLASH driver of STM3210E-EVAL board.

STM3210E_EVAL BSP User Manual

STM3210E_EVAL SRAM

STM3210E-EVAL

Modules

- Private_Types_Definitions
- Private_Defines
- Private_Macros
- Private_Variables
- Private_Function_Prototypes
- Exported_Functions
- Exported_Types
- Exported_Constants
- Exported_Macro

STM3210E_EVAL BSP User Manual

STM3210E_EVAL_AUDIO

STM3210E-EVAL

This file includes the low layer audio driver available on STM3210E-Eval eval board. More...

Modules

- AUDIO_Private_Types
- AUDIO_Private_Defines
- AUDIO Private_Macros
- AUDIO_Private_Variables
- AUDIO_Private_Function_Prototypes
- AUDIO_OUT_Exported_Functions
- AUDIO_Exported_Types
- AUDIO_OUT_Exported_Constants
- AUDIO_Exported_Macros

Detailed Description

This file includes the low layer audio driver available on STM3210E-Eval eval board.

STM3210E_EVAL BSP User Manual

Defines

Exported_Constants

STM3210E-EVAL SD

Defines

#define	**SD_DETECT_PIN** GPIO_PIN_11
#define	**SD_DETECT_GPIO_PORT** GPIOF
#define	**__SD_DETECT_GPIO_CLK_ENABLE**() __HAL_RCC_GPIO
#define	**SD_DETECT_IRQn** EXTI15_10_IRQn
#define	**SD_DATATIMEOUT** ((uint32_t)100000000)
#define	**SD_PRESENT** ((uint8_t)0x01)
#define	**SD_NOT_PRESENT** ((uint8_t)0x00)
#define	**__DMAx_TxRx_CLK_ENABLE** __HAL_RCC_DMA2_CLK_E
#define	**SD_DMAx_Tx_INSTANCE** DMA2_Channel4
#define	**SD_DMAx_Rx_INSTANCE** DMA2_Channel4
#define	**SD_DMAx_Tx_IRQn** DMA2_Channel4_5_IRQn
#define	**SD_DMAx_Rx_IRQn** DMA2_Channel4_5_IRQn
#define	**SD_DMAx_Tx_IRQHandler** DMA2_Channel4_5_IRQHandle
#define	**SD_DMAx_Rx_IRQHandler** DMA2_Channel4_5_IRQHandle

Define Documentation

#define __DMAx_TxRx_CLK_ENABLE __HAL_RCC_DMA2_CLK_E

Definition at line 99 of file stm3210e_eval_sd.h.

Referenced by SD_MspInit().

#define __SD_DETECT_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPI

Definition at line 90 of file stm3210e_eval_sd.h.

Referenced by SD_MspInit().

#define SD_DATATIMEOUT ((uint32_t)100000000)

Definition at line 93 of file stm3210e_eval_sd.h.

Referenced by BSP_SD_ReadBlocks_DMA(), and BSP_SD_WriteBlocks_DMA().

#define SD_DETECT_GPIO_PORT GPIOF

Definition at line 89 of file stm3210e_eval_sd.h.

Referenced by BSP_SD_IsDetected(), BSP_SD_ITConfig(), and SD_MspInit().

#define SD_DETECT_IRQn EXTI15_10_IRQn

Definition at line 91 of file stm3210e_eval_sd.h.

Referenced by **BSP_SD_ITConfig()**.

#define SD_DETECT_PIN GPIO_PIN_11

Definition at line **88** of file **stm3210e_eval_sd.h**.

Referenced by **BSP_SD_IsDetected()**, **BSP_SD_ITConfig()**, and **SD_MspInit()**.

#define SD_DMAx_Rx_INSTANCE DMA2_Channel4

Definition at line **101** of file **stm3210e_eval_sd.h**.

Referenced by **SD_DMAConfigRx()**.

#define SD_DMAx_Rx_IRQHandler DMA2_Channel4_5_IRQHandle

Definition at line **105** of file **stm3210e_eval_sd.h**.

#define SD_DMAx_Rx_IRQn DMA2_Channel4_5_IRQn

Definition at line **103** of file **stm3210e_eval_sd.h**.

Referenced by **SD_DMAConfigRx()**.

#define SD_DMAx_Tx_INSTANCE DMA2_Channel4

Definition at line **100** of file **stm3210e_eval_sd.h**.

Referenced by **SD_DMAConfigTx()**.

#define SD_DMAx_Tx_IRQHandler DMA2_Channel4_5_IRQHandle

Definition at line 104 of file stm3210e_eval_sd.h.

#define SD_DMAx_Tx_IRQn DMA2_Channel4_5_IRQn

Definition at line 102 of file stm3210e_eval_sd.h.

Referenced by SD_DMAConfigTx().

#define SD_NOT_PRESENT ((uint8_t)0x00)

Definition at line 96 of file stm3210e_eval_sd.h.

Referenced by BSP_SD_IsDetected().

#define SD_PRESENT ((uint8_t)0x01)

Definition at line 95 of file stm3210e_eval_sd.h.

Referenced by BSP_SD_Init(), and BSP_SD_IsDetected().

STM3210E_EVAL BSP User Manual

Exported_Constants

STM3210E_EVAL SRAM

Defines

#define	**SRAM_OK** 0x00 SRAM status structure definition.
#define	**SRAM_ERROR** 0x01
#define	**SRAM_DEVICE_ADDR** ((uint32_t)0x68000000)
#define	**SRAM_DEVICE_SIZE** ((uint32_t)0x200000) /* SRAM device MBytes */
#define	**SRAM_MEMORY_WIDTH** FSMC_NORSRAM_MEM_BUS_W
#define	**SRAM_BURSTACCESS** FSMC_BURST_ACCESS_MODE_D
#define	**SRAM_WRITEBURST** FSMC_WRITE_BURST_DISABLE
#define	**__SRAM_DMAx_CLK_ENABLE** __HAL_RCC_DMA2_CLK_
#define	**SRAM_DMAx_STREAM** DMA2_Channel1
#define	**SRAM_DMAx_IRQn** DMA2_Channel1_IRQn
#define	**SRAM_DMAx_IRQHandler** DMA2_Channel1_IRQHandler

Define Documentation

#define __SRAM_DMAx_CLK_ENABLE __HAL_RCC_DMA2_CLK_

Definition at line **97** of file **stm3210e_eval_sram.h**.

#define SRAM_BURSTACCESS FSMC_BURST_ACCESS_MODE_D

Definition at line **90** of file **stm3210e_eval_sram.h**.

Referenced by **BSP_SRAM_Init()**.

#define SRAM_DEVICE_ADDR ((uint32_t)0x68000000)

Definition at line **84** of file **stm3210e_eval_sram.h**.

#define SRAM_DEVICE_SIZE ((uint32_t)0x200000) /* SRAM device

Definition at line **85** of file **stm3210e_eval_sram.h**.

#define SRAM_DMAx_IRQHandler DMA2_Channel1_IRQHandler

Definition at line **100** of file **stm3210e_eval_sram.h**.

#define SRAM_DMAx_IRQn DMA2_Channel1_IRQn

Definition at line **99** of file **stm3210e_eval_sram.h**.

#define SRAM_DMAx_STREAM DMA2_Channel1

Definition at line **98** of file **stm3210e_eval_sram.h**.

#define SRAM_ERROR 0x01

Definition at line **82** of file **stm3210e_eval_sram.h**.

Referenced by **BSP_SRAM_Init()**, **BSP_SRAM_ReadData()**, **BSP_SRAM_ReadData_DMA()**, **BSP_SRAM_WriteData()**, and **BSP_SRAM_WriteData_DMA()**.

#define SRAM_MEMORY_WIDTH FSMC_NORSRAM_MEM_BUS_W

Definition at line **88** of file **stm3210e_eval_sram.h**.

Referenced by **BSP_SRAM_Init()**.

#define SRAM_OK 0x00

SRAM status structure definition.

Definition at line **81** of file **stm3210e_eval_sram.h**.

Referenced by **BSP_SRAM_Init()**, **BSP_SRAM_ReadData()**, **BSP_SRAM_ReadData_DMA()**, **BSP_SRAM_WriteData()**, and **BSP_SRAM_WriteData_DMA()**.

#define SRAM_WRITEBURST FSMC_WRITE_BURST_DISABLE

Definition at line **93** of file **stm3210e_eval_sram.h**.

Referenced by **BSP_SRAM_Init()**.

User Manual by  1.7.6.1

STM3210E_EVAL BSP User Manual

Private Defines

STM3210E-EVAL Common

Defines

#define	**__STM3210E_EVAL_BSP_VERSION_MAIN** (0x06) STM3210E EVAL BSP Driver version number V6.0.0.
#define	**__STM3210E_EVAL_BSP_VERSION_SUB1** (0x00)
#define	**__STM3210E_EVAL_BSP_VERSION_SUB2** (0x01)
#define	**__STM3210E_EVAL_BSP_VERSION_RC** (0x00)
#define	**__STM3210E_EVAL_BSP_VERSION**
#define	**TFT_LCD_BASE** FSMC_BANK1_4
#define	**TFT_LCD** ((**LCD_CONTROLLER_TypeDef** *) **TFT_LCD_BASE**)

Define Documentation

#define __STM3210E_EVAL_BSP_VERSION

Value:

```
((__STM3210E_EVAL_BSP_VERSION_MAIN << 24)\
                                                  |(
__STM3210E_EVAL_BSP_VERSION_SUB1 << 16)\
                                                  |(
__STM3210E_EVAL_BSP_VERSION_SUB2 << 8 )\
                                                  |(
__STM3210E_EVAL_BSP_VERSION_RC))
```

Definition at line **80** of file **stm3210e_eval.c**.

Referenced by **BSP_GetVersion()**.

#define __STM3210E_EVAL_BSP_VERSION_MAIN (0x06)

STM3210E EVAL BSP Driver version number V6.0.0.

[31:24] main version

Definition at line **76** of file **stm3210e_eval.c**.

#define __STM3210E_EVAL_BSP_VERSION_RC (0x00)

[7:0] release candidate

Definition at line **79** of file **stm3210e_eval.c**.

#define __STM3210E_EVAL_BSP_VERSION_SUB1 (0x00)

[23:16] sub1 version

Definition at line **77** of file **stm3210e_eval.c**.

#define __STM3210E_EVAL_BSP_VERSION_SUB2 (0x01)

[15:8] sub2 version

Definition at line **78** of file **stm3210e_eval.c**.

#define TFT_LCD ((LCD_CONTROLLER_TypeDef *) TFT_LCD_BA

Definition at line **88** of file **stm3210e_eval.c**.

Referenced by **FSMC_BANK1NORSRAM4_ReadData()**, **FSMC_BANK1NORSRAM4_WriteData()**, and **FSMC_BANK1NORSRAM4_WriteReg()**.

#define TFT_LCD_BASE FSMC_BANK1_4

Definition at line **87** of file **stm3210e_eval.c**.

STM3210E_EVAL BSP User Manual

Private Macros

STM3210E-EVAL LCD

Defines

```
#define ABS(X)  ((X) > 0 ? (X) : -(X))
```

Define Documentation

#define ABS (X) ((X) > 0 ? (X) : -(X))

Definition at line **103** of file **stm3210e_eval_lcd.c**.

Referenced by **BSP_LCD_DrawLine()**, and **BSP_LCD_FillTriangle()**.

STM3210E_EVAL BSP User Manual

AUDIO_OUT_Exported_Constants

STM3210E_EVAL_AUDIO

Defines

#define	**I2SOUT** SPI2
#define	**I2SOUT_CLK_ENABLE**() __HAL_RCC_SPI2_CLK_ENABLE
#define	**I2SOUT_SCK_SD_CLK_ENABLE**() __HAL_RCC_GPIOB_C
#define	**I2SOUT_MCK_CLK_ENABLE**() __HAL_RCC_GPIOC_CLK_
#define	**I2SOUT_WS_CLK_ENABLE**() __HAL_RCC_GPIOB_CLK_E
#define	**I2SOUT_WS_PIN** GPIO_PIN_12 /* PB.12*/
#define	**I2SOUT_SCK_PIN** GPIO_PIN_13 /* PB.13*/
#define	**I2SOUT_SD_PIN** GPIO_PIN_15 /* PB.15*/
#define	**I2SOUT_MCK_PIN** GPIO_PIN_6 /* PC.06*/
#define	**I2SOUT_SCK_SD_GPIO_PORT** GPIOB
#define	**I2SOUT_WS_GPIO_PORT** GPIOB
#define	**I2SOUT_MCK_GPIO_PORT** GPIOC
#define	**I2SOUT_DMAx_CLK_ENABLE**() __HAL_RCC_DMA1_CLK_
#define	**I2SOUT_DMAx_CHANNEL** DMA1_Channel5
#define	**I2SOUT_DMAx_IRQ** DMA1_Channel5_IRQn
#define	**I2SOUT_DMAx_PERIPH_DATA_SIZE** DMA_PDATAALIGN_
#define	**I2SOUT_DMAx_MEM_DATA_SIZE** DMA_MDATAALIGN_HA
#define	**DMA_MAX_SZE** 0xFFFF
#define	**I2SOUT_IRQHandler** DMA1_Channel5_IRQHandler
#define	**AUDIO_OUT_IRQ_PREPRIO** 5 /* Select the preemption prio the highest) */
#define	**AUDIO_OK** 0
#define	**AUDIO_ERROR** 1
#define	**AUDIO_TIMEOUT** 2

Define Documentation

#define AUDIO_ERROR 1

Definition at line 115 of file stm3210e_eval_audio.h.

Referenced by BSP_AUDIO_OUT_Init(), BSP_AUDIO_OUT_Pause(), BSP_AUDIO_OUT_Play(), BSP_AUDIO_OUT_Resume(), BSP_AUDIO_OUT_SetMute(), BSP_AUDIO_OUT_SetOutputMode(), BSP_AUDIO_OUT_SetVolume(), and BSP_AUDIO_OUT_Stop().

#define AUDIO_OK 0

Definition at line 114 of file stm3210e_eval_audio.h.

Referenced by BSP_AUDIO_OUT_Init(), BSP_AUDIO_OUT_Pause(), BSP_AUDIO_OUT_Play(), BSP_AUDIO_OUT_Resume(), BSP_AUDIO_OUT_SetMute(), BSP_AUDIO_OUT_SetOutputMode(), BSP_AUDIO_OUT_SetVolume(), BSP_AUDIO_OUT_Stop(), I2SOUT_SendDummyData_Start(), and I2SOUT_SendDummyData_Stop().

#define AUDIO_OUT_IRQ_PREPRIO 5 /* Select the preemption pri

Definition at line 107 of file stm3210e_eval_audio.h.

Referenced by I2SOUT_MspInit().

#define AUDIO_TIMEOUT 2

Definition at line 116 of file stm3210e_eval_audio.h.

#define DMA_MAX_SZE 0xFFFF

Definition at line 102 of file stm3210e_eval_audio.h.

#define I2SOUT SPI2

Definition at line 83 of file stm3210e_eval_audio.h.

Referenced by HAL_I2S_ErrorCallback(), HAL_I2S_TxCpltCallback(), HAL_I2S_TxHalfCpltCallback(), I2SOUT_Init(), and I2SOUT_MspInit().

#define I2SOUT_CLK_ENABLE ( ) __HAL_RCC_SPI2_CLK_ENAB

Definition at line 84 of file stm3210e_eval_audio.h.

Referenced by I2SOUT_MspInit().

#define I2SOUT_DMAx_CHANNEL DMA1_Channel5

Definition at line 98 of file stm3210e_eval_audio.h.

Referenced by I2SOUT_MspInit().

#define I2SOUT_DMAx_CLK_ENABLE ( ) __HAL_RCC_DMA1_CL

Definition at line 97 of file stm3210e_eval_audio.h.

Referenced by I2SOUT_MspInit().

#define I2SOUT_DMAx_IRQ DMA1_Channel5_IRQn

Definition at line **99** of file **stm3210e_eval_audio.h**.

Referenced by **I2SOUT_MspInit()**.

#define I2SOUT_DMAx_MEM_DATA_SIZE DMA_MDATAALIGN_HA

Definition at line **101** of file **stm3210e_eval_audio.h**.

Referenced by **I2SOUT_MspInit()**.

#define I2SOUT_DMAx_PERIPH_DATA_SIZE DMA_PDATAALIGN_

Definition at line **100** of file **stm3210e_eval_audio.h**.

Referenced by **I2SOUT_MspInit()**.

#define I2SOUT_IRQHandler DMA1_Channel5_IRQHandler

Definition at line **104** of file **stm3210e_eval_audio.h**.

#define I2SOUT_MCK_CLK_ENABLE ( ) __HAL_RCC_GPIOC_CLI

Definition at line **86** of file **stm3210e_eval_audio.h**.

Referenced by **I2SOUT_MspInit()**.

#define I2SOUT_MCK_GPIO_PORT GPIOC

Definition at line **94** of file **stm3210e_eval_audio.h**.

Referenced by **I2SOUT_MspInit()**.

#define I2SOUT_MCK_PIN GPIO_PIN_6 /* PC.06*/

Definition at line 91 of file stm3210e_eval_audio.h.

Referenced by I2SOUT_MspInit().

#define I2SOUT_SCK_PIN GPIO_PIN_13 /* PB.13*/

Definition at line 89 of file stm3210e_eval_audio.h.

Referenced by I2SOUT_MspInit().

#define I2SOUT_SCK_SD_CLK_ENABLE ( ) __HAL_RCC_GPIOB_

Definition at line 85 of file stm3210e_eval_audio.h.

Referenced by I2SOUT_MspInit().

#define I2SOUT_SCK_SD_GPIO_PORT GPIOB

Definition at line 92 of file stm3210e_eval_audio.h.

Referenced by I2SOUT_MspInit().

#define I2SOUT_SD_PIN GPIO_PIN_15 /* PB.15*/

Definition at line 90 of file stm3210e_eval_audio.h.

Referenced by I2SOUT_MspInit().

#define I2SOUT_WS_CLK_ENABLE ( ) __HAL_RCC_GPIOB_CLK_

Definition at line **87** of file **stm3210e_eval_audio.h**.

Referenced by **I2SOUT_MspInit()**.

#define I2SOUT_WS_GPIO_PORT GPIOB

Definition at line **93** of file **stm3210e_eval_audio.h**.

Referenced by **I2SOUT_MspInit()**.

#define I2SOUT_WS_PIN GPIO_PIN_12 /* PB.12*/

Definition at line **88** of file **stm3210e_eval_audio.h**.

Referenced by **I2SOUT_MspInit()**.

STM3210E_EVAL BSP User Manual

STM3210E_EVAL_COMPONENT

Exported Constants

Defines

#define	**FLASH_SPI_CS_LOW**() HAL_GPIO_WritePin(**FLASH_SPI_** **FLASH_SPI_CS_PIN**, GPIO_PIN_RESET) M25P FLASH SPI Chip Select macro definition.
#define	**FLASH_SPI_CS_HIGH**() HAL_GPIO_WritePin(**FLASH_SPI_** **FLASH_SPI_CS_PIN**, GPIO_PIN_SET)
#define	**FLASH_SPI_CS_PIN** GPIO_PIN_2 /* PB.02*/ M25P FLASH SPI Control Interface pins.
#define	**FLASH_SPI_CS_GPIO_PORT** GPIOB
#define	**FLASH_SPI_CS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIO
#define	**FLASH_SPI_CS_GPIO_CLK_DISABLE**() __HAL_RCC_GPI
#define	**FLASH_SPI_CMD_WRITE** 0x02 M25P FLASH SPI supported commands.
#define	**FLASH_SPI_CMD_WRSR** 0x01
#define	**FLASH_SPI_CMD_WREN** 0x06
#define	**FLASH_SPI_CMD_READ** 0x03
#define	**FLASH_SPI_CMD_RDSR** 0x05
#define	**FLASH_SPI_CMD_RDID** 0x9F
#define	**FLASH_SPI_CMD_SE** 0xD8
#define	**FLASH_SPI_CMD_BE** 0xC7
#define	**FLASH_SPI_WIP_FLAG** 0x01
#define	**FLASH_SPI_DUMMY_BYTE** 0xA5
#define	**FLASH_SPI_PAGESIZE** 0x100
#define	**FLASH_SPI_M25P128_ID** 0x202018
#define	**FLASH_SPI_M25P64_ID** 0x202017
#define	**AUDIO_I2C_ADDRESS** 0x27 AUDIO I2C Interface pins.
#define	**AUDIO_RESET_GPIO_CLK_ENABLE**() __HAL_RCC_GPIO
#define	**AUDIO_RESET_PIN** GPIO_PIN_11
#define	**AUDIO_RESET_GPIO** GPIOG

Define Documentation

#define AUDIO_I2C_ADDRESS 0x27

AUDIO I2C Interface pins.

Definition at line 485 of file stm3210e_eval.h.

Referenced by BSP_AUDIO_OUT_Init(), BSP_AUDIO_OUT_Pause(), BSP_AUDIO_OUT_Play(), BSP_AUDIO_OUT_Resume(), BSP_AUDIO_OUT_SetMute(), BSP_AUDIO_OUT_SetOutputMode(), BSP_AUDIO_OUT_SetVolume(), and BSP_AUDIO_OUT_Stop().

#define AUDIO_RESET_GPIO GPIOG

Definition at line 490 of file stm3210e_eval.h.

Referenced by AUDIO_IO_Init(), and BSP_AUDIO_OUT_Stop().

#define AUDIO_RESET_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPI

Definition at line 488 of file stm3210e_eval.h.

Referenced by AUDIO_IO_Init().

#define AUDIO_RESET_PIN GPIO_PIN_11

Definition at line 489 of file stm3210e_eval.h.

Referenced by AUDIO_IO_Init(), and BSP_AUDIO_OUT_Stop().

#define FLASH_SPI_CMD_BE 0xC7

Bulk Erase instruction

Definition at line 470 of file stm3210e_eval.h.

Referenced by BSP_SERIAL_FLASH_EraseBulk().

#define FLASH_SPI_CMD_RDID 0x9F

Read identification

Definition at line 468 of file stm3210e_eval.h.

#define FLASH_SPI_CMD_RDSR 0x05

Read Status Register instruction

Definition at line 467 of file stm3210e_eval.h.

Referenced by FLASH_SPI_IO_WaitForWriteEnd().

#define FLASH_SPI_CMD_READ 0x03

Read from Memory instruction

Definition at line 466 of file stm3210e_eval.h.

Referenced by FLASH_SPI_IO_ReadData().

#define FLASH_SPI_CMD_SE 0xD8

Sector Erase instruction

Definition at line **469** of file **stm3210e_eval.h**.

Referenced by **BSP_SERIAL_FLASH_EraseSector()**.

#define FLASH_SPI_CMD_WREN 0x06

Write enable instruction

Definition at line **465** of file **stm3210e_eval.h**.

Referenced by **FLASH_SPI_IO_WriteEnable()**.

#define FLASH_SPI_CMD_WRITE 0x02

M25P FLASH SPI supported commands.

Write to Memory instruction

Definition at line **463** of file **stm3210e_eval.h**.

Referenced by **BSP_SERIAL_FLASH_WritePage()**.

#define FLASH_SPI_CMD_WRSR 0x01

Write Status Register instruction

Definition at line **464** of file **stm3210e_eval.h**.

#define FLASH_SPI_CS_GPIO_CLK_DISABLE ( ) __HAL_RCC_GI

Definition at line **458** of file **stm3210e_eval.h**.

#define FLASH_SPI_CS_GPIO_CLK_ENABLE ( ) __HAL_RCC_GP

Definition at line **457** of file **stm3210e_eval.h**.

Referenced by **FLASH_SPI_IO_Init()**.

#define FLASH_SPI_CS_GPIO_PORT GPIOB

Definition at line **456** of file **stm3210e_eval.h**.

Referenced by **FLASH_SPI_IO_Init()**.

#define FLASH_SPI_CS_HIGH ( ) HAL_GPIO_WritePin(FLASH_SP

Definition at line **450** of file **stm3210e_eval.h**.

Referenced by **FLASH_SPI_IO_Init()**, **FLASH_SPI_IO_ReadData()**, **FLASH_SPI_IO_ReadID()**, **FLASH_SPI_IO_WaitForWriteEnd()**, and **FLASH_SPI_IO_WriteEnable()**.

#define FLASH_SPI_CS_LOW ( ) HAL_GPIO_WritePin(FLASH_SP

M25P FLASH SPI Chip Select macro definition.

Definition at line **449** of file **stm3210e_eval.h**.

Referenced by **FLASH_SPI_IO_ReadData()**, **FLASH_SPI_IO_ReadID()**, **FLASH_SPI_IO_WaitForWriteEnd()**, and **FLASH_SPI_IO_WriteEnable()**.

#define FLASH_SPI_CS_PIN GPIO_PIN_2 /* PB.02*/

M25P FLASH SPI Control Interface pins.

Definition at line **455** of file **stm3210e_eval.h**.

Referenced by **FLASH_SPI_IO_Init()**.

#define FLASH_SPI_DUMMY_BYTE 0xA5

Definition at line **474** of file **stm3210e_eval.h**.

Referenced by **FLASH_SPI_IO_ReadData()**, **FLASH_SPI_IO_ReadID()**, **FLASH_SPI_IO_WaitForWriteEnd()**, and **SPIx_Read()**.

#define FLASH_SPI_M25P128_ID 0x202018

Definition at line **477** of file **stm3210e_eval.h**.

#define FLASH_SPI_M25P64_ID 0x202017

Definition at line **478** of file **stm3210e_eval.h**.

#define FLASH_SPI_PAGESIZE 0x100

Definition at line **475** of file **stm3210e_eval.h**.

Referenced by **BSP_SERIAL_FLASH_WriteData()**.

#define FLASH_SPI_WIP_FLAG 0x01

Write In Progress (WIP) flag

Definition at line **472** of file **stm3210e_eval.h**.

Referenced by **FLASH_SPI_IO_WaitForWriteEnd()**.

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP
User Manual by  1.7.6.1

STM3210E_EVAL BSP User Manual

Link Operations Functions

STM3210E-EVAL Common

Functions

void	**LCD_IO_Init** (void) Initializes LCD low level.
void	**LCD_IO_WriteData** (uint16_t RegValue) Writes data on LCD data register.
void	**LCD_IO_WriteMultipleData** (uint8_t *pData, uint32_t Size) Writes multiple data on LCD data register.
void	**LCD_IO_WriteReg** (uint8_t Reg) Writes register on LCD register.
uint16_t	**LCD_IO_ReadData** (uint16_t Reg) Reads data from LCD data register.
void	**LCD_Delay** (uint32_t Delay) Wait for loop in ms.
HAL_StatusTypeDef	**FLASH_SPI_IO_Init** (void) Initializes the FLASH SPI and put it into StandBy State (Ready for data transfer).
uint8_t	**FLASH_SPI_IO_WriteByte** (uint8_t Data) Write a byte on the FLASH SPI.
uint8_t	**FLASH_SPI_IO_ReadByte** (void) Read a byte from the FLASH SPI.
HAL_StatusTypeDef	**FLASH_SPI_IO_ReadData** (uint32_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Read data from FLASH SPI driver.
void	**FLASH_SPI_IO_WriteEnable** (void) Select the FLASH SPI and send "Write Enable" instruction.
HAL_StatusTypeDef	**FLASH_SPI_IO_WaitForWriteEnd** (void) Wait response from the FLASH SPI and Deselect the device.
uint32_t	**FLASH_SPI_IO_ReadID** (void)

	Reads FLASH SPI identification.
void	**TSENSOR_IO_Init** (void) Initializes peripherals used by the I2C Temperature Sensor driver.
void	**TSENSOR_IO_Write** (uint16_t DevAddress, uint8_t *pBuffer, uint8_t WriteAddr, uint16_t Length) Writes one byte to the TSENSOR.
void	**TSENSOR_IO_Read** (uint16_t DevAddress, uint8_t *pBuffer, uint8_t ReadAddr, uint16_t Length) Reads one byte from the TSENSOR.
uint16_t	**TSENSOR_IO_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if Temperature Sensor is ready for communication.
void	**AUDIO_IO_Init** (void) Initializes Audio low level.
void	**AUDIO_IO_Write** (uint8_t Addr, uint8_t Reg, uint8_t Value) Writes a single data.
uint8_t	**AUDIO_IO_Read** (uint8_t Addr, uint8_t Reg) Reads a single data.

Function Documentation

void AUDIO_IO_Init (void)

Initializes Audio low level.

Return values:
None

Definition at line **1309** of file **stm3210e_eval.c**.

References **AUDIO_RESET_GPIO**, **AUDIO_RESET_GPIO_CLK_ENABLE**, **AUDIO_RESET_PIN**, and **I2Cx_Init()**.

uint8_t AUDIO_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 **1356** of file **stm3210e_eval.c**.

References **I2Cx_ReadData()**.

void AUDIO_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 **1345** of file **stm3210e_eval.c**.

References **I2Cx_WriteData()**.

HAL_StatusTypeDef FLASH_SPI_IO_Init (void)

Initializes the FLASH SPI and put it into StandBy State (Ready for data transfer).

Return values:

None

Definition at line **1084** of file **stm3210e_eval.c**.

References **FLASH_SPI_CS_GPIO_CLK_ENABLE**, **FLASH_SPI_CS_GPIO_PORT**, **FLASH_SPI_CS_HIGH**, **FLASH_SPI_CS_PIN**, and **SPIx_Init()**.

Referenced by **BSP_SERIAL_FLASH_Init()**.

uint8_t FLASH_SPI_IO_ReadByte (void)

Read a byte from the FLASH SPI.

Return values:

uint8_t (The received byte).

Definition at line **1124** of file **stm3210e_eval.c**.

References **SPIx_Read()**.

HAL_StatusTypeDef FLASH_SPI_IO_ReadData (uint32_t MemAddr
uint8_t * pBuffer,
uint32_t BufferSiz
)

Read data from FLASH SPI driver.

Parameters:

MemAddress,: Internal memory address

pBuffer,: Pointer to data buffer

BufferSize,: Amount of data to be read

Return values:

HAL_StatusTypeDef HAL Status

< Select the FLASH: Chip Select low

< Send "Read from Memory " instruction

< Send ReadAddr high nibble address byte to read from

< Send ReadAddr medium nibble address byte to read from

< Send ReadAddr low nibble address byte to read from

< while there is data to be read

< Read a byte from the FLASH

< **Point** to the next location where the byte read will be saved

Definition at line **1142** of file **stm3210e_eval.c**.

References **FLASH_SPI_CMD_READ**, **FLASH_SPI_CS_HIGH**, **FLASH_SPI_CS_LOW**, **FLASH_SPI_DUMMY_BYTE**, and **SPIx_Write()**.

Referenced by **BSP_SERIAL_FLASH_ReadData()**.

uint32_t FLASH_SPI_IO_ReadID (void)

Reads FLASH SPI identification.

Return values:

FLASH identification

< Select the FLASH: Chip Select low

< Send "RDID " instruction

< Read a byte from the FLASH

< Read a byte from the FLASH

< Read a byte from the FLASH

< Deselect the FLASH: Chip Select high

Definition at line **1227** of file **stm3210e_eval.c**.

References **FLASH_SPI_CS_HIGH**, **FLASH_SPI_CS_LOW**, **FLASH_SPI_DUMMY_BYTE**, and **SPIx_Write()**.

Referenced by **BSP_SERIAL_FLASH_ReadID()**.

HAL_StatusTypeDef FLASH_SPI_IO_WaitForWriteEnd (void)

Wait response from the FLASH SPI and Deselect the device.

Return values:

HAL_StatusTypeDef HAL Status

< Select the FLASH: Chip Select low

< Select the FLASH: Chip Select low

< Send "Read Status Register" instruction

< Loop as long as the memory is busy with a write cycle

< Send a dummy byte to generate the clock needed by the FLASH and put the value of the status register in FLASH_Status variable

< Deselect the FLASH: Chip Select high

Definition at line **1194** of file **stm3210e_eval.c**.

References **FLASH_SPI_CMD_RDSR**, **FLASH_SPI_CS_HIGH**, **FLASH_SPI_CS_LOW**, **FLASH_SPI_DUMMY_BYTE**, **FLASH_SPI_WIP_FLAG**, and **SPIx_Write()**.

Referenced by **BSP_SERIAL_FLASH_EraseBulk()**, **BSP_SERIAL_FLASH_EraseSector()**, and **BSP_SERIAL_FLASH_WritePage()**.

uint8_t FLASH_SPI_IO_WriteByte (uint8_t Data)

Write a byte on the FLASH SPI.

Parameters:

Data,: byte to send.

Return values:

None

Definition at line **1114** of file **stm3210e_eval.c**.

References **SPIx_Write()**.

Referenced by **BSP_SERIAL_FLASH_EraseBulk()**, **BSP_SERIAL_FLASH_EraseSector()**, and **BSP_SERIAL_FLASH_WritePage()**.

void FLASH_SPI_IO_WriteEnable (void)

Select the FLASH SPI and send "Write Enable" instruction.

Return values:

None

< Select the FLASH: Chip Select low

< Send "Write Enable" instruction

< Select the FLASH: Chip Select low

< Select the FLASH: Chip Select low

Definition at line **1175** of file **stm3210e_eval.c**.

References **FLASH_SPI_CMD_WREN**, **FLASH_SPI_CS_HIGH**, **FLASH_SPI_CS_LOW**, and **SPIx_Write()**.

Referenced by **BSP_SERIAL_FLASH_EraseBulk()**, **BSP_SERIAL_FLASH_EraseSector()**, and **BSP_SERIAL_FLASH_WritePage()**.

void LCD_Delay (uint32_t Delay)

Wait for loop in ms.

Parameters:

Delay in ms.

Return values:

None

Definition at line **1070** of file **stm3210e_eval.c**.

void LCD_IO_Init (void)

Initializes LCD low level.

Return values:

None

Definition at line **1006** of file **stm3210e_eval.c**.

References **FSMC_BANK1NORSRAM4_Init()**.

uint16_t LCD_IO_ReadData (uint16_t Reg)

Reads data from LCD data register.

Parameters:

Reg,: Register to be read

Return values:

Read data.

Definition at line **1059** of file **stm3210e_eval.c**.

References **FSMC_BANK1NORSRAM4_ReadData()**.

void LCD_IO_WriteData (uint16_t RegValue)

Writes data on LCD data register.

Parameters:

RegValue,: Data to be written

Return values:

None

Definition at line **1016** of file **stm3210e_eval.c**.

References **FSMC_BANK1NORSRAM4_WriteData()**.

void LCD_IO_WriteMultipleData (uint8_t * pData,
uint32_t Size
)

Writes multiple data on LCD data register.

Parameters:

pData,: Data to be written

Size,: number of data to write

Return values:

None

Definition at line **1027** of file **stm3210e_eval.c**.

References **FSMC_BANK1NORSRAM4_WriteData()**.

void LCD_IO_WriteReg (uint8_t Reg)

Writes register on LCD register.

Parameters:

Reg,: Register to be written

Return values:

None

Definition at line **1049** of file **stm3210e_eval.c**.

References **FSMC_BANK1NORSRAM4_WriteReg()**.

void TSENSOR_IO_Init (void)

Initializes peripherals used by the I2C Temperature Sensor driver.

Return values:

None

Definition at line **1261** of file **stm3210e_eval.c**.

References **I2Cx_Init()**.

uint16_t TSENSOR_IO_IsDeviceReady (uint16_t DevAddress, uint32_t Trials)

Checks if Temperature Sensor is ready for communication.

Parameters:

DevAddress,: Target device address

Trials,: Number of trials

Return values:

HAL status

Definition at line **1298** of file **stm3210e_eval.c**.

References **I2Cx_IsDeviceReady()**.

void TSENSOR_IO_Read (**uint16_t DevAddress,**
uint8_t * pBuffer,
uint8_t ReadAddr,
uint16_t Length
)

Reads one byte from the TSENSOR.

Parameters:

DevAddress,:	Target device address
pBuffer	: pointer to the buffer that receives the data read from the TSENSOR.
ReadAddr	: TSENSOR's internal address to read from.
Length,:	Number of data to read

Return values:

None

Definition at line **1287** of file **stm3210e_eval.c**.

References **I2Cx_ReadBuffer()**.

void TSENSOR_IO_Write (**uint16_t DevAddress,**
uint8_t * pBuffer,
uint8_t WriteAddr,
uint16_t Length
)

Writes one byte to the TSENSOR.

Parameters:

DevAddress,:	Target device address
pBuffer,:	Pointer to data buffer
WriteAddr,:	TSENSOR's internal address to write to.
Length,:	Number of data to write

Return values:

None

Definition at line **1274** of file **stm3210e_eval.c**.

References **I2Cx_WriteBuffer()**.

STM3210E_EVAL BSP User Manual

AUDIO_Private_Variables

STM3210E_EVAL_AUDIO

Defines

```
#define AUDIO_SAMPLE_SENDDUMMYDATA_SIZE 8
```

Variables

const uint16_t	**audio_sample_SendDummyData** []
static __IO uint32_t	**SendDummyData** = DISABLE
static AUDIO_DrvTypeDef *	**pAudioDrv**
I2S_HandleTypeDef	**hAudioOutI2s**

Define Documentation

#define AUDIO_SAMPLE_SENDDUMMYDATA_SIZE 8

Definition at line **172** of file **stm3210e_eval_audio.c**.

Referenced by **BSP_AUDIO_OUT_Init()**.

Variable Documentation

const uint16_t audio_sample_SendDummyData[]

Initial value:

```
  {0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000
, 0x0000, 0x0000}
```

Definition at line 174 of file stm3210e_eval_audio.c.

Referenced by BSP_AUDIO_OUT_Init().

I2S_HandleTypeDef hAudioOutI2s

Definition at line 182 of file stm3210e_eval_audio.c.

Referenced by BSP_AUDIO_OUT_ChangeBuffer(), BSP_AUDIO_OUT_Init(), BSP_AUDIO_OUT_Pause(), BSP_AUDIO_OUT_Play(), BSP_AUDIO_OUT_Resume(), BSP_AUDIO_OUT_Stop(), I2SOUT_Init(), I2SOUT_MspInit(), I2SOUT_SendDummyData_Start(), and I2SOUT_SendDummyData_Stop().

AUDIO_DrvTypeDef* pAudioDrv [static]

Definition at line 181 of file stm3210e_eval_audio.c.

Referenced by BSP_AUDIO_OUT_Init(), BSP_AUDIO_OUT_Pause(), BSP_AUDIO_OUT_Play(), BSP_AUDIO_OUT_Resume(), BSP_AUDIO_OUT_SetMute(), BSP_AUDIO_OUT_SetOutputMode(), BSP_AUDIO_OUT_SetVolume(), and BSP_AUDIO_OUT_Stop().

__IO uint32_t SendDummyData = DISABLE [static]

Definition at line 177 of file stm3210e_eval_audio.c.

Referenced by BSP_AUDIO_OUT_Init(), HAL_I2S_TxCpltCallback(), and I2SOUT_MspInit().

STM3210E_EVAL BSP User Manual

Private Variables

STM3210E-EVAL LCD

Variables

LCD_DrawPropTypeDef	**DrawProp**
static LCD_DrvTypeDef *	**lcd_drv**
static uint8_t	**bitmap** [**MAX_HEIGHT_FONT** ***MAX_WIDTH_FONT** *2+**OFFSET_BITMAP**] = {0}
static uint32_t	**LCD_SwapXY** = 0

Variable Documentation

uint8_t bitmap[MAX_HEIGHT_FONT *MAX_WIDTH_FONT *2+OFFSI

Definition at line 117 of file stm3210e_eval_lcd.c.

Referenced by LCD_DrawChar().

LCD_DrawPropTypeDef DrawProp

Definition at line 112 of file stm3210e_eval_lcd.c.

LCD_DrvTypeDef* lcd_drv [static]

Definition at line 114 of file stm3210e_eval_lcd.c.

Referenced by BSP_LCD_DisplayOff(), BSP_LCD_DisplayOn(), BSP_LCD_DrawBitmap(), BSP_LCD_DrawHLine(), BSP_LCD_DrawVLine(), BSP_LCD_GetXSize(), BSP_LCD_GetYSize(), BSP_LCD_Init(), BSP_LCD_ReadPixel(), LCD_DrawPixel(), and LCD_SetDisplayWindow().

uint32_t LCD_SwapXY = 0 [static]

Definition at line 119 of file stm3210e_eval_lcd.c.

Referenced by BSP_LCD_DrawBitmap(), BSP_LCD_DrawHLine(), BSP_LCD_DrawVLine(), BSP_LCD_Init(), and LCD_DrawPixel().

STM3210E_EVAL BSP User Manual

Defines

Exported_Constants

STM3210E_EVAL NOR

Defines

#define	**NOR_STATUS_OK** 0x00 NOR status structure definition.
#define	**NOR_STATUS_ERROR** 0x01
#define	**NOR_DEVICE_ADDR** ((uint32_t)0x64000000)
#define	**NOR_MEMORY_WIDTH** FSMC_NORSRAM_MEM_BUS_WI
#define	**NOR_BURSTACCESS** FSMC_BURST_ACCESS_MODE_DI
#define	**NOR_WRITEBURST** FSMC_WRITE_BURST_DISABLE
#define	**BLOCKERASE_TIMEOUT** ((uint32_t)0x00A00000) /* NOR b erase timeout */
#define	**CHIPERASE_TIMEOUT** ((uint32_t)0x30000000) /* NOR chip timeout */
#define	**PROGRAM_TIMEOUT** ((uint32_t)0x00004400) /* NOR progra timeout */
#define	**NOR_READY_BUSY_PIN** GPIO_PIN_6
#define	**NOR_READY_BUSY_GPIO** GPIOD
#define	**NOR_READY_STATE** GPIO_PIN_SET
#define	**NOR_BUSY_STATE** GPIO_PIN_RESET

Define Documentation

#define BLOCKERASE_TIMEOUT ((uint32_t)0x00A00000) /* NOR b

Definition at line 95 of file stm3210e_eval_nor.h.

Referenced by BSP_NOR_Erase_Block().

#define CHIPERASE_TIMEOUT ((uint32_t)0x30000000) /* NOR chi

Definition at line 96 of file stm3210e_eval_nor.h.

Referenced by BSP_NOR_Erase_Chip().

#define NOR_BURSTACCESS FSMC_BURST_ACCESS_MODE_DI

Definition at line 88 of file stm3210e_eval_nor.h.

Referenced by BSP_NOR_Init().

#define NOR_BUSY_STATE GPIO_PIN_RESET

Definition at line 103 of file stm3210e_eval_nor.h.

Referenced by HAL_NOR_MspWait().

#define NOR_DEVICE_ADDR ((uint32_t)0x64000000)

Definition at line 83 of file stm3210e_eval_nor.h.

Referenced by BSP_NOR_Erase_Block(), BSP_NOR_Erase_Chip(), BSP_NOR_ProgramData(),

BSP_NOR_ReadData(), and **BSP_NOR_WriteData()**.

#define NOR_MEMORY_WIDTH FSMC_NORSRAM_MEM_BUS_WI

Definition at line **86** of file **stm3210e_eval_nor.h**.

Referenced by **BSP_NOR_Init()**.

#define NOR_READY_BUSY_GPIO GPIOD

Definition at line **101** of file **stm3210e_eval_nor.h**.

Referenced by **HAL_NOR_MspWait()**.

#define NOR_READY_BUSY_PIN GPIO_PIN_6

Definition at line **100** of file **stm3210e_eval_nor.h**.

Referenced by **HAL_NOR_MspWait()**.

#define NOR_READY_STATE GPIO_PIN_SET

Definition at line **102** of file **stm3210e_eval_nor.h**.

Referenced by **HAL_NOR_MspWait()**.

#define NOR_STATUS_ERROR 0x01

Definition at line **81** of file **stm3210e_eval_nor.h**.

Referenced by **BSP_NOR_Erase_Block()**, **BSP_NOR_Erase_Chip()**, **BSP_NOR_Init()**,

BSP_NOR_ProgramData(), **BSP_NOR_Read_ID()**, **BSP_NOR_ReadData()**, and **BSP_NOR_WriteData()**.

#define NOR_STATUS_OK 0x00

NOR status structure definition.

Definition at line **80** of file **stm3210e_eval_nor.h**.

Referenced by **BSP_NOR_Erase_Block()**, **BSP_NOR_Erase_Chip()**, **BSP_NOR_Init()**, **BSP_NOR_ProgramData()**, **BSP_NOR_Read_ID()**, **BSP_NOR_ReadData()**, and **BSP_NOR_WriteData()**.

#define NOR_WRITEBURST FSMC_WRITE_BURST_DISABLE

Definition at line **91** of file **stm3210e_eval_nor.h**.

Referenced by **BSP_NOR_Init()**.

#define PROGRAM_TIMEOUT ((uint32_t)0x00004400) /* NOR prog

Definition at line **97** of file **stm3210e_eval_nor.h**.

Referenced by **BSP_NOR_ProgramData()**, and **BSP_NOR_WriteData()**.

STM3210E_EVAL BSP User Manual

Functions

AUDIO_OUT_Exported_Functions

STM3210E_EVAL_AUDIO

Functions

uint8_t	**BSP_AUDIO_OUT_Init** (uint16_t OutputDevice, uint8_t Volume, uint32_t AudioFreq) Configure the audio peripherals.
uint8_t	**BSP_AUDIO_OUT_Play** (uint16_t *pBuffer, uint32_t Size) Starts playing audio stream from a data buffer for a determined size.
void	**BSP_AUDIO_OUT_ChangeBuffer** (uint16_t *pData, uint16_t Size) Sends n-Bytes on the I2S interface.
uint8_t	**BSP_AUDIO_OUT_Pause** (void) This function Pauses the audio file stream.
uint8_t	**BSP_AUDIO_OUT_Resume** (void) This function Resumes the audio file stream.
uint8_t	**BSP_AUDIO_OUT_Stop** (uint32_t Option) Stops audio playing and Power down the Audio Codec.
uint8_t	**BSP_AUDIO_OUT_SetVolume** (uint8_t Volume) Controls the current audio volume level.
uint8_t	**BSP_AUDIO_OUT_SetMute** (uint32_t Cmd) Enables or disables the MUTE mode by software.
uint8_t	**BSP_AUDIO_OUT_SetOutputMode** (uint8_t Output) Switch dynamically (while audio file is played) the output target (speaker or headphone).
void	**BSP_AUDIO_OUT_SetFrequency** (uint32_t AudioFreq) Update the audio frequency.
void	**HAL_I2S_TxCpltCallback** (I2S_HandleTypeDef *hi2s) Tx Transfer completed callbacks.
void	**HAL_I2S_TxHalfCpltCallback** (I2S_HandleTypeDef *hi2s) Tx Transfer Half completed callbacks.

void	**HAL_I2S_ErrorCallback** (I2S_HandleTypeDef *hi2s) I2S error callbacks.
__weak void	**BSP_AUDIO_OUT_TransferComplete_CallBack** (void) Manages the DMA full Transfer complete event.
__weak void	**BSP_AUDIO_OUT_HalfTransfer_CallBack** (void) Manages the DMA Half Transfer complete event.
__weak void	**BSP_AUDIO_OUT_Error_CallBack** (void) Manages the DMA FIFO error event.
static void	**I2SOUT_MspInit** (void) AUDIO OUT I2S MSP Init.
static void	**I2SOUT_Init** (uint32_t AudioFreq) Initializes the Audio Codec audio interface (I2S)
static uint8_t	**I2SOUT_SendDummyData_Start** (uint16_t *pBuffer, uint32_t Size) Starts sending dummy data as audio stream: function used to generate I2S master clock MCK for audio codec device.
static uint8_t	**I2SOUT_SendDummyData_Stop** (void) Stops sending dummy data as audio stream: function used to stop to generate I2S master clock MCK for audio codec device.

Function Documentation

void BSP_AUDIO_OUT_ChangeBuffer (uint16_t * pData,
uint16_t Size
)

Sends n-Bytes on the I2S interface.

Parameters:

pData,: pointer on data address

Size,: number of data to be written

Return values:

None

Definition at line **301** of file **stm3210e_eval_audio.c**.

References **hAudioOutI2s**.

void BSP_AUDIO_OUT_Error_CallBack (void)

Manages the DMA FIFO error event.

Return values:

None

Definition at line **534** of file **stm3210e_eval_audio.c**.

Referenced by **HAL_I2S_ErrorCallback()**.

void BSP_AUDIO_OUT_HalfTransfer_CallBack (void)

Manages the DMA Half Transfer complete event.

Return values:

None

Definition at line **526** of file **stm3210e_eval_audio.c**.

Referenced by **HAL_I2S_TxHalfCpltCallback()**.

uint8_t BSP_AUDIO_OUT_Init (uint16_t OutputDevice,
uint8_t Volume,
uint32_t AudioFreq
)

Configure the audio peripherals.

Parameters:

OutputDevice,:	OUTPUT_DEVICE_SPEAKER, OUTPUT_DEVICE_HEADPHONE, OUTPUT_DEVICE_BOTH.
Volume,:	Initial volume level (from 0 (Mute) to 100 (Max))
AudioFreq,:	Audio frequency used to play the audio stream.

Return values:

0 if correct communication, else wrong communication

Definition at line **212** of file **stm3210e_eval_audio.c**.

References **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, **audio_sample_SendDummyData**, **AUDIO_SAMPLE_SENDDUMMYDATA_SIZE**, **hAudioOutI2s**, **I2SOUT_Init()**, **I2SOUT_SendDummyData_Start()**, **I2SOUT_SendDummyData_Stop()**, **pAudioDrv**, and **SendDummyData**.

uint8_t BSP_AUDIO_OUT_Pause (void)

This function Pauses the audio file stream.

In case of using DMA, the DMA Pause feature is used.

> **Note:**
> When calling **BSP_AUDIO_OUT_Pause()** function for pause, only **BSP_AUDIO_OUT_Resume()** function should be called for resume (use of **BSP_AUDIO_OUT_Play()** function for resume could lead to unexpected behavior).

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **314** of file **stm3210e_eval_audio.c**.

References **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, **hAudioOutI2s**, and **pAudioDrv**.

uint8_t BSP_AUDIO_OUT_Play (uint16_t * pBuffer, uint32_t Size)

Starts playing audio stream from a data buffer for a determined size.

Parameters:

pBuffer,: Pointer to the buffer

Size,: Number of audio data BYTES.

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **279** of file **stm3210e_eval_audio.c**.

References **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, **DMA_MAX**, **hAudioOutI2s**, and **pAudioDrv**.

uint8_t BSP_AUDIO_OUT_Resume (void)

This function Resumes the audio file stream.

Note:
When calling **BSP_AUDIO_OUT_Pause()** function for pause, only **BSP_AUDIO_OUT_Resume()** function should be called for resume (use of **BSP_AUDIO_OUT_Play()** function for resume could lead to unexpected behavior).

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **338** of file **stm3210e_eval_audio.c**.

References **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, **hAudioOutI2s**, and **pAudioDrv**.

void BSP_AUDIO_OUT_SetFrequency (uint32_t AudioFreq)

Update the audio frequency.

Parameters:

AudioFreq,: Audio frequency used to play the audio stream.

Return values:

None

Note:
This API should be called after the **BSP_AUDIO_OUT_Init()** to adjust the audio frequency.

Definition at line **455** of file **stm3210e_eval_audio.c**.

References **I2SOUT_Init()**.

uint8_t BSP_AUDIO_OUT_SetMute (uint32_t Cmd)

Enables or disables the MUTE mode by software.

Parameters:

Cmd,: could be AUDIO_MUTE_ON to mute sound or AUDIO_MUTE_OFF to unmute the codec and restore previous volume level.

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **413** of file **stm3210e_eval_audio.c**.

References **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, and **pAudioDrv**.

uint8_t BSP_AUDIO_OUT_SetOutputMode (uint8_t Output)

Switch dynamically (while audio file is played) the output target (speaker or headphone).

Parameters:

Output,: specifies the audio output target: OUTPUT_DEVICE_SPEAKER, OUTPUT_DEVICE_HEADPHONE, OUTPUT_DEVICE_BOTH

Return values:

AUDIO_OK if correct communication, else wrong

communication

Definition at line **434** of file **stm3210e_eval_audio.c**.

References **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, and **pAudioDrv**.

uint8_t BSP_AUDIO_OUT_SetVolume (uint8_t Volume)

Controls the current audio volume level.

Parameters:

Volume,: Volume level to be set in percentage from 0% to 100% (0 for Mute and 100 for Max volume level).

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **393** of file **stm3210e_eval_audio.c**.

References **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, and **pAudioDrv**.

uint8_t BSP_AUDIO_OUT_Stop (uint32_t Option)

Stops audio playing and Power down the Audio Codec.

Parameters:

Option,: could be one of the following parameters

- CODEC_PDWN_HW: completely shut down the codec (physically). Then need to reconfigure the Codec after power on.

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **361** of file **stm3210e_eval_audio.c**.

References **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, **AUDIO_RESET_GPIO**, **AUDIO_RESET_PIN**, **hAudioOutI2s**, and **pAudioDrv**.

void BSP_AUDIO_OUT_TransferComplete_CallBack (void)

Manages the DMA full Transfer complete event.

Return values:

None

Definition at line **518** of file **stm3210e_eval_audio.c**.

Referenced by **HAL_I2S_TxCpltCallback()**.

void HAL_I2S_ErrorCallback (I2S_HandleTypeDef * hi2s)

I2S error callbacks.

Parameters:

hi2s,: I2S handle

Return values:

None

Definition at line **504** of file **stm3210e_eval_audio.c**.

References **BSP_AUDIO_OUT_Error_CallBack()**, and **I2SOUT**.

void HAL_I2S_TxCpltCallback (I2S_HandleTypeDef * hi2s)

Tx Transfer completed callbacks.

Parameters:

hi2s,: I2S handle

Return values:

None

Definition at line **466** of file **stm3210e_eval_audio.c**.

References **BSP_AUDIO_OUT_TransferComplete_CallBack()**, **I2SOUT**, and **SendDummyData**.

void HAL_I2S_TxHalfCpltCallback (I2S_HandleTypeDef * hi2s)

Tx Transfer Half completed callbacks.

Parameters:

hi2s,: I2S handle

Return values:

None

Definition at line **489** of file **stm3210e_eval_audio.c**.

References **BSP_AUDIO_OUT_HalfTransfer_CallBack()**, and **I2SOUT**.

static void I2SOUT_Init (uint32_t AudioFreq) [static]

Initializes the Audio Codec audio interface (I2S)

Parameters:

AudioFreq,: Audio frequency to be configured for the I2S

peripheral.

Definition at line 618 of file stm3210e_eval_audio.c.

References hAudioOutI2s, I2SOUT, and I2SOUT_MspInit().

Referenced by BSP_AUDIO_OUT_Init(), and BSP_AUDIO_OUT_SetFrequency().

static void I2SOUT_MspInit (void) [static]

AUDIO OUT I2S MSP Init.

Return values:
None

Definition at line 546 of file stm3210e_eval_audio.c.

References AUDIO_OUT_IRQ_PREPRIO, hAudioOutI2s, I2SOUT, I2SOUT_CLK_ENABLE, I2SOUT_DMAx_CHANNEL, I2SOUT_DMAx_CLK_ENABLE, I2SOUT_DMAx_IRQ, I2SOUT_DMAx_MEM_DATA_SIZE, I2SOUT_DMAx_PERIPH_DATA_SIZE, I2SOUT_MCK_CLK_ENABLE, I2SOUT_MCK_GPIO_PORT, I2SOUT_MCK_PIN, I2SOUT_SCK_PIN, I2SOUT_SCK_SD_CLK_ENABLE, I2SOUT_SCK_SD_GPIO_PORT, I2SOUT_SD_PIN, I2SOUT_WS_CLK_ENABLE, I2SOUT_WS_GPIO_PORT, I2SOUT_WS_PIN, and SendDummyData.

Referenced by I2SOUT_Init().

static uint8_t I2SOUT_SendDummyData_Start (uint16_t * pBuffer, uint32_t Size) [static]

Starts sending dummy data as audio stream: function used to generate I2S master clock MCK for audio codec device.

Parameters:

pBuffer,:	Pointer to the buffer
Size,:	Number of audio data BYTES.

Return values:

AUDIO_OK	if correct communication, else wrong communication

Definition at line **652** of file **stm3210e_eval_audio.c**.

References **AUDIO_OK**, **DMA_MAX**, and **hAudioOutI2s**.

Referenced by **BSP_AUDIO_OUT_Init()**.

static uint8_t I2SOUT_SendDummyData_Stop (void) [static]

Stops sending dummy data as audio stream: function used to stop to generate I2S master clock MCK for audio codec device.

Return values:

AUDIO_OK	if correct communication, else wrong communication

Definition at line **666** of file **stm3210e_eval_audio.c**.

References **AUDIO_OK**, and **hAudioOutI2s**.

Referenced by **BSP_AUDIO_OUT_Init()**.

STM3210E_EVAL BSP User Manual

Exported Functions

STM3210E-EVAL Common

Functions

uint32_t	**BSP_GetVersion** (void) This method returns the STM3210E EVAL BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LED GPIO.
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** Button_Mode) Configures push button GPIO and EXTI Line.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
uint8_t	**BSP_JOY_Init** (**JOYMode_TypeDef** Joy_Mode) Configures all button of the joystick in GPIO or EXTI modes.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the current joystick status.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) Configures COM port.

Function Documentation

void BSP_COM_Init (COM_TypeDef COM,
UART_HandleTypeDef * huart
)

Configures COM port.

Parameters:

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

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

Return values:

None

Definition at line **505** of file **stm3210e_eval.c**.

References **COM_RX_PIN**, **COM_RX_PORT**, **COM_TX_PIN**, **COM_TX_PORT**, **COM_USART**, **COMx_CLK_ENABLE**, **COMx_RX_GPIO_CLK_ENABLE**, and **COMx_TX_GPIO_CLK_ENABLE**.

uint32_t BSP_GetVersion (void)

This method returns the STM3210E EVAL BSP Driver revision.

Return values:

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

Definition at line **260** of file **stm3210e_eval.c**.

References __STM3210E_EVAL_BSP_VERSION.

JOYState_TypeDef BSP_JOY_GetState (void)

Returns the current joystick status.

Return values:

Code of the joystick key pressed This code can be one of the following values:

- JOY_SEL
- JOY_DOWN
- JOY_LEFT
- JOY_RIGHT
- JOY_UP
- JOY_NONE

Definition at line 478 of file stm3210e_eval.c.

References JOY_NONE, JOY_PIN, JOY_PORT, JOY_SEL, and JOYn.

uint8_t BSP_JOY_Init (JOYMode_TypeDef Joy_Mode)

Configures all button of the joystick in GPIO or EXTI modes.

Parameters:

Joy_Mode,: Joystick mode. This parameter can be one of the following values:

- JOY_MODE_GPIO: Joystick pins will be used as simple IOs
- JOY_MODE_EXTI: Joystick pins will be connected to EXTI line with interrupt generation capability

Return values:

HAL_OK,: if all initializations are OK. Other value if error.

Definition at line **430** of file **stm3210e_eval.c**.

References **JOY_IRQn**, **JOY_MODE_EXTI**, **JOY_MODE_GPIO**, **JOY_NONE**, **JOY_PIN**, **JOY_PORT**, **JOY_SEL**, **JOYn**, and **JOYx_GPIO_CLK_ENABLE**.

void BSP_LED_Init (Led_TypeDef Led)

Configures LED GPIO.

Parameters:

Led,: Specifies the Led to be configured. This parameter can be one of following parameters:

- LED1
- LED2
- LED3
- LED4

Return values:

None

Definition at line **275** of file **stm3210e_eval.c**.

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

void BSP_LED_Off (Led_TypeDef Led)

Turns selected LED Off.

Parameters:

Led,: Specifies the Led to be set off. This parameter can be one of following parameters:

- LED1
- LED2
- LED3
- LED4

Return values:

None

Definition at line **318** of file **stm3210e_eval.c**.

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

void BSP_LED_On (Led_TypeDef Led)

Turns selected LED On.

Parameters:

Led,: Specifies the Led to be set on. This parameter can be one of following parameters:

- LED1
- LED2
- LED3
- LED4

Return values:

None

Definition at line **303** of file **stm3210e_eval.c**.

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

void BSP_LED_Toggle (Led_TypeDef Led)

Toggles the selected LED.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Return values:

None

Definition at line **333** of file **stm3210e_eval.c**.

References **LED_PIN**, and **LED_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_TAMPER: Key/Tamper Push Button

Return values:

Button state

Definition at line **416** of file **stm3210e_eval.c**.

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

void BSP_PB_Init (Button_TypeDef Button,
ButtonMode_TypeDef Button_Mode
)

Configures push button GPIO and EXTI Line.

Parameters:

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

- BUTTON_TAMPER: Key/Tamper Push Button
- BUTTON_SEL : Sel Push Button on Joystick
- BUTTON_LEFT : Left Push Button on Joystick
- BUTTON_RIGHT : Right Push Button on Joystick
- BUTTON_DOWN : Down Push Button on Joystick
- BUTTON_UP : Up Push Button on Joystick

Button_Mode,: Button mode requested. This parameter can be one of the following values:

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

Return values:

None

Definition at line **357** of file **stm3210e_eval.c**.

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

BUTTONx_GPIO_CLK_ENABLE.

STM3210E_EVAL BSP User Manual

Exported Functions

STM3210E-EVAL LCD

Functions

uint8_t	**BSP_LCD_Init** (void) Initializes the LCD.
uint32_t	**BSP_LCD_GetXSize** (void) Gets the LCD X size.
uint32_t	**BSP_LCD_GetYSize** (void) Gets the LCD Y size.
uint16_t	**BSP_LCD_GetTextColor** (void) Gets the LCD text color.
uint16_t	**BSP_LCD_GetBackColor** (void) Gets the LCD background color.
void	**BSP_LCD_SetTextColor** (uint16_t Color) Sets the LCD text color.
void	**BSP_LCD_SetBackColor** (uint16_t Color) Sets the LCD background color.
void	**BSP_LCD_SetFont** (sFONT *pFonts) Sets the LCD text font.
sFONT *	**BSP_LCD_GetFont** (void) Gets the LCD text font.
void	**BSP_LCD_Clear** (uint16_t Color) Clears the hole LCD.
void	**BSP_LCD_ClearStringLine** (uint16_t Line) Clears the selected line.
void	**BSP_LCD_DisplayChar** (uint16_t Xpos, uint16_t Ypos, uint8_t Ascii) Displays one character.
void	**BSP_LCD_DisplayStringAt** (uint16_t Xpos, uint16_t Ypos, uint8_t *pText, **Line_ModeTypdef** Mode) Displays characters on the LCD.
void	**BSP_LCD_DisplayStringAtLine** (uint16_t Line, uint8_t *pText) Displays a character on the LCD.

uint16_t	**BSP_LCD_ReadPixel** (uint16_t Xpos, uint16_t Ypos) Reads an LCD pixel.
void	**BSP_LCD_DrawHLine** (uint16_t Xpos, uint16_t Ypos, uint16_t Length) Draws an horizontal line.
void	**BSP_LCD_DrawVLine** (uint16_t Xpos, uint16_t Ypos, uint16_t Length) Draws a vertical line.
void	**BSP_LCD_DrawLine** (uint16_t X1, uint16_t Y1, uint16_t X2, uint16_t Y2) Draws an uni-line (between two points).
void	**BSP_LCD_DrawRect** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Draws a rectangle.
void	**BSP_LCD_DrawCircle** (uint16_t Xpos, uint16_t Ypos, uint16_t Radius) Draws a circle.
void	**BSP_LCD_DrawPolygon** (**pPoint** pPoints, uint16_t PointCount) Draws an poly-line (between many points).
void	**BSP_LCD_DrawEllipse** (int Xpos, int Ypos, int XRadius, int YRadius) Draws an ellipse on LCD.
void	**BSP_LCD_DrawBitmap** (uint16_t Xpos, uint16_t Ypos, uint8_t *pBmp) Draws a bitmap picture loaded in the internal Flash (32 bpp).
void	**BSP_LCD_FillRect** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Draws a full rectangle.
void	**BSP_LCD_FillCircle** (uint16_t Xpos, uint16_t Ypos, uint16_t Radius) Draws a full circle.
	BSP_LCD_FillTriangle (uint16_t X1, uint16_t X2, uint16_t

void	X3, uint16_t Y1, uint16_t Y2, uint16_t Y3) Fill triangle.
void	**BSP_LCD_FillPolygon** (**pPoint** pPoints, uint16_t PointCount) Displays a full poly-line (between many points).
void	**BSP_LCD_FillEllipse** (int Xpos, int Ypos, int XRadius, int YRadius) Draws a full ellipse.
void	**BSP_LCD_DisplayOn** (void) Enables the display.
void	**BSP_LCD_DisplayOff** (void) Disables the display.
void	**BSP_LCD_SetTextColor** (__IO uint16_t Color)
void	**BSP_LCD_SetBackColor** (__IO uint16_t Color)
void	**BSP_LCD_DrawPixel** (uint16_t Xpos, uint16_t Ypos, uint16_t RGB_Code)
void	**BSP_LCD_DrawRGBImage** (uint16_t Xpos, uint16_t Ypos, uint16_t Xsize, uint16_t Ysize, uint8_t *pbmp)

Function Documentation

void BSP_LCD_Clear (uint16_t Color)

Clears the hole LCD.

Parameters:

Color,: Color of the background

Return values:

None

Definition at line **275** of file **stm3210e_eval_lcd.c**.

References **BSP_LCD_DrawHLine()**, **BSP_LCD_GetXSize()**, **BSP_LCD_GetYSize()**, **BSP_LCD_SetTextColor()**, and **LCD_DrawPropTypeDef::TextColor**.

void BSP_LCD_ClearStringLine (uint16_t Line)

Clears the selected line.

Parameters:

Line,: Line to be cleared This parameter can be one of the following values:

- 0..9: if the Current fonts is Font16x24
- 0..19: if the Current fonts is Font12x12 or Font8x12
- 0..29: if the Current fonts is Font8x8

Return values:

None

Definition at line **300** of file **stm3210e_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**,

BSP_LCD_FillRect(), **BSP_LCD_GetXSize()**, **BSP_LCD_SetTextColor()**, **LCD_DrawPropTypeDef::pFont**, and **LCD_DrawPropTypeDef::TextColor**.

void BSP_LCD_DisplayChar (uint16_t Xpos,
uint16_t Ypos,
uint8_t Ascii
)

Displays one character.

Parameters:

Xpos,: Start column address

Ypos,: Line where to display the character shape.

Ascii,: Character ascii code This parameter must be a number between Min_Data = 0x20 and Max_Data = 0x7E

Return values:

None

Definition at line **320** of file **stm3210e_eval_lcd.c**.

References **LCD_DrawChar()**, and **LCD_DrawPropTypeDef::pFont**.

Referenced by **BSP_LCD_DisplayStringAt()**.

void BSP_LCD_DisplayOff (void)

Disables the display.

Return values:

None

Definition at line **991** of file **stm3210e_eval_lcd.c**.

References **lcd_drv**.

void BSP_LCD_DisplayOn (void)

Enables the display.

Return values:

None

Definition at line **982** of file **stm3210e_eval_lcd.c**.

References **lcd_drv**.

void BSP_LCD_DisplayStringAt (uint16_t Xpos,
uint16_t Ypos,
uint8_t * pText,
Line_ModeTypdef Mode
)

Displays characters on the LCD.

Parameters:

Xpos,: X position (in pixel)

Ypos,: Y position (in pixel)

pText,: Pointer to string to display on LCD

Mode,: Display mode This parameter can be one of the following values:

- CENTER_MODE
- RIGHT_MODE
- LEFT_MODE

Return values:

None

Definition at line **338** of file **stm3210e_eval_lcd.c**.

References **BSP_LCD_DisplayChar()**, **BSP_LCD_GetXSize()**, **CENTER_MODE**, **LEFT_MODE**, **LCD_DrawPropTypeDef::pFont**, and **RIGHT_MODE**.

Referenced by **BSP_LCD_DisplayStringAtLine()**.

void BSP_LCD_DisplayStringAtLine (uint16_t Line,
uint8_t * pText
)

Displays a character on the LCD.

Parameters:

Line,: Line where to display the character shape This parameter can be one of the following values:

- 0..9: if the Current fonts is Font16x24
- 0..19: if the Current fonts is Font12x12 or Font8x12
- 0..29: if the Current fonts is Font8x8

pText,: Pointer to string to display on LCD

Return values:

None

Definition at line **397** of file **stm3210e_eval_lcd.c**.

References **BSP_LCD_DisplayStringAt()**, and **LEFT_MODE**.

void BSP_LCD_DrawBitmap (uint16_t Xpos,
uint16_t Ypos,
uint8_t * pBmp

)

Draws a bitmap picture loaded in the internal Flash (32 bpp).

Parameters:

Xpos,: Bmp X position in the LCD

Ypos,: Bmp Y position in the LCD

pBmp,: Pointer to Bmp picture address in the internal Flash

Return values:

None

Definition at line **697** of file **stm3210e_eval_lcd.c**.

References **BSP_LCD_GetXSize()**, **BSP_LCD_GetYSize()**, **lcd_drv**, **LCD_SetDisplayWindow()**, and **LCD_SwapXY**.

Referenced by **LCD_DrawChar()**.

void BSP_LCD_DrawCircle (uint16_t Xpos,
uint16_t Ypos,
uint16_t Radius
)

Draws a circle.

Parameters:

Xpos,: X position

Ypos,: Y position

Radius,: Circle radius

Return values:

None

Definition at line **585** of file **stm3210e_eval_lcd.c**.

References **BSP_LCD_SetFont()**, **LCD_DEFAULT_FONT**, **LCD_DrawPixel()**, and **LCD_DrawPropTypeDef::TextColor**.

Referenced by **BSP_LCD_FillCircle()**.

void BSP_LCD_DrawEllipse (int Xpos,
int Ypos,
int XRadius,
int YRadius
)

Draws an ellipse on LCD.

Parameters:

Xpos,: X position
Ypos,: Y position
XRadius,: Ellipse X radius
YRadius,: Ellipse Y radius

Return values:

None

Definition at line **664** of file **stm3210e_eval_lcd.c**.

References **LCD_DrawPixel()**, and **LCD_DrawPropTypeDef::TextColor**.

void BSP_LCD_DrawHLine (uint16_t Xpos,
uint16_t Ypos,
uint16_t Length
)

Draws an horizontal line.

Parameters:

Xpos,: X position

Ypos,: Y position

Length,: Line length

Return values:

None

Definition at line **427** of file **stm3210e_eval_lcd.c**.

References **LCD_DrawPixel()**, **lcd_drv**, **LCD_SwapXY**, and **LCD_DrawPropTypeDef::TextColor**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_DrawRect()**, and **BSP_LCD_FillRect()**.

```
void BSP_LCD_DrawLine ( uint16_t X1,
                        uint16_t Y1,
                        uint16_t X2,
                        uint16_t Y2
                      )
```

Draws an uni-line (between two points).

Parameters:

X1,: **Point** 1 X position

Y1,: **Point** 1 Y position

X2,: **Point** 2 X position

Y2,: **Point** 2 Y position

Return values:

None

Definition at line **492** of file **stm3210e_eval_lcd.c**.

References **ABS**, **LCD_DrawPixel()**, and **LCD_DrawPropTypeDef::TextColor**.

Referenced by **BSP_LCD_DrawPolygon()**, and **BSP_LCD_FillTriangle()**.

void BSP_LCD_DrawPixel (uint16_t Xpos,
uint16_t Ypos,
uint16_t RGB_Code
)

void BSP_LCD_DrawPolygon (pPoint pPoints,
uint16_t PointCount
)

Draws an poly-line (between many points).

Parameters:

pPoints,: Pointer to the points array

PointCount,: Number of points

Return values:

None

Definition at line **635** of file **stm3210e_eval_lcd.c**.

References **BSP_LCD_DrawLine()**, **Point::X**, and **Point::Y**.

void BSP_LCD_DrawRect (uint16_t Xpos,
uint16_t Ypos,
uint16_t Width,
uint16_t Height
)

Draws a rectangle.

Parameters:

Xpos,: X position
Ypos,: Y position
Width,: Rectangle width
Height,: Rectangle height

Return values:

None

Definition at line **567** of file **stm3210e_eval_lcd.c**.

References **BSP_LCD_DrawHLine()**, and **BSP_LCD_DrawVLine()**.

```
void BSP_LCD_DrawRGBImage ( uint16_t  Xpos,
                            uint16_t  Ypos,
                            uint16_t  Xsize,
                            uint16_t  Ysize,
                            uint8_t *  pbmp
                          )
```

```
void BSP_LCD_DrawVLine ( uint16_t  Xpos,
                         uint16_t  Ypos,
                         uint16_t  Length
                       )
```

Draws a vertical line.

Parameters:

Xpos,: X position
Ypos,: Y position
Length,: Line length

Return values:

None

Definition at line **458** of file **stm3210e_eval_lcd.c**.

References **BSP_LCD_GetXSize()**, **BSP_LCD_GetYSize()**, **LCD_DrawPixel()**, **lcd_drv**, **LCD_SetDisplayWindow()**, **LCD_SwapXY**, and **LCD_DrawPropTypeDef::TextColor**.

Referenced by **BSP_LCD_DrawRect()**, **BSP_LCD_FillCircle()**, and **BSP_LCD_FillEllipse()**.

void BSP_LCD_FillCircle (uint16_t Xpos,
uint16_t Ypos,
uint16_t Radius
)

Draws a full circle.

Parameters:

Xpos,: X position

Ypos,: Y position

Radius,: Circle radius

Return values:

None

Definition at line **757** of file **stm3210e_eval_lcd.c**.

References **BSP_LCD_DrawCircle()**, **BSP_LCD_DrawVLine()**, **BSP_LCD_SetTextColor()**, and **LCD_DrawPropTypeDef::TextColor**.

void BSP_LCD_FillEllipse (int Xpos,

```
                                    int  Ypos,
                                    int  XRadius,
                                    int  YRadius
                                  )
```

Draws a full ellipse.

Parameters:

Xpos,:	X position
Ypos,:	Y position
XRadius,:	Ellipse X radius
YRadius,:	Ellipse Y radius

Return values:

None

Definition at line **952** of file **stm3210e_eval_lcd.c**.

References **BSP_LCD_DrawVLine()**.

```
void BSP_LCD_FillPolygon ( pPoint    pPoints,
                           uint16_t  PointCount
                         )
```

Displays a full poly-line (between many points).

Parameters:

pPoints,:	pointer to the points array.
PointCount,:	Number of points.

Return values:

None

Definition at line **883** of file **stm3210e_eval_lcd.c**.

References BSP_LCD_FillTriangle(), POLY_X, POLY_Y, Point::X, and Point::Y.

void BSP_LCD_FillRect (uint16_t Xpos,
uint16_t Ypos,
uint16_t Width,
uint16_t Height
)

Draws a full rectangle.

Parameters:

Xpos,:	X position
Ypos,:	Y position
Width,:	Rectangle width
Height,:	Rectangle height

Return values:

None

Definition at line 740 of file stm3210e_eval_lcd.c.

References BSP_LCD_DrawHLine(), BSP_LCD_SetTextColor(), and LCD_DrawPropTypeDef::TextColor.

Referenced by BSP_LCD_ClearStringLine().

void BSP_LCD_FillTriangle (uint16_t X1,
uint16_t X2,
uint16_t X3,
uint16_t Y1,
uint16_t Y2,
uint16_t Y3

)

Fill triangle.

Parameters:

X1,: specifies the point 1 x position.
Y1,: specifies the point 1 y position.
X2,: specifies the point 2 x position.
Y2,: specifies the point 2 y position.
X3,: specifies the point 3 x position.
Y3,: specifies the point 3 y position.

Return values:

None

Definition at line **809** of file **stm3210e_eval_lcd.c**.

References **ABS**, and **BSP_LCD_DrawLine()**.

Referenced by **BSP_LCD_FillPolygon()**.

uint16_t BSP_LCD_GetBackColor (void)

Gets the LCD background color.

Return values:

Used background color

Definition at line **226** of file **stm3210e_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**.

sFONT * BSP_LCD_GetFont (void)

Gets the LCD text font.

Return values:
Used font

Definition at line **265** of file **stm3210e_eval_lcd.c**.

References **LCD_DrawPropTypeDef::pFont**.

uint16_t BSP_LCD_GetTextColor (void)

Gets the LCD text color.

Return values:
Used text color.

Definition at line **217** of file **stm3210e_eval_lcd.c**.

References **LCD_DrawPropTypeDef::TextColor**.

uint32_t BSP_LCD_GetXSize (void)

Gets the LCD X size.

Return values:
Used LCD X size

Definition at line **199** of file **stm3210e_eval_lcd.c**.

References **lcd_drv**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_ClearStringLine()**, **BSP_LCD_DisplayStringAt()**, **BSP_LCD_DrawBitmap()**, and **BSP_LCD_DrawVLine()**.

uint32_t BSP_LCD_GetYSize (void)

Gets the LCD Y size.

Return values:
Used LCD Y size

Definition at line **208** of file **stm3210e_eval_lcd.c**.

References **lcd_drv**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_DrawBitmap()**, and **BSP_LCD_DrawVLine()**.

uint8_t BSP_LCD_Init (void)

Initializes the LCD.

Return values:
LCD state

Definition at line **143** of file **stm3210e_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**, **BSP_LCD_SetFont()**, **LCD_DEFAULT_FONT**, **lcd_drv**, **LCD_ERROR**, **LCD_OK**, **LCD_SwapXY**, **LCD_DrawPropTypeDef::pFont**, and **LCD_DrawPropTypeDef::TextColor**.

uint16_t BSP_LCD_ReadPixel (uint16_t Xpos, uint16_t Ypos)

Reads an LCD pixel.

Parameters:

Xpos,: X position

Ypos,: Y position

Return values:

RGB pixel color

Definition at line **408** of file **stm3210e_eval_lcd.c**.

References **lcd_drv**.

void BSP_LCD_SetBackColor (__IO uint16_t Color)

void BSP_LCD_SetBackColor (uint16_t Color)

Sets the LCD background color.

Parameters:

Color,: Background color code RGB(5-6-5)

Return values:

None

Definition at line **246** of file **stm3210e_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**.

void BSP_LCD_SetFont (sFONT * pFonts)

Sets the LCD text font.

Parameters:

pFonts,: Font to be used

Return values:

None

Definition at line **256** of file **stm3210e_eval_lcd.c**.

References **LCD_DrawPropTypeDef::pFont**.

Referenced by **BSP_LCD_DrawCircle()**, and **BSP_LCD_Init()**.

void BSP_LCD_SetTextColor (__IO uint16_t Color)

void BSP_LCD_SetTextColor (uint16_t Color)

Sets the LCD text color.

Parameters:

Color,: Text color code RGB(5-6-5)

Return values:

None

Definition at line **236** of file **stm3210e_eval_lcd.c**.

References **LCD_DrawPropTypeDef::TextColor**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_ClearStringLine()**, **BSP_LCD_FillCircle()**, and **BSP_LCD_FillRect()**.

STM3210E_EVAL BSP User Manual

Exported_Functions

STM3210E_EVAL NAND

Functions

uint8_t	**BSP_NAND_Init** (void) Initializes the NAND device.
uint8_t	**BSP_NAND_ReadData** (NAND_AddressTypeDef BlockAddress, uint8_t *pData, uint32_t uwNumPage) Reads an amount of data from the NAND device.
uint8_t	**BSP_NAND_WriteData** (NAND_AddressTypeDef BlockAddress, uint8_t *pData, uint32_t uwNumPage) Writes an amount of data to the NAND device.
uint8_t	**BSP_NAND_Erase_Block** (NAND_AddressTypeDef BlockAddress) Erases the specified block of the NAND device.
uint8_t	**BSP_NAND_Read_ID** (NAND_IDTypeDef *pNAND_ID) Reads NAND flash IDs.
uint8_t	**BSP_NAND_Erase_Chip** (void)
static void	**NAND_MspInit** (void) Initializes the NAND MSP.

Function Documentation

uint8_t BSP_NAND_Erase_Block (NAND_AddressTypeDef BlockA

Erases the specified block of the NAND device.

Parameters:

BlockAddress,: Block address to Erase

Return values:

NAND memory status

Definition at line **224** of file **stm3210e_eval_nand.c**.

References **NAND_OK**, and **nandHandle**.

uint8_t BSP_NAND_Erase_Chip (void)

uint8_t BSP_NAND_Init (void)

Initializes the NAND device.

Return values:

NAND memory status

Definition at line **146** of file **stm3210e_eval_nand.c**.

References **NAND_BLOCK_SIZE**, **NAND_MAX_ZONE**, **NAND_MspInit()**, **NAND_OK**, **NAND_PAGE_SIZE**, **NAND_SPARE_AREA_SIZE**, **NAND_ZONE_SIZE**, **nandHandle**, and **Timing**.

uint8_t BSP_NAND_Read_ID (NAND_IDTypeDef * pNAND_ID)

Reads NAND flash IDs.

Parameters:

pNAND_ID : Pointer to NAND ID structure

Return values:

NAND memory status

Definition at line **240** of file **stm3210e_eval_nand.c**.

References **NAND_OK**, and **nandHandle**.

uint8_t BSP_NAND_ReadData (NAND_AddressTypeDef BlockAddr
uint8_t * pData,
uint32_t uwNumPa
)

Reads an amount of data from the NAND device.

Parameters:

BlockAddress,: Block address to Read
pData,: Pointer to data to be read
uwNumPage,: Number of Pages to read to Block

Return values:

NAND memory status

Definition at line **190** of file **stm3210e_eval_nand.c**.

References **NAND_OK**, and **nandHandle**.

uint8_t BSP_NAND_WriteData (NAND_AddressTypeDef BlockAddr
uint8_t * pData,
uint32_t uwNumPa

)

Writes an amount of data to the NAND device.

Parameters:

BlockAddress,:	Block address to Write
pData,:	Pointer to data to be written
uwNumPage,:	Number of Pages to write to Block

Return values:

NAND memory status

Definition at line **208** of file **stm3210e_eval_nand.c**.

References **NAND_OK**, and **nandHandle**.

static void NAND_MspInit (void) [static]

Initializes the NAND MSP.

Return values:

None

< CLE, ALE, D0->D3, NOE, NWE and NCE2 NAND pin configuration

< D4->D7 NAND pin configuration

< NWAIT NAND pin configuration

< INT2 NAND pin configuration

Definition at line **256** of file **stm3210e_eval_nand.c**.

Referenced by **BSP_NAND_Init()**.

User Manual by  1.7.6.1

STM3210E_EVAL BSP User Manual

Exported_Functions

STM3210E_EVAL NOR

Functions

uint8_t	**BSP_NOR_Init** (void) Initializes the NOR device.
uint8_t	**BSP_NOR_ReadData** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Reads an amount of data from the NOR device.
void	**BSP_NOR_ReturnToReadMode** (void) Returns the NOR memory to read mode.
uint8_t	**BSP_NOR_WriteData** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Writes an amount of data to the NOR device.
uint8_t	**BSP_NOR_ProgramData** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Programs an amount of data to the NOR device.
uint8_t	**BSP_NOR_Erase_Block** (uint32_t BlockAddress) Erases the specified block of the NOR device.
uint8_t	**BSP_NOR_Erase_Chip** (void) Erases the entire NOR chip.
uint8_t	**BSP_NOR_Read_ID** (NOR_IDTypeDef *pNOR_ID) Reads NOR flash IDs.
void	**HAL_NOR_MspWait** (NOR_HandleTypeDef *hnor, uint32_t Timeout) NOR BSP Wait for Ready/Busy signal.
static void	**NOR_MspInit** (void) Initializes the NOR MSP.

Function Documentation

uint8_t BSP_NOR_Erase_Block (uint32_t BlockAddress)

Erases the specified block of the NOR device.

Parameters:

BlockAddress,: Block address to erase

Return values:

NOR memory status

Definition at line **279** of file **stm3210e_eval_nor.c**.

References **BLOCKERASE_TIMEOUT**, **NOR_DEVICE_ADDR**, **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, and **norHandle**.

uint8_t BSP_NOR_Erase_Chip (void)

Erases the entire NOR chip.

Return values:

NOR memory status

Definition at line **299** of file **stm3210e_eval_nor.c**.

References **CHIPERASE_TIMEOUT**, **NOR_DEVICE_ADDR**, **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, and **norHandle**.

uint8_t BSP_NOR_Init (void)

Initializes the NOR device.

Return values:

NOR memory status

Definition at line **151** of file **stm3210e_eval_nor.c**.

References **NOR_BURSTACCESS**, **NOR_MEMORY_WIDTH**, **NOR_MspInit()**, **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, **NOR_WRITEBURST**, **norHandle**, and **Timing**.

uint8_t BSP_NOR_ProgramData (uint32_t uwStartAddress,
uint16_t * pData,
uint32_t uwDataSize
)

Programs an amount of data to the NOR device.

Parameters:

uwStartAddress,:	Write start address
pData,:	Pointer to data to be written
uwDataSize,:	Size of data to write

Return values:

NOR memory status

Definition at line **258** of file **stm3210e_eval_nor.c**.

References **NOR_DEVICE_ADDR**, **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, **norHandle**, and **PROGRAM_TIMEOUT**.

uint8_t BSP_NOR_Read_ID (NOR_IDTypeDef * pNOR_ID)

Reads NOR flash IDs.

Parameters:

pNOR_ID : Pointer to NOR ID structure

Return values:

NOR memory status

Definition at line **320** of file **stm3210e_eval_nor.c**.

References **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, and **norHandle**.

uint8_t BSP_NOR_ReadData (uint32_t uwStartAddress,
uint16_t * pData,
uint32_t uwDataSize
)

Reads an amount of data from the NOR device.

Parameters:

uwStartAddress,:	Read start address
pData,:	Pointer to data to be read
uwDataSize,:	Size of data to read

Return values:

NOR memory status

Definition at line **199** of file **stm3210e_eval_nor.c**.

References **NOR_DEVICE_ADDR**, **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, and **norHandle**.

void BSP_NOR_ReturnToReadMode (void)

Returns the NOR memory to read mode.

Return values:

None

Definition at line 215 of file stm3210e_eval_nor.c.

References norHandle.

uint8_t BSP_NOR_WriteData (uint32_t uwStartAddress,
uint16_t * pData,
uint32_t uwDataSize
)

Writes an amount of data to the NOR device.

Parameters:

uwStartAddress,:	Write start address
pData,:	Pointer to data to be written
uwDataSize,:	Size of data to write

Return values:

NOR memory status

Definition at line 227 of file stm3210e_eval_nor.c.

References NOR_DEVICE_ADDR, NOR_STATUS_ERROR, NOR_STATUS_OK, norHandle, and PROGRAM_TIMEOUT.

void HAL_NOR_MspWait (NOR_HandleTypeDef * hnor,
uint32_t Timeout
)

NOR BSP Wait for Ready/Busy signal.

Parameters:

hnor,:	Pointer to NOR handle
Timeout,:	Timeout duration

Return values:

None

Definition at line **402** of file **stm3210e_eval_nor.c**.

References **NOR_BUSY_STATE**, **NOR_READY_BUSY_GPIO**, **NOR_READY_BUSY_PIN**, and **NOR_READY_STATE**.

static void NOR_MspInit (void) [static]

Initializes the NOR MSP.

Return values:

None

< NOR Data lines configuration

< NOR Address lines configuration

< NOE and NWE configuration

< NE2 configuration

< Configure PD6 for NOR memory Ready/Busy signal

Definition at line **336** of file **stm3210e_eval_nor.c**.

Referenced by **BSP_NOR_Init()**.

STM3210E_EVAL BSP User Manual

Functions

STM3210E_EVAL_SD_Exported_Functions

STM3210E-EVAL SD

Functions

uint8_t	**BSP_SD_Init** (void) Initializes the SD card device.
uint8_t	**BSP_SD_ITConfig** (void) Configures Interrupt mode for SD detection pin.
void	**BSP_SD_DetectIT** (void) SD detect IT treatment.
void	**BSP_SD_DetectCallback** (void) SD detect IT detection callback.
uint8_t	**BSP_SD_ReadBlocks** (uint32_t *pData, uint64_t ReadAddr, uint32_t BlockSize, uint32_t NumOfBlocks) Reads block(s) from a specified address in an SD card, in polling mode.
uint8_t	**BSP_SD_WriteBlocks** (uint32_t *pData, uint64_t WriteAddr, uint32_t BlockSize, uint32_t NumOfBlocks) Writes block(s) to a specified address in an SD card, in polling mode.
uint8_t	**BSP_SD_ReadBlocks_DMA** (uint32_t *pData, uint64_t ReadAddr, uint32_t BlockSize, uint32_t NumOfBlocks) Reads block(s) from a specified address in an SD card, in DMA mode.
uint8_t	**BSP_SD_WriteBlocks_DMA** (uint32_t *pData, uint64_t WriteAddr, uint32_t BlockSize, uint32_t NumOfBlocks)

	Writes block(s) to a specified address in an SD card, in DMA mode.
uint8_t	**BSP_SD_Erase** (uint64_t StartAddr, uint64_t EndAddr) Erases the specified memory area of the given SD card.
void	**BSP_SD_IRQHandler** (void) Handles SD card interrupt request.
void	**BSP_SD_DMA_Tx_IRQHandler** (void) Handles SD DMA Tx transfer interrupt request.
void	**BSP_SD_DMA_Rx_IRQHandler** (void) Handles SD DMA Rx transfer interrupt request.
HAL_SD_TransferStateTypedef	**BSP_SD_GetStatus** (void) Gets the current SD card data status.
void	**BSP_SD_GetCardInfo** (HAL_SD_CardInfoTypedef *CardInfo) Get SD information about specific SD card.
uint8_t	**BSP_SD_IsDetected** (void) Detects if SD card is correctly plugged in the memory slot or not.

Function Documentation

void BSP_SD_DetectCallback (void)

SD detect IT detection callback.

Return values:
None

Definition at line 219 of file stm3210e_eval_sd.c.

Referenced by BSP_SD_DetectIT().

void BSP_SD_DetectIT (void)

SD detect IT treatment.

Return values:
None

Definition at line 208 of file stm3210e_eval_sd.c.

References BSP_SD_DetectCallback().

void BSP_SD_DMA_Rx_IRQHandler (void)

Handles SD DMA Rx transfer interrupt request.

Return values:
None

Definition at line 373 of file stm3210e_eval_sd.c.

References uSdHandle.

void BSP_SD_DMA_Tx_IRQHandler (void)

Handles SD DMA Tx transfer interrupt request.

Return values:

None

Definition at line **364** of file **stm3210e_eval_sd.c**.

References **uSdHandle**.

uint8_t BSP_SD_Erase (uint64_t StartAddr,
uint64_t EndAddr
)

Erases the specified memory area of the given SD card.

Parameters:

StartAddr,: Start byte address

EndAddr,: End byte address

Return values:

SD status

Definition at line **339** of file **stm3210e_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, and **uSdHandle**.

void BSP_SD_GetCardInfo (HAL_SD_CardInfoTypedef * CardInfo)

Get SD information about specific SD card.

Parameters:

CardInfo,: Pointer to HAL_SD_CardInfoTypedef structure

Return values:

None

Definition at line **396** of file **stm3210e_eval_sd.c**.

References **uSdHandle**.

HAL_SD_TransferStateTypedef BSP_SD_GetStatus (void)

Gets the current SD card data status.

Return values:

Data transfer state. This value can be one of the following values:

- SD_TRANSFER_OK: No data transfer is acting
- SD_TRANSFER_BUSY: Data transfer is acting
- SD_TRANSFER_ERROR: Data transfer error

Definition at line **386** of file **stm3210e_eval_sd.c**.

References **uSdHandle**.

uint8_t BSP_SD_Init (void)

Initializes the SD card device.

Return values:

SD status.

Definition at line **122** of file **stm3210e_eval_sd.c**.

References **BSP_SD_IsDetected()**, **MSD_ERROR**, **MSD_OK**, **SD_MspInit()**, **SD_PRESENT**, **uSdCardInfo**, and **uSdHandle**.

void BSP_SD_IRQHandler (void)

Handles SD card interrupt request.

Return values:

None

Definition at line **355** of file **stm3210e_eval_sd.c**.

References **uSdHandle**.

uint8_t BSP_SD_IsDetected (void)

Detects if SD card is correctly plugged in the memory slot or not.

Return values:

Returns if SD is detected or not

Definition at line **192** of file **stm3210e_eval_sd.c**.

References **SD_DETECT_GPIO_PORT**, **SD_DETECT_PIN**, **SD_NOT_PRESENT**, and **SD_PRESENT**.

Referenced by **BSP_SD_Init()**.

uint8_t BSP_SD_ITConfig (void)

Configures Interrupt mode for SD detection pin.

Return values:

Returns 0

Definition at line **170** of file **stm3210e_eval_sd.c**.

References **SD_DETECT_GPIO_PORT**, **SD_DETECT_IRQn**, and

SD_DETECT_PIN.

uint8_t BSP_SD_ReadBlocks (uint32_t * pData,
uint64_t ReadAddr,
uint32_t BlockSize,
uint32_t NumOfBlocks
)

Reads block(s) from a specified address in an SD card, in polling mode.

Parameters:

pData,:	Pointer to the buffer that will contain the data to transmit
ReadAddr,:	Address from where data is to be read
BlockSize,:	SD card data block size, that should be 512
NumOfBlocks,:	Number of SD blocks to read

Return values:

SD status

Definition at line **235** of file **stm3210e_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, and **uSdHandle**.

uint8_t BSP_SD_ReadBlocks_DMA (uint32_t * pData,
uint64_t ReadAddr,
uint32_t BlockSize,
uint32_t NumOfBlocks
)

Reads block(s) from a specified address in an SD card, in DMA mode.

Parameters:

pData,:	Pointer to the buffer that will contain the data to transmit
ReadAddr,:	Address from where data is to be read
BlockSize,:	SD card data block size, that should be 512
NumOfBlocks,:	Number of SD blocks to read

Return values:

SD status

Definition at line **275** of file **stm3210e_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, **SD_DATATIMEOUT**, **SD_DMAConfigRx()**, and **uSdHandle**.

uint8_t BSP_SD_WriteBlocks (uint32_t * pData,
uint64_t WriteAddr,
uint32_t BlockSize,
uint32_t NumOfBlocks
)

Writes block(s) to a specified address in an SD card, in polling mode.

Parameters:

pData,:	Pointer to the buffer that will contain the data to transmit
WriteAddr,:	Address from where data is to be written
BlockSize,:	SD card data block size, that should be 512
NumOfBlocks,:	Number of SD blocks to write

Return values:

SD status

Definition at line **255** of file **stm3210e_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, and **uSdHandle**.

uint8_t BSP_SD_WriteBlocks_DMA (uint32_t * pData,
uint64_t WriteAddr,
uint32_t BlockSize,
uint32_t NumOfBlocks
)

Writes block(s) to a specified address in an SD card, in DMA mode.

Parameters:

pData,:	Pointer to the buffer that will contain the data to transmit
WriteAddr,:	Address from where data is to be written
BlockSize,:	SD card data block size, that should be 512
NumOfBlocks,:	Number of SD blocks to write

Return values:

SD status

Definition at line **308** of file **stm3210e_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, **SD_DATATIMEOUT**, **SD_DMAConfigTx()**, and **uSdHandle**.

STM3210E_EVAL BSP User Manual

Exported_Functions

STM3210E-EVAL SD

Functions

uint8_t	**BSP_SD_Init** (void) Initializes the SD card device.
uint8_t	**BSP_SD_ITConfig** (void) Configures Interrupt mode for SD detection pin.
uint8_t	**BSP_SD_IsDetected** (void) Detects if SD card is correctly plugged in the memory slot or not.
void	**BSP_SD_DetectIT** (void) SD detect IT treatment.
__weak void	**BSP_SD_DetectCallback** (void) SD detect IT detection callback.
uint8_t	**BSP_SD_ReadBlocks** (uint32_t *pData, uint64_t ReadAddr, uint32_t BlockSize, uint32_t NumOfBlocks) Reads block(s) from a specified address in an SD card, in polling mode.
uint8_t	**BSP_SD_WriteBlocks** (uint32_t *pData, uint64_t WriteAddr, uint32_t BlockSize, uint32_t NumOfBlocks) Writes block(s) to a specified address in an SD card, in polling mode.
uint8_t	**BSP_SD_ReadBlocks_DMA** (uint32_t *pData, uint64_t ReadAddr, uint32_t BlockSize, uint32_t NumOfBlocks) Reads block(s) from a specified address in an SD card, in DMA mode.

uint8_t	**BSP_SD_WriteBlocks_DMA** (uint32_t *pData, uint64_t WriteAddr, uint32_t BlockSize, uint32_t NumOfBlocks) Writes block(s) to a specified address in an SD card, in DMA mode.
uint8_t	**BSP_SD_Erase** (uint64_t StartAddr, uint64_t EndAddr) Erases the specified memory area of the given SD card.
void	**BSP_SD_IRQHandler** (void) Handles SD card interrupt request.
void	**BSP_SD_DMA_Tx_IRQHandler** (void) Handles SD DMA Tx transfer interrupt request.
void	**BSP_SD_DMA_Rx_IRQHandler** (void) Handles SD DMA Rx transfer interrupt request.
HAL_SD_TransferStateTypedef	**BSP_SD_GetStatus** (void) Gets the current SD card data status.
void	**BSP_SD_GetCardInfo** (HAL_SD_CardInfoTypedef *CardInfo) Get SD information about specific SD card.

Function Documentation

__weak void BSP_SD_DetectCallback (void)

SD detect IT detection callback.

Return values:
None

Definition at line 219 of file stm3210e_eval_sd.c.

Referenced by BSP_SD_DetectIT().

void BSP_SD_DetectIT (void)

SD detect IT treatment.

Return values:
None

Definition at line 208 of file stm3210e_eval_sd.c.

References BSP_SD_DetectCallback().

void BSP_SD_DMA_Rx_IRQHandler (void)

Handles SD DMA Rx transfer interrupt request.

Return values:
None

Definition at line 373 of file stm3210e_eval_sd.c.

References uSdHandle.

void BSP_SD_DMA_Tx_IRQHandler (void)

Handles SD DMA Tx transfer interrupt request.

Return values:

None

Definition at line **364** of file **stm3210e_eval_sd.c**.

References **uSdHandle**.

uint8_t BSP_SD_Erase (uint64_t StartAddr,
uint64_t EndAddr
)

Erases the specified memory area of the given SD card.

Parameters:

StartAddr,: Start byte address

EndAddr,: End byte address

Return values:

SD status

Definition at line **339** of file **stm3210e_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, and **uSdHandle**.

void BSP_SD_GetCardInfo (HAL_SD_CardInfoTypedef * CardInfo)

Get SD information about specific SD card.

Parameters:

CardInfo,: Pointer to HAL_SD_CardInfoTypedef structure

Return values:

None

Definition at line **396** of file **stm3210e_eval_sd.c**.

References **uSdHandle**.

HAL_SD_TransferStateTypedef BSP_SD_GetStatus (void)

Gets the current SD card data status.

Return values:

Data transfer state. This value can be one of the following values:

- SD_TRANSFER_OK: No data transfer is acting
- SD_TRANSFER_BUSY: Data transfer is acting
- SD_TRANSFER_ERROR: Data transfer error

Definition at line **386** of file **stm3210e_eval_sd.c**.

References **uSdHandle**.

uint8_t BSP_SD_Init (void)

Initializes the SD card device.

Return values:

SD status.

Definition at line **122** of file **stm3210e_eval_sd.c**.

References **BSP_SD_IsDetected()**, **MSD_ERROR**, **MSD_OK**, **SD_MspInit()**, **SD_PRESENT**, **uSdCardInfo**, and **uSdHandle**.

void BSP_SD_IRQHandler (void)

Handles SD card interrupt request.

Return values:

None

Definition at line **355** of file **stm3210e_eval_sd.c**.

References **uSdHandle**.

uint8_t BSP_SD_IsDetected (void)

Detects if SD card is correctly plugged in the memory slot or not.

Return values:

Returns if SD is detected or not

Definition at line **192** of file **stm3210e_eval_sd.c**.

References **SD_DETECT_GPIO_PORT**, **SD_DETECT_PIN**, **SD_NOT_PRESENT**, and **SD_PRESENT**.

Referenced by **BSP_SD_Init()**.

uint8_t BSP_SD_ITConfig (void)

Configures Interrupt mode for SD detection pin.

Return values:

Returns 0

Definition at line **170** of file **stm3210e_eval_sd.c**.

References **SD_DETECT_GPIO_PORT**, **SD_DETECT_IRQn**, and

SD_DETECT_PIN.

uint8_t BSP_SD_ReadBlocks (uint32_t * pData,
uint64_t ReadAddr,
uint32_t BlockSize,
uint32_t NumOfBlocks
)

Reads block(s) from a specified address in an SD card, in polling mode.

Parameters:

pData,:	Pointer to the buffer that will contain the data to transmit
ReadAddr,:	Address from where data is to be read
BlockSize,:	SD card data block size, that should be 512
NumOfBlocks,:	Number of SD blocks to read

Return values:

SD status

Definition at line **235** of file **stm3210e_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, and **uSdHandle**.

uint8_t BSP_SD_ReadBlocks_DMA (uint32_t * pData,
uint64_t ReadAddr,
uint32_t BlockSize,
uint32_t NumOfBlocks
)

Reads block(s) from a specified address in an SD card, in DMA mode.

Parameters:

pData,:	Pointer to the buffer that will contain the data to transmit
ReadAddr,:	Address from where data is to be read
BlockSize,:	SD card data block size, that should be 512
NumOfBlocks,:	Number of SD blocks to read

Return values:

SD status

Definition at line **275** of file **stm3210e_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, **SD_DATATIMEOUT**, **SD_DMAConfigRx()**, and **uSdHandle**.

uint8_t BSP_SD_WriteBlocks (uint32_t * pData,
uint64_t WriteAddr,
uint32_t BlockSize,
uint32_t NumOfBlocks
)

Writes block(s) to a specified address in an SD card, in polling mode.

Parameters:

pData,:	Pointer to the buffer that will contain the data to transmit
WriteAddr,:	Address from where data is to be written
BlockSize,:	SD card data block size, that should be 512
NumOfBlocks,:	Number of SD blocks to write

Return values:

SD status

Definition at line **255** of file **stm3210e_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, and **uSdHandle**.

uint8_t BSP_SD_WriteBlocks_DMA (uint32_t * pData,
uint64_t WriteAddr,
uint32_t BlockSize,
uint32_t NumOfBlocks
)

Writes block(s) to a specified address in an SD card, in DMA mode.

Parameters:

pData,:	Pointer to the buffer that will contain the data to transmit
WriteAddr,:	Address from where data is to be written
BlockSize,:	SD card data block size, that should be 512
NumOfBlocks,:	Number of SD blocks to write

Return values:

SD status

Definition at line **308** of file **stm3210e_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, **SD_DATATIMEOUT**, **SD_DMAConfigTx()**, and **uSdHandle**.

STM3210E_EVAL BSP User Manual

Exported_Functions

STM3210E_EVAL Serial FLASH

Functions

uint8_t	**BSP_SERIAL_FLASH_Init** (void) Initializes peripherals used by the Serial FLASH device.
uint8_t	**BSP_SERIAL_FLASH_EraseSector** (uint32_t SectorAddr) Erases the specified FLASH sector.
uint8_t	**BSP_SERIAL_FLASH_EraseBulk** (void) Erases the entire FLASH.
uint8_t	**BSP_SERIAL_FLASH_WritePage** (uint32_t uwStartAddress, uint8_t *pData, uint32_t uwDataSize) Writes more than one byte to the FLASH with a single WRITE cycle (Page WRITE sequence).
uint8_t	**BSP_SERIAL_FLASH_WriteData** (uint32_t uwStartAddress, uint8_t *pData, uint32_t uwDataSize) Writes block of data to the FLASH.
uint8_t	**BSP_SERIAL_FLASH_ReadData** (uint32_t uwStartAddress, uint8_t *pData, uint32_t uwDataSize) Reads a block of data from the FLASH.
uint32_t	**BSP_SERIAL_FLASH_ReadID** (void) Reads FLASH identification.
HAL_StatusTypeDef	**FLASH_SPI_IO_Init** (void) Initializes the FLASH SPI and put it into StandBy State (Ready for data transfer).
uint8_t	**FLASH_SPI_IO_WriteByte** (uint8_t Data) Write a byte on the FLASH SPI.
uint8_t	**FLASH_SPI_IO_ReadByte** (void) Read a byte from the FLASH SPI.
	FLASH_SPI_IO_ReadData (uint32_t

HAL_StatusTypeDef	MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Read data from FLASH SPI driver.
void	**FLASH_SPI_IO_WriteEnable** (void) Select the FLASH SPI and send "Write Enable" instruction.
HAL_StatusTypeDef	**FLASH_SPI_IO_WaitForWriteEnd** (void) Wait response from the FLASH SPI and Deselect the device.
uint32_t	**FLASH_SPI_IO_ReadID** (void) Reads FLASH SPI identification.

Function Documentation

uint8_t BSP_SERIAL_FLASH_EraseBulk (void)

Erases the entire FLASH.

Return values:

FLASH_OK (0x00) if operation is correctly performed, else return FLASH_ERROR (0x01).

< Bulk Erase

< Select the FLASH and send "Write Enable" instruction

< Send Bulk Erase instruction

< Wait the end of Flash writing and Deselect the FLASH

Definition at line **139** of file **stm3210e_eval_serialflash.c**.

References **FLASH_ERROR**, **FLASH_OK**, **FLASH_SPI_CMD_BE**, **FLASH_SPI_IO_WaitForWriteEnd()**, **FLASH_SPI_IO_WriteByte()**, and **FLASH_SPI_IO_WriteEnable()**.

uint8_t BSP_SERIAL_FLASH_EraseSector (uint32_t SectorAddr)

Erases the specified FLASH sector.

Parameters:

SectorAddr,: address of the sector to erase.

Return values:

FLASH_OK (0x00) if operation is correctly performed, else return FLASH_ERROR (0x01).

< Sector Erase

< Select the FLASH and send "Write Enable" instruction

< Send Sector Erase instruction

< Send SectorAddr high nibble address byte

< Send SectorAddr medium nibble address byte

< Send SectorAddr low nibble address byte

< Wait the end of Flash writing and Deselect the FLASH

Definition at line **108** of file **stm3210e_eval_serialflash.c**.

References **FLASH_ERROR**, **FLASH_OK**, **FLASH_SPI_CMD_SE**, **FLASH_SPI_IO_WaitForWriteEnd()**, **FLASH_SPI_IO_WriteByte()**, and **FLASH_SPI_IO_WriteEnable()**.

uint8_t BSP_SERIAL_FLASH_Init (void)

Initializes peripherals used by the Serial FLASH device.

Return values:

FLASH_OK (0x00) if operation is correctly performed, else return FLASH_ERROR (0x01).

Definition at line **90** of file **stm3210e_eval_serialflash.c**.

References **FLASH_ERROR**, **FLASH_OK**, and **FLASH_SPI_IO_Init()**.

uint8_t BSP_SERIAL_FLASH_ReadData (uint32_t uwStartAddress
uint8_t * pData,
uint32_t uwDataSize

)

Reads a block of data from the FLASH.

Parameters:

pData,:	pointer to the buffer that receives the data read from the FLASH.
uwStartAddress,:	FLASH's internal address to read from.
uwDataSize,:	number of bytes to read from the FLASH.

Return values:

FLASH_OK	(0x00) if operation is correctly performed, else return FLASH_ERROR (0x01).

Definition at line **302** of file **stm3210e_eval_serialflash.c**.

References **FLASH_ERROR**, **FLASH_OK**, and **FLASH_SPI_IO_ReadData()**.

uint32_t BSP_SERIAL_FLASH_ReadID (void)

Reads FLASH identification.

Return values:

FLASH	identification

Definition at line **319** of file **stm3210e_eval_serialflash.c**.

References **FLASH_SPI_IO_ReadID()**.

uint8_t BSP_SERIAL_FLASH_WriteData (uint32_t uwStartAddress
uint8_t * pData,
uint32_t uwDataSize
)

Writes block of data to the FLASH.

In this function, the number of WRITE cycles are reduced, using Page WRITE sequence.

Parameters:

pData,:	pointer to the buffer containing the data to be written to the FLASH.
uwStartAddress,:	FLASH's internal address to write to.
uwDataSize,:	number of bytes to write to the FLASH.

Return values:

FLASH_OK	(0x00) if operation is correctly performed, else return FLASH_ERROR (0x01).

< uwStartAddress is FLASH_SPI_PAGESIZE aligned

< uwDataSize < FLASH_SPI_PAGESIZE

< uwDataSize > FLASH_SPI_PAGESIZE

< uwStartAddress is not FLASH_SPI_PAGESIZE aligned

< uwDataSize < FLASH_SPI_PAGESIZE

< (uwDataSize + uwStartAddress) > FLASH_SPI_PAGESIZE

< uwDataSize > BSP_SERIAL_FLASH_PAGESIZE

Definition at line **215** of file **stm3210e_eval_serialflash.c**.

References **BSP_SERIAL_FLASH_WritePage()**, **FLASH_ERROR**, **FLASH_OK**, and **FLASH_SPI_PAGESIZE**.

uint8_t BSP_SERIAL_FLASH_WritePage (uint32_t uwStartAddres
uint8_t * pData,
uint32_t uwDataSize

)

Writes more than one byte to the FLASH with a single WRITE cycle (Page WRITE sequence).

Note:
The number of byte can't exceed the FLASH page size (FLASH_SPI_PAGESIZE).

Parameters:

pData,:	pointer to the buffer containing the data to be written to the FLASH.
uwStartAddress,:	FLASH's internal address to write to.
uwDataSize,:	number of bytes to write to the FLASH, must be equal or less than "FLASH_SPI_PAGESIZE" value.

Return values:

FLASH_OK	(0x00) if operation is correctly performed, else return FLASH_ERROR (0x01).

< Select the FLASH and send "Write Enable" instruction

< Send "Write to Memory " instruction

< Send uwStartAddress high nibble address byte to write to

< Send uwStartAddress medium nibble address byte to write to

< Send uwStartAddress low nibble address byte to write to

< while there is data to be written on the FLASH

< Send the current byte

< **Point** on the next byte to be written

Definition at line **171** of file **stm3210e_eval_serialflash.c**.

References **FLASH_ERROR**, **FLASH_OK**, **FLASH_SPI_CMD_WRITE**, **FLASH_SPI_IO_WaitForWriteEnd()**, **FLASH_SPI_IO_WriteByte()**, and **FLASH_SPI_IO_WriteEnable()**.

Referenced by **BSP_SERIAL_FLASH_WriteData()**.

HAL_StatusTypeDef FLASH_SPI_IO_Init (void)

Initializes the FLASH SPI and put it into StandBy State (Ready for data transfer).

Return values:

None

Definition at line **1084** of file **stm3210e_eval.c**.

References **FLASH_SPI_CS_GPIO_CLK_ENABLE**, **FLASH_SPI_CS_GPIO_PORT**, **FLASH_SPI_CS_HIGH**, **FLASH_SPI_CS_PIN**, and **SPIx_Init()**.

Referenced by **BSP_SERIAL_FLASH_Init()**.

uint8_t FLASH_SPI_IO_ReadByte (void)

Read a byte from the FLASH SPI.

Return values:

uint8_t (The received byte).

Definition at line **1124** of file **stm3210e_eval.c**.

References **SPIx_Read()**.

HAL_StatusTypeDef FLASH_SPI_IO_ReadData (uint32_t MemAdd
uint8_t * pBuffer,

uint32_t BufferSiz
)

Read data from FLASH SPI driver.

Parameters:

MemAddress,:	Internal memory address
pBuffer,:	Pointer to data buffer
BufferSize,:	Amount of data to be read

Return values:

HAL_StatusTypeDef	HAL Status

< Select the FLASH: Chip Select low

< Send "Read from Memory " instruction

< Send ReadAddr high nibble address byte to read from

< Send ReadAddr medium nibble address byte to read from

< Send ReadAddr low nibble address byte to read from

< while there is data to be read

< Read a byte from the FLASH

< **Point** to the next location where the byte read will be saved

Definition at line **1142** of file **stm3210e_eval.c**.

References **FLASH_SPI_CMD_READ**, **FLASH_SPI_CS_HIGH**, **FLASH_SPI_CS_LOW**, **FLASH_SPI_DUMMY_BYTE**, and **SPIx_Write()**.

Referenced by **BSP_SERIAL_FLASH_ReadData()**.

uint32_t FLASH_SPI_IO_ReadID (void)

Reads FLASH SPI identification.

Return values:

FLASH identification

< Select the FLASH: Chip Select low

< Send "RDID " instruction

< Read a byte from the FLASH

< Read a byte from the FLASH

< Read a byte from the FLASH

< Deselect the FLASH: Chip Select high

Definition at line **1227** of file **stm3210e_eval.c**.

References **FLASH_SPI_CS_HIGH**, **FLASH_SPI_CS_LOW**, **FLASH_SPI_DUMMY_BYTE**, and **SPIx_Write()**.

Referenced by **BSP_SERIAL_FLASH_ReadID()**.

HAL_StatusTypeDef FLASH_SPI_IO_WaitForWriteEnd (void)

Wait response from the FLASH SPI and Deselect the device.

Return values:

HAL_StatusTypeDef HAL Status

< Select the FLASH: Chip Select low

< Select the FLASH: Chip Select low

< Send "Read Status Register" instruction

< Loop as long as the memory is busy with a write cycle

< Send a dummy byte to generate the clock needed by the FLASH and put the value of the status register in FLASH_Status variable

< Deselect the FLASH: Chip Select high

Definition at line **1194** of file **stm3210e_eval.c**.

References **FLASH_SPI_CMD_RDSR**, **FLASH_SPI_CS_HIGH**, **FLASH_SPI_CS_LOW**, **FLASH_SPI_DUMMY_BYTE**, **FLASH_SPI_WIP_FLAG**, and **SPIx_Write()**.

Referenced by **BSP_SERIAL_FLASH_EraseBulk()**, **BSP_SERIAL_FLASH_EraseSector()**, and **BSP_SERIAL_FLASH_WritePage()**.

uint8_t FLASH_SPI_IO_WriteByte (uint8_t Data)

Write a byte on the FLASH SPI.

Parameters:

Data,: byte to send.

Return values:

None

Definition at line **1114** of file **stm3210e_eval.c**.

References **SPIx_Write()**.

Referenced by **BSP_SERIAL_FLASH_EraseBulk()**, **BSP_SERIAL_FLASH_EraseSector()**, and **BSP_SERIAL_FLASH_WritePage()**.

void FLASH_SPI_IO_WriteEnable (void)

Select the FLASH SPI and send "Write Enable" instruction.

Return values:

None

< Select the FLASH: Chip Select low

< Send "Write Enable" instruction

< Select the FLASH: Chip Select low

< Select the FLASH: Chip Select low

Definition at line **1175** of file **stm3210e_eval.c**.

References **FLASH_SPI_CMD_WREN**, **FLASH_SPI_CS_HIGH**, **FLASH_SPI_CS_LOW**, and **SPIx_Write()**.

Referenced by **BSP_SERIAL_FLASH_EraseBulk()**, **BSP_SERIAL_FLASH_EraseSector()**, and **BSP_SERIAL_FLASH_WritePage()**.

STM3210E_EVAL BSP User Manual

Exported_Functions

STM3210E_EVAL SRAM

Functions

uint8_t	**BSP_SRAM_Init** (void) Initializes the SRAM device.
uint8_t	**BSP_SRAM_ReadData** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Reads an amount of data from the SRAM device in polling mode.
uint8_t	**BSP_SRAM_ReadData_DMA** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Reads an amount of data from the SRAM device in DMA mode.
uint8_t	**BSP_SRAM_WriteData** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Writes an amount of data from the SRAM device in polling mode.
uint8_t	**BSP_SRAM_WriteData_DMA** (uint32_t uwStartAddress, uint16_t *pData, uint32_t uwDataSize) Writes an amount of data from the SRAM device in DMA mode.
void	**BSP_SRAM_DMA_IRQHandler** (void) Handles SRAM DMA transfer interrupt request.
static void	**SRAM_MspInit** (void) Initializes SRAM MSP.

Function Documentation

void BSP_SRAM_DMA_IRQHandler (void)

Handles SRAM DMA transfer interrupt request.

Return values:
None

Definition at line 271 of file stm3210e_eval_sram.c.

References sramHandle.

uint8_t BSP_SRAM_Init (void)

Initializes the SRAM device.

Return values:
SRAM status

Definition at line 150 of file stm3210e_eval_sram.c.

References SRAM_BURSTACCESS, SRAM_ERROR, SRAM_MEMORY_WIDTH, SRAM_MspInit(), SRAM_OK, SRAM_WRITEBURST, sramHandle, and Timing.

uint8_t BSP_SRAM_ReadData (uint32_t uwStartAddress, uint16_t * pData, uint32_t uwDataSize)

Reads an amount of data from the SRAM device in polling mode.

Parameters:

uwStartAddress,: Read start address
pData,: Pointer to data to be read
uwDataSize,: Size of read data from the memory

Return values:
SRAM status

Definition at line **198** of file **stm3210e_eval_sram.c**.

References **SRAM_ERROR**, **SRAM_OK**, and **sramHandle**.

uint8_t BSP_SRAM_ReadData_DMA (uint32_t uwStartAddress,
uint16_t * pData,
uint32_t uwDataSize
)

Reads an amount of data from the SRAM device in DMA mode.

Parameters:
uwStartAddress,: Read start address
pData,: Pointer to data to be read
uwDataSize,: Size of read data from the memory

Return values:
SRAM status

Definition at line **217** of file **stm3210e_eval_sram.c**.

References **SRAM_ERROR**, **SRAM_OK**, and **sramHandle**.

uint8_t BSP_SRAM_WriteData (uint32_t uwStartAddress,
uint16_t * pData,
uint32_t uwDataSize
)

Writes an amount of data from the SRAM device in polling mode.

Parameters:

uwStartAddress,:	Write start address
pData,:	Pointer to data to be written
uwDataSize,:	Size of written data from the memory

Return values:

SRAM status

Definition at line **236** of file **stm3210e_eval_sram.c**.

References **SRAM_ERROR**, **SRAM_OK**, and **sramHandle**.

uint8_t BSP_SRAM_WriteData_DMA (uint32_t uwStartAddress,
uint16_t * pData,
uint32_t uwDataSize
)

Writes an amount of data from the SRAM device in DMA mode.

Parameters:

uwStartAddress,:	Write start address
pData,:	Pointer to data to be written
uwDataSize,:	Size of written data from the memory

Return values:

SRAM status

Definition at line **255** of file **stm3210e_eval_sram.c**.

References **SRAM_ERROR**, **SRAM_OK**, and **sramHandle**.

static void SRAM_MspInit (void) [static]

Initializes SRAM MSP.

Return values:

None

< SRAM Data lines configuration

< SRAM Address lines configuration

< NOE and NWE configuration

< NE3 configuration

< NBL0, NBL1 configuration

Definition at line **280** of file **stm3210e_eval_sram.c**.

References **sramHandle**.

Referenced by **BSP_SRAM_Init()**.

STM3210E_EVAL BSP User Manual

Exported Functions

STM3210E_EVAL TSENSOR

Functions

uint32_t	**BSP_TSENSOR_Init** (void) Initializes peripherals used by the I2C Temperature Sensor driver.
uint8_t	**BSP_TSENSOR_ReadStatus** (void) Returns the Temperature Sensor status.
uint16_t	**BSP_TSENSOR_ReadTemp** (void) Read Temperature register of STLM75.

Function Documentation

uint32_t BSP_TSENSOR_Init (void)

Initializes peripherals used by the I2C Temperature Sensor driver.

Return values:

TSENSOR status

Definition at line **99** of file **stm3210e_eval_tsensor.c**.

References **tsensor_drv**, **TSENSOR_ERROR**, **TSENSOR_I2C_ADDRESS_A01**, **TSENSOR_I2C_ADDRESS_A02**, **TSENSOR_MAX_TRIALS**, **TSENSOR_OK**, and **TSENSORAddress**.

uint8_t BSP_TSENSOR_ReadStatus (void)

Returns the Temperature Sensor status.

Return values:

The Temperature Sensor status.

Definition at line **151** of file **stm3210e_eval_tsensor.c**.

References **tsensor_drv**, and **TSENSORAddress**.

uint16_t BSP_TSENSOR_ReadTemp (void)

Read Temperature register of STLM75.

Return values:

STLM75 measured temperature value.

Definition at line **160** of file **stm3210e_eval_tsensor.c**.

References **tsensor_drv**, and **TSENSORAddress**.

STM3210E_EVAL BSP User Manual

Exported Types

STM3210E-EVAL Common

Enumerations

enum	**Led_TypeDef** {
	LED1 = 0, **LED2** = 1, **LED3** = 2, **LED4** = 3,
	LED_GREEN = LED1, **LED_ORANGE** = LED2, **LED_RED** = LED3, **LED_BLUE** = LED4
	}
	LED Types Definition. More...

enum	**Button_TypeDef** {
	BUTTON_WAKEUP = 0, **BUTTON_TAMPER** = 1,
	BUTTON_KEY = 2, **BUTTON_SEL** = 3,
	BUTTON_LEFT = 4, **BUTTON_RIGHT** = 5,
	BUTTON_DOWN = 6, **BUTTON_UP** = 7
	}
	BUTTON Types Definition. More...

enum	**ButtonMode_TypeDef** { **BUTTON_MODE_GPIO** = 0, **BUTTON_MODE_EXTI** = 1, **BUTTON_MODE_EVT** = 2 }

enum	**JOYState_TypeDef** {
	JOY_SEL = 0, **JOY_LEFT** = 1, **JOY_RIGHT** = 2,
	JOY_DOWN = 3,
	JOY_UP = 4, **JOY_NONE** = 5
	}
	JOYSTICK Types Definition. More...

enum	**JOYMode_TypeDef** { **JOY_MODE_GPIO** = 0, **JOY_MODE_EXTI** = 1 }
enum	**COM_TypeDef** { **COM1** = 0, **COM2** = 1 } COM Types Definition. More...

Enumeration Type Documentation

enum Button_TypeDef

BUTTON Types Definition.

Enumerator:

BUTTON_WAKEUP
BUTTON_TAMPER
BUTTON_KEY
BUTTON_SEL
BUTTON_LEFT
BUTTON_RIGHT
BUTTON_DOWN
BUTTON_UP

Definition at line **86** of file **stm3210e_eval.h**.

enum ButtonMode_TypeDef

Enumerator:

BUTTON_MODE_GPIO
BUTTON_MODE_EXTI
BUTTON_MODE_EVT

Definition at line **99** of file **stm3210e_eval.h**.

enum COM_TypeDef

COM Types Definition.

Enumerator:

COM1
COM2

Definition at line **131** of file **stm3210e_eval.h**.

enum JOYMode_TypeDef

Enumerator:
JOY_MODE_GPIO
JOY_MODE_EXTI

Definition at line **121** of file **stm3210e_eval.h**.

enum JOYState_TypeDef

JOYSTICK Types Definition.

Enumerator:
JOY_SEL
JOY_LEFT
JOY_RIGHT
JOY_DOWN
JOY_UP
JOY_NONE

Definition at line **110** of file **stm3210e_eval.h**.

enum Led_TypeDef

LED Types Definition.

Enumerator:
LED1

LED2
LED3
LED4
LED_GREEN
LED_ORANGE
LED_RED
LED_BLUE

Definition at line **69** of file **stm3210e_eval.h**.

STM3210E_EVAL BSP User Manual

Private Variables

STM3210E-EVAL Common

Variables

GPIO_TypeDef *	**LED_PORT** [**LEDn**] LED variables.
const uint16_t	**LED_PIN** [**LEDn**]
GPIO_TypeDef *	**BUTTON_PORT** [**BUTTONn**] BUTTON variables.
const uint16_t	**BUTTON_PIN** [**BUTTONn**]
const uint8_t	**BUTTON_IRQn** [**BUTTONn**]
GPIO_TypeDef *	**JOY_PORT** [**JOYn**] JOYSTICK variables.
const uint16_t	**JOY_PIN** [**JOYn**]
const uint8_t	**JOY_IRQn** [**JOYn**]
USART_TypeDef *	**COM_USART** [**COMn**] = {**EVAL_COM1**, **EVAL_COM2**} COM variables.
GPIO_TypeDef *	**COM_TX_PORT** [**COMn**] = {**EVAL_COM1_TX_GPIO_PORT**, **EVAL_COM2_TX_GPIO_PORT**}
GPIO_TypeDef *	**COM_RX_PORT** [**COMn**] = {**EVAL_COM1_RX_GPIO_PORT**, **EVAL_COM2_RX_GPIO_PORT**}
const uint16_t	**COM_TX_PIN** [**COMn**] = {**EVAL_COM1_TX_PIN**, **EVAL_COM2_TX_PIN**}
const uint16_t	**COM_RX_PIN** [**COMn**] = {**EVAL_COM1_RX_PIN**, **EVAL_COM2_RX_PIN**}
uint32_t	**SpixTimeout** = **EVAL_SPIx_TIMEOUT_MAX** BUS variables.
static SPI_HandleTypeDef	**heval_Spi**
uint32_t	**I2cxTimeout** = **EVAL_I2Cx_TIMEOUT_MAX**

I2C_HandleTypeDef **heval_I2c**

Variable Documentation

const uint8_t BUTTON_IRQn[BUTTONn]

Initial value:

```
 {WAKEUP_BUTTON_EXTI_IRQn,
                                                TAMPER_BUTT
ON_EXTI_IRQn,
                                                KEY_BUTTON_
EXTI_IRQn,
                                                SEL_JOY_EXT
I_IRQn,
                                                LEFT_JOY_EX
TI_IRQn,
                                                RIGHT_JOY_E
XTI_IRQn,
                                                DOWN_JOY_EX
TI_IRQn,
                                                UP_JOY_EXTI
_IRQn}
```

Definition at line **132** of file **stm3210e_eval.c**.

Referenced by **BSP_PB_Init()**.

const uint16_t BUTTON_PIN[BUTTONn]

Initial value:

```
 {WAKEUP_BUTTON_PIN,
                                                TAMPER_BUTT
ON_PIN,
                                                KEY_BUTTON_
PIN,
                                                SEL_JOY_PIN
,
                                                LEFT_JOY_PIN
```

```
,
                                                RIGHT_JOY_P
IN,
                                                DOWN_JOY_PIN
,
                                                UP_JOY_PIN}
```

Definition at line **123** of file **stm3210e_eval.c**.

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

GPIO_TypeDef* BUTTON_PORT[BUTTONn]

Initial value:

```
 {WAKEUP_BUTTON_GPIO_PORT,
                                                TAMPER_BUTT
ON_GPIO_PORT,
                                                KEY_BUTTON_
GPIO_PORT,
                                                SEL_JOY_GPI
O_PORT,
                                                LEFT_JOY_GP
IO_PORT,
                                                RIGHT_JOY_G
PIO_PORT,
                                                DOWN_JOY_GP
IO_PORT,
                                                UP_JOY_GPIO
_PORT}
```

BUTTON variables.

Definition at line **114** of file **stm3210e_eval.c**.

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

const uint16_t COM_RX_PIN[COMn] = {EVAL_COM1_RX_PIN, EVAL

Definition at line 173 of file stm3210e_eval.c.

Referenced by BSP_COM_Init().

GPIO_TypeDef* COM_RX_PORT[COMn] = {EVAL_COM1_RX_GPIO_

Definition at line 169 of file stm3210e_eval.c.

Referenced by BSP_COM_Init().

const uint16_t COM_TX_PIN[COMn] = {EVAL_COM1_TX_PIN, EVAL

Definition at line 171 of file stm3210e_eval.c.

Referenced by BSP_COM_Init().

GPIO_TypeDef* COM_TX_PORT[COMn] = {EVAL_COM1_TX_GPIO_

Definition at line 167 of file stm3210e_eval.c.

Referenced by BSP_COM_Init().

USART_TypeDef* COM_USART[COMn] = {EVAL_COM1, EVAL_COM

COM variables.

Definition at line 165 of file stm3210e_eval.c.

Referenced by BSP_COM_Init().

I2C_HandleTypeDef heval_I2c

Definition at line **185** of file **stm3210e_eval.c**.

Referenced by **I2Cx_Error()**, **I2Cx_Init()**, **I2Cx_IsDeviceReady()**, **I2Cx_ReadBuffer()**, **I2Cx_ReadData()**, **I2Cx_WriteBuffer()**, and **I2Cx_WriteData()**.

SPI_HandleTypeDef heval_Spi `[static]`

Definition at line **180** of file **stm3210e_eval.c**.

Referenced by **SPIx_Error()**, **SPIx_Init()**, and **SPIx_Write()**.

uint32_t I2cxTimeout = EVAL_I2Cx_TIMEOUT_MAX

Definition at line **184** of file **stm3210e_eval.c**.

Referenced by **I2Cx_IsDeviceReady()**, **I2Cx_ReadBuffer()**, **I2Cx_ReadData()**, **I2Cx_WriteBuffer()**, and **I2Cx_WriteData()**.

const uint8_t JOY_IRQn[JOYn]

Initial value:

```
 {SEL_JOY_EXTI_IRQn,
                                    LEFT_JOY_EXTI_IRQn
,
                                    RIGHT_JOY_EXTI_IR
Qn,
                                    DOWN_JOY_EXTI_IRQn
,
                                    UP_JOY_EXTI_IRQn}
```

Definition at line **156** of file **stm3210e_eval.c**.

Referenced by **BSP_JOY_Init()**.

const uint16_t JOY_PIN[JOYn]

Initial value:

```
 {SEL_JOY_PIN,
                                    LEFT_JOY_PIN,
                                    RIGHT_JOY_PIN,
                                    DOWN_JOY_PIN,
                                    UP_JOY_PIN}
```

Definition at line **150** of file **stm3210e_eval.c**.

Referenced by **BSP_JOY_GetState()**, and **BSP_JOY_Init()**.

GPIO_TypeDef* JOY_PORT[JOYn]

Initial value:

```
 {SEL_JOY_GPIO_PORT,
                                    LEFT_JOY_GPIO_PORT
,
                                    RIGHT_JOY_GPIO_PO
RT,
                                    DOWN_JOY_GPIO_PORT
,
                                    UP_JOY_GPIO_PORT}
```

JOYSTICK variables.

Definition at line **144** of file **stm3210e_eval.c**.

Referenced by **BSP_JOY_GetState()**, and **BSP_JOY_Init()**.

const uint16_t LED_PIN[LEDn]

Initial value:

```
{LED1_PIN,
                                    LED2_PIN,
                                    LED3_PIN,
                                    LED4_PIN}
```

Definition at line **106** of file **stm3210e_eval.c**.

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

GPIO_TypeDef* LED_PORT[LEDn]

Initial value:

```
{LED1_GPIO_PORT,
                                    LED2_GPIO_PORT,
                                    LED3_GPIO_PORT,
                                    LED4_GPIO_PORT}
```

LED variables.

Definition at line **101** of file **stm3210e_eval.c**.

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

uint32_t SpixTimeout = EVAL_SPIx_TIMEOUT_MAX

BUS variables.

Definition at line **179** of file **stm3210e_eval.c**.

Referenced by **SPIx_Write()**.

User Manual by  1.7.6.1

STM3210E_EVAL BSP User Manual

STM3210E_EVAL_BUTTON

Exported Constants

Defines

#define **JOYn** 5

#define **BUTTONn** 3 + **JOYn**

#define **TAMPER_BUTTON_PIN** GPIO_PIN_13 /* PC.13*/
Tamper push-button.

#define **TAMPER_BUTTON_GPIO_PORT** GPIOC

#define **TAMPER_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_G

#define **TAMPER_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_C

#define **TAMPER_BUTTON_EXTI_IRQn** EXTI15_10_IRQn

#define **KEY_BUTTON_PIN** GPIO_PIN_8 /* PG.08*/
Key push-button.

#define **KEY_BUTTON_GPIO_PORT** GPIOG

#define **KEY_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOG

#define **KEY_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_GPIO

#define **KEY_BUTTON_EXTI_IRQn** EXTI9_5_IRQn

#define **WAKEUP_BUTTON_PIN** GPIO_PIN_0 /* PA.00*/
Wake-up push-button.

#define **WAKEUP_BUTTON_GPIO_PORT** GPIOA

#define **WAKEUP_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_G

#define **WAKEUP_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_(

#define **WAKEUP_BUTTON_EXTI_IRQn** EXTI0_IRQn

#define **RIGHT_JOY_PIN** GPIO_PIN_13 /* PG.13*/
Joystick Right push-button.

#define **RIGHT_JOY_GPIO_PORT** GPIOG

#define **RIGHT_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOG_

#define **RIGHT_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOG_

#define **RIGHT_JOY_EXTI_IRQn** EXTI15_10_IRQn

#define **LEFT_JOY_PIN** GPIO_PIN_14 /* PG.14*/
Joystick Left push-button.

#define **LEFT_JOY_GPIO_PORT** GPIOG

#define **LEFT_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOG_C

#define **LEFT_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOG_C

#define **LEFT_JOY_EXTI_IRQn** EXTI15_10_IRQn

#define **UP_JOY_PIN** GPIO_PIN_15 /* PG.15*/
Joystick Up push-button.

#define **UP_JOY_GPIO_PORT** GPIOG

#define **UP_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOG_CLK

#define **UP_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOG_CLK

#define **UP_JOY_EXTI_IRQn** EXTI15_10_IRQn

#define **DOWN_JOY_PIN** GPIO_PIN_3 /* PD.03*/
Joystick Down push-button.

#define **DOWN_JOY_GPIO_PORT** GPIOD

#define **DOWN_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOD_

#define **DOWN_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOD_

#define **DOWN_JOY_EXTI_IRQn** EXTI3_IRQn

#define **SEL_JOY_PIN** GPIO_PIN_7 /* PG.07*/
Joystick Sel push-button.

#define **SEL_JOY_GPIO_PORT** GPIOG

#define **SEL_JOY_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOG_CL

#define **SEL_JOY_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOG_CL

#define **SEL_JOY_EXTI_IRQn** EXTI9_5_IRQn

#define **BUTTONx_GPIO_CLK_ENABLE**(__BUTTON__)

#define **BUTTONx_GPIO_CLK_DISABLE**(__BUTTON__)

#define **JOYx_GPIO_CLK_ENABLE**(__JOY__)

#define **JOYx_GPIO_CLK_DISABLE**(__JOY__)

Define Documentation

#define BUTTONn 3 + JOYn

Definition at line **198** of file **stm3210e_eval.h**.

#define BUTTONx_GPIO_CLK_DISABLE (__BUTTON__)

Value:

```
(((__BUTTON__) == BUTTON_TAMPER) ? TAMPER_BUTTON_
GPIO_CLK_DISABLE()  :\

((__BUTTON__) == BUTTON_KEY) ? KEY_BUTTON_GPIO_CL
K_DISABLE()  :\

((__BUTTON__) == BUTTON_WAKEUP) ? WAKEUP_BUTTON_G
PIO_CLK_DISABLE()  :\

((__BUTTON__) == BUTTON_SEL) ? SEL_JOY_GPIO_CLK_D
ISABLE()  :\

((__BUTTON__) == BUTTON_LEFT) ? LEFT_JOY_GPIO_CLK
_DISABLE()  :\

((__BUTTON__) == BUTTON_RIGHT) ? RIGHT_JOY_GPIO_C
LK_DISABLE()  :\

((__BUTTON__) == BUTTON_DOWN) ? DOWN_JOY_GPIO_CLK
_DISABLE()  :\

((__BUTTON__) == BUTTON_UP) ? UP_JOY_GPIO_CLK_DIS
ABLE() : 0 )
```

Definition at line **281** of file **stm3210e_eval.h**.

#define BUTTONx_GPIO_CLK_ENABLE (__BUTTON__)

Value:

```
do { if ((__BUTTON__) == BUTTON_TAMPER) TAMPER_BU
TTON_GPIO_CLK_ENABLE(); else\

if ((__BUTTON__) == BUTTON_KEY) KEY_BUTTON_GPIO_C
LK_ENABLE(); else\

if ((__BUTTON__) == BUTTON_WAKEUP) WAKEUP_BUTTON_
GPIO_CLK_ENABLE(); else\

if ((__BUTTON__) == BUTTON_SEL) SEL_JOY_GPIO_CLK_
ENABLE(); else\

if ((__BUTTON__) == BUTTON_LEFT) LEFT_JOY_GPIO_CL
K_ENABLE(); else\

if ((__BUTTON__) == BUTTON_RIGHT) RIGHT_JOY_GPIO_
CLK_ENABLE(); else\

if ((__BUTTON__) == BUTTON_DOWN) DOWN_JOY_GPIO_CL
K_ENABLE(); else\

if ((__BUTTON__) == BUTTON_UP) UP_JOY_GPIO_CLK_EN
ABLE();} while(0)
```

Definition at line **272** of file **stm3210e_eval.h**.

Referenced by **BSP_PB_Init()**.

#define DOWN_JOY_EXTI_IRQn EXTI3_IRQn

Definition at line **261** of file **stm3210e_eval.h**.

#define DOWN_JOY_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOI

Definition at line 260 of file stm3210e_eval.h.

#define DOWN_JOY_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOD

Definition at line 259 of file stm3210e_eval.h.

#define DOWN_JOY_GPIO_PORT GPIOD

Definition at line 258 of file stm3210e_eval.h.

#define DOWN_JOY_PIN GPIO_PIN_3 /* PD.03*/

Joystick Down push-button.

Definition at line 257 of file stm3210e_eval.h.

#define JOYn 5

Definition at line 197 of file stm3210e_eval.h.

Referenced by BSP_JOY_GetState(), and BSP_JOY_Init().

#define JOYx_GPIO_CLK_DISABLE (__JOY__)

Value:

```
(((__JOY__) == JOY_SEL) ? SEL_JOY_GPIO_CLK_DISABLE
() :\
                                                ((__JOY
__) == JOY_LEFT) ? LEFT_JOY_GPIO_CLK_DISABLE() :\
                                                ((__JOY
```

```
__) == JOY_RIGHT) ? RIGHT_JOY_GPIO_CLK_DISABLE()
:\
                                                ((__JOY
__) == JOY_DOWN) ? DOWN_JOY_GPIO_CLK_DISABLE() :\
                                                ((__JOY
__) == JOY_UP) ? UP_JOY_GPIO_CLK_DISABLE() : 0 )
```

Definition at line **296** of file **stm3210e_eval.h**.

#define JOYx_GPIO_CLK_ENABLE (__JOY__)

Value:

```
do { if ((__JOY__) == JOY_SEL) SEL_JOY_GPIO_CLK_E
NABLE(); else\
                                                  if
((__JOY__) == JOY_LEFT) LEFT_JOY_GPIO_CLK_ENABLE(
); else\
                                                  if
((__JOY__) == JOY_RIGHT) RIGHT_JOY_GPIO_CLK_ENABLE
(); else\
                                                  if
((__JOY__) == JOY_DOWN) DOWN_JOY_GPIO_CLK_ENABLE(
); else\
                                                  if
((__JOY__) == JOY_UP) UP_JOY_GPIO_CLK_ENABLE();}
while(0)
```

Definition at line **290** of file **stm3210e_eval.h**.

Referenced by **BSP_JOY_Init()**.

#define KEY_BUTTON_EXTI_IRQn EXTI9_5_IRQn

Definition at line **216** of file **stm3210e_eval.h**.

#define KEY_BUTTON_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPI

Definition at line 215 of file stm3210e_eval.h.

#define KEY_BUTTON_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPI(

Definition at line 214 of file stm3210e_eval.h.

#define KEY_BUTTON_GPIO_PORT GPIOG

Definition at line 213 of file stm3210e_eval.h.

#define KEY_BUTTON_PIN GPIO_PIN_8 /* PG.08*/

Key push-button.

Definition at line 212 of file stm3210e_eval.h.

#define LEFT_JOY_EXTI_IRQn EXTI15_10_IRQn

Definition at line 243 of file stm3210e_eval.h.

#define LEFT_JOY_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOG_

Definition at line 242 of file stm3210e_eval.h.

#define LEFT_JOY_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOG_

Definition at line 241 of file stm3210e_eval.h.

#define LEFT_JOY_GPIO_PORT GPIOG

Definition at line 240 of file stm3210e_eval.h.

#define LEFT_JOY_PIN GPIO_PIN_14 /* PG.14*/

Joystick Left push-button.

Definition at line 239 of file stm3210e_eval.h.

#define RIGHT_JOY_EXTI_IRQn EXTI15_10_IRQn

Definition at line 234 of file stm3210e_eval.h.

#define RIGHT_JOY_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIO(

Definition at line 233 of file stm3210e_eval.h.

#define RIGHT_JOY_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOG

Definition at line 232 of file stm3210e_eval.h.

#define RIGHT_JOY_GPIO_PORT GPIOG

Definition at line 231 of file stm3210e_eval.h.

#define RIGHT_JOY_PIN GPIO_PIN_13 /* PG.13*/

Joystick Right push-button.

Definition at line **230** of file **stm3210e_eval.h**.

#define SEL_JOY_EXTI_IRQn EXTI9_5_IRQn

Definition at line **270** of file **stm3210e_eval.h**.

#define SEL_JOY_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOG_(

Definition at line **269** of file **stm3210e_eval.h**.

#define SEL_JOY_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOG_C

Definition at line **268** of file **stm3210e_eval.h**.

#define SEL_JOY_GPIO_PORT GPIOG

Definition at line **267** of file **stm3210e_eval.h**.

#define SEL_JOY_PIN GPIO_PIN_7 /* PG.07*/

Joystick Sel push-button.

Definition at line **266** of file **stm3210e_eval.h**.

#define TAMPER_BUTTON_EXTI_IRQn EXTI15_10_IRQn

Definition at line **207** of file **stm3210e_eval.h**.

#define TAMPER_BUTTON_GPIO_CLK_DISABLE ( ) __HAL_RCC_

Definition at line **206** of file **stm3210e_eval.h**.

#define TAMPER_BUTTON_GPIO_CLK_ENABLE ( ) __HAL_RCC_

Definition at line **205** of file **stm3210e_eval.h**.

#define TAMPER_BUTTON_GPIO_PORT GPIOC

Definition at line **204** of file **stm3210e_eval.h**.

#define TAMPER_BUTTON_PIN GPIO_PIN_13 /* PC.13*/

Tamper push-button.

Definition at line **203** of file **stm3210e_eval.h**.

#define UP_JOY_EXTI_IRQn EXTI15_10_IRQn

Definition at line **252** of file **stm3210e_eval.h**.

#define UP_JOY_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOG_C

Definition at line **251** of file **stm3210e_eval.h**.

#define UP_JOY_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOG_CI

Definition at line **250** of file **stm3210e_eval.h**.

#define UP_JOY_GPIO_PORT GPIOG

Definition at line **249** of file **stm3210e_eval.h**.

#define UP_JOY_PIN GPIO_PIN_15 /* PG.15*/

Joystick Up push-button.

Definition at line **248** of file **stm3210e_eval.h**.

#define WAKEUP_BUTTON_EXTI_IRQn EXTI0_IRQn

Definition at line **225** of file **stm3210e_eval.h**.

#define WAKEUP_BUTTON_GPIO_CLK_DISABLE ( ) __HAL_RCC

Definition at line **224** of file **stm3210e_eval.h**.

#define WAKEUP_BUTTON_GPIO_CLK_ENABLE ( ) __HAL_RCC_

Definition at line **223** of file **stm3210e_eval.h**.

#define WAKEUP_BUTTON_GPIO_PORT GPIOA

Definition at line **222** of file **stm3210e_eval.h**.

#define WAKEUP_BUTTON_PIN GPIO_PIN_0 /* PA.00*/

Wake-up push-button.

Definition at line **221** of file **stm3210e_eval.h**.

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP
User Manual by  1.7.6.1

STM3210E_EVAL BSP User Manual

STM3210E_EVAL_COM

Exported Constants

Defines

#define **COMn** 2

#define **EVAL_COM1** USART1
Definition for COM port1, connected to USART1.

#define **EVAL_COM1_CLK_ENABLE**() __HAL_RCC_USART1_CLK_

#define **EVAL_COM1_CLK_DISABLE**() __HAL_RCC_USART1_CLK

#define **EVAL_COM1_TX_PIN** GPIO_PIN_9 /* PA.09*/

#define **EVAL_COM1_TX_GPIO_PORT** GPIOA

#define **EVAL_COM1_TX_GPIO_CLK_ENABLE**() __HAL_RCC_GPI

#define **EVAL_COM1_TX_GPIO_CLK_DISABLE**() __HAL_RCC_GP

#define **EVAL_COM1_RX_PIN** GPIO_PIN_10 /* PA.10*/

#define **EVAL_COM1_RX_GPIO_PORT** GPIOA

#define **EVAL_COM1_RX_GPIO_CLK_ENABLE**() __HAL_RCC_GPI

#define **EVAL_COM1_RX_GPIO_CLK_DISABLE**() __HAL_RCC_GP

#define **EVAL_COM1_IRQn** USART1_IRQn

#define **EVAL_COM2** USART2
Definition for COM port2, connected to USART2.

#define **EVAL_COM2_CLK_ENABLE**() __HAL_RCC_USART2_CLK_

#define **EVAL_COM2_CLK_DISABLE**() __HAL_RCC_USART2_CLK

#define **EVAL_COM2_TX_PIN** GPIO_PIN_2 /* PA.02*/

#define **EVAL_COM2_TX_GPIO_PORT** GPIOA

#define **EVAL_COM2_TX_GPIO_CLK_ENABLE**() __HAL_RCC_GPI

#define **EVAL_COM2_TX_GPIO_CLK_DISABLE**() __HAL_RCC_GP

#define **EVAL_COM2_RX_PIN** GPIO_PIN_3 /* PA.03*/

#define **EVAL_COM2_RX_GPIO_PORT** GPIOA

#define **EVAL_COM2_RX_GPIO_CLK_ENABLE**() __HAL_RCC_GPI

#define **EVAL_COM2_RX_GPIO_CLK_DISABLE**() __HAL_RCC_GP

#define **EVAL_COM2_IRQn** USART2_IRQn

#define **COMx_CLK_ENABLE**(__INDEX__)

#define **COMx_CLK_DISABLE**(__INDEX__)

#define **COMx_TX_GPIO_CLK_ENABLE**(__INDEX__)

#define **COMx_TX_GPIO_CLK_DISABLE**(__INDEX__)

```
#define COMx_RX_GPIO_CLK_ENABLE(__INDEX__)
#define COMx_RX_GPIO_CLK_DISABLE(__INDEX__)
```

Define Documentation

#define COMn 2

Definition at line 309 of file stm3210e_eval.h.

#define COMx_CLK_DISABLE (__INDEX__)

Value:

```
(((__INDEX__) == COM1) ? EVAL_COM1_CLK_DISABLE()
:\

((__INDEX__) == COM2) ? EVAL_COM2_CLK_DISABLE() :
 0)
```

Definition at line 352 of file stm3210e_eval.h.

#define COMx_CLK_ENABLE (__INDEX__)

Value:

```
do { if((__INDEX__) == COM1) EVAL_COM1_CLK_ENABLE
(); else\

    if((__INDEX__) == COM2) EVAL_COM2_CLK_ENABLE(
);} while(0)
```

Definition at line 349 of file stm3210e_eval.h.

Referenced by BSP_COM_Init().

#define COMx_RX_GPIO_CLK_DISABLE (__INDEX__)

Value:

```
(((__INDEX__) == COM1) ? EVAL_COM1_RX_GPIO_CLK_DI
```

```
SABLE() :\

((__INDEX__) == COM2) ? EVAL_COM2_RX_GPIO_CLK_DIS
ABLE() : 0)
```

Definition at line **364** of file **stm3210e_eval.h**.

#define COMx_RX_GPIO_CLK_ENABLE (__INDEX__)

Value:

```
do { if((__INDEX__) == COM1) EVAL_COM1_RX_GPIO_CL
K_ENABLE(); else\

    if((__INDEX__) == COM2) EVAL_COM2_RX_GPIO_CLK
_ENABLE();} while(0)
```

Definition at line **361** of file **stm3210e_eval.h**.

Referenced by **BSP_COM_Init()**.

#define COMx_TX_GPIO_CLK_DISABLE (__INDEX__)

Value:

```
(((__INDEX__) == COM1) ? EVAL_COM1_TX_GPIO_CLK_DI
SABLE() :\

((__INDEX__) == COM2) ? EVAL_COM2_TX_GPIO_CLK_DIS
ABLE() : 0)
```

Definition at line **358** of file **stm3210e_eval.h**.

#define COMx_TX_GPIO_CLK_ENABLE (__INDEX__)

Value:

```
do { if((__INDEX__) == COM1) EVAL_COM1_TX_GPIO_CL
```

```
K_ENABLE(); else\

    if((__INDEX__) == COM2) EVAL_COM2_TX_GPIO_CLK
_ENABLE();} while(0)
```

Definition at line **355** of file **stm3210e_eval.h**.

Referenced by **BSP_COM_Init()**.

#define EVAL_COM1 USART1

Definition for COM port1, connected to USART1.

Definition at line **314** of file **stm3210e_eval.h**.

#define EVAL_COM1_CLK_DISABLE ( ) __HAL_RCC_USART1_CL

Definition at line **316** of file **stm3210e_eval.h**.

#define EVAL_COM1_CLK_ENABLE ( ) __HAL_RCC_USART1_CL

Definition at line **315** of file **stm3210e_eval.h**.

#define EVAL_COM1_IRQn USART1_IRQn

Definition at line **328** of file **stm3210e_eval.h**.

#define EVAL_COM1_RX_GPIO_CLK_DISABLE ( ) __HAL_RCC_C

Definition at line **326** of file **stm3210e_eval.h**.

#define EVAL_COM1_RX_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line 325 of file stm3210e_eval.h.

#define EVAL_COM1_RX_GPIO_PORT GPIOA

Definition at line 324 of file stm3210e_eval.h.

#define EVAL_COM1_RX_PIN GPIO_PIN_10 /* PA.10*/

Definition at line 323 of file stm3210e_eval.h.

#define EVAL_COM1_TX_GPIO_CLK_DISABLE ( ) __HAL_RCC_G

Definition at line 321 of file stm3210e_eval.h.

#define EVAL_COM1_TX_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line 320 of file stm3210e_eval.h.

#define EVAL_COM1_TX_GPIO_PORT GPIOA

Definition at line 319 of file stm3210e_eval.h.

#define EVAL_COM1_TX_PIN GPIO_PIN_9 /* PA.09*/

Definition at line 318 of file stm3210e_eval.h.

#define EVAL_COM2 USART2

Definition for COM port2, connected to USART2.

Definition at line **333** of file **stm3210e_eval.h**.

#define EVAL_COM2_CLK_DISABLE ( ) __HAL_RCC_USART2_CL

Definition at line **335** of file **stm3210e_eval.h**.

#define EVAL_COM2_CLK_ENABLE ( ) __HAL_RCC_USART2_CL

Definition at line **334** of file **stm3210e_eval.h**.

#define EVAL_COM2_IRQn USART2_IRQn

Definition at line **347** of file **stm3210e_eval.h**.

#define EVAL_COM2_RX_GPIO_CLK_DISABLE ( ) __HAL_RCC_G

Definition at line **345** of file **stm3210e_eval.h**.

#define EVAL_COM2_RX_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line **344** of file **stm3210e_eval.h**.

#define EVAL_COM2_RX_GPIO_PORT GPIOA

Definition at line **343** of file **stm3210e_eval.h**.

#define EVAL_COM2_RX_PIN GPIO_PIN_3 /* PA.03*/

Definition at line **342** of file **stm3210e_eval.h**.

#define EVAL_COM2_TX_GPIO_CLK_DISABLE ( ) __HAL_RCC_G

Definition at line **340** of file **stm3210e_eval.h**.

#define EVAL_COM2_TX_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line **339** of file **stm3210e_eval.h**.

#define EVAL_COM2_TX_GPIO_PORT GPIOA

Definition at line **338** of file **stm3210e_eval.h**.

#define EVAL_COM2_TX_PIN GPIO_PIN_2 /* PA.02*/

Definition at line **337** of file **stm3210e_eval.h**.

STM3210E_EVAL BSP User Manual

AUDIO_Exported_Macros

STM3210E_EVAL_AUDIO

Defines

#define	**DMA_MAX**(_X_) (((_X_) <= **DMA_MAX_SZE**)? (_X_):**DMA_MAX_SZE**)

Define Documentation

#define DMA_MAX (_X_) (((_X_) <= DMA_MAX_SZE)? (_X_):DMA

Definition at line **131** of file **stm3210e_eval_audio.h**.

Referenced by **BSP_AUDIO_OUT_Play()**, and **I2SOUT_SendDummyData_Start()**.

STM3210E_EVAL BSP User Manual

Defines

STM3210E_EVAL_BUS

Exported Constants

Defines

#define	**EVAL_I2Cx_SCL_PIN** GPIO_PIN_6 /* PB.06*/
#define	**EVAL_I2Cx_SCL_GPIO_PORT** GPIOB
#define	**EVAL_I2Cx_SDA_PIN** GPIO_PIN_7 /* PB.07*/
#define	**EVAL_I2Cx_SDA_GPIO_PORT** GPIOB
#define	**EVAL_I2Cx** I2C1
#define	**EVAL_I2Cx_CLK_ENABLE**() __HAL_RCC_I2C1_CLK_ENA
#define	**EVAL_I2Cx_SDA_GPIO_CLK_ENABLE**() __HAL_RCC_GPI
#define	**EVAL_I2Cx_SCL_GPIO_CLK_ENABLE**() __HAL_RCC_GPI
#define	**EVAL_I2Cx_FORCE_RESET**() __HAL_RCC_I2C1_FORCE_
#define	**EVAL_I2Cx_RELEASE_RESET**() __HAL_RCC_I2C1_RELE
#define	**EVAL_I2Cx_EV_IRQn** I2C1_EV_IRQn
#define	**EVAL_I2Cx_EV_IRQHandler** I2C1_EV_IRQHandler
#define	**EVAL_I2Cx_ER_IRQn** I2C1_ER_IRQn
#define	**EVAL_I2Cx_ER_IRQHandler** I2C1_ER_IRQHandler
#define	**EVAL_I2Cx_TIMEOUT_MAX** 3000
#define	**EVAL_SPIx** SPI1
#define	**EVAL_SPIx_CLK_ENABLE**() __HAL_RCC_SPI1_CLK_ENA
#define	**EVAL_SPIx_SCK_GPIO_PORT** GPIOA /* PA.05*/
#define	**EVAL_SPIx_SCK_PIN** GPIO_PIN_5
#define	**EVAL_SPIx_SCK_GPIO_CLK_ENABLE**() __HAL_RCC_GPI
#define	**EVAL_SPIx_SCK_GPIO_CLK_DISABLE**() __HAL_RCC_GP
#define	**EVAL_SPIx_MISO_MOSI_GPIO_PORT** GPIOA
#define	**EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE**() __HAL_R
#define	**EVAL_SPIx_MISO_MOSI_GPIO_CLK_DISABLE**() __HAL_R
#define	**EVAL_SPIx_MISO_PIN** GPIO_PIN_6 /* PA.06*/
#define	**EVAL_SPIx_MOSI_PIN** GPIO_PIN_7 /* PA.07*/
#define	**EVAL_SPIx_TIMEOUT_MAX** 1000

Define Documentation

#define EVAL_I2Cx I2C1

Definition at line 386 of file stm3210e_eval.h.

Referenced by I2Cx_Init(), and I2Cx_MspInit().

#define EVAL_I2Cx_CLK_ENABLE ( ) __HAL_RCC_I2C1_CLK_EN

Definition at line 387 of file stm3210e_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_ER_IRQHandler I2C1_ER_IRQHandler

Definition at line 398 of file stm3210e_eval.h.

#define EVAL_I2Cx_ER_IRQn I2C1_ER_IRQn

Definition at line 397 of file stm3210e_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_EV_IRQHandler I2C1_EV_IRQHandler

Definition at line 396 of file stm3210e_eval.h.

#define EVAL_I2Cx_EV_IRQn I2C1_EV_IRQn

Definition at line 395 of file stm3210e_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_FORCE_RESET ( ) __HAL_RCC_I2C1_FORCE

Definition at line 391 of file stm3210e_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_RELEASE_RESET ( ) __HAL_RCC_I2C1_REL

Definition at line 392 of file stm3210e_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SCL_GPIO_CLK_ENABLE ( ) __HAL_RCC_GI

Definition at line 389 of file stm3210e_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SCL_GPIO_PORT GPIOB

Definition at line 381 of file stm3210e_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SCL_PIN GPIO_PIN_6 /* PB.06*/

Definition at line 380 of file stm3210e_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SDA_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line 388 of file stm3210e_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SDA_GPIO_PORT GPIOB

Definition at line 383 of file stm3210e_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SDA_PIN GPIO_PIN_7 /* PB.07*/

Definition at line 382 of file stm3210e_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_TIMEOUT_MAX 3000

Definition at line 414 of file stm3210e_eval.h.

#define EVAL_SPIx SPI1

Definition at line 417 of file stm3210e_eval.h.

Referenced by SPIx_Init().

#define EVAL_SPIx_CLK_ENABLE ( ) __HAL_RCC_SPI1_CLK_EN

Definition at line 418 of file stm3210e_eval.h.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_MISO_MOSI_GPIO_CLK_DISABLE ( ) __HAL_

Definition at line **427** of file **stm3210e_eval.h**.

#define EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE ( ) __HAL_

Definition at line **426** of file **stm3210e_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_MISO_MOSI_GPIO_PORT GPIOA

Definition at line **425** of file **stm3210e_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_MISO_PIN GPIO_PIN_6 /* PA.06*/

Definition at line **428** of file **stm3210e_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_MOSI_PIN GPIO_PIN_7 /* PA.07*/

Definition at line **429** of file **stm3210e_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_SCK_GPIO_CLK_DISABLE ( ) __HAL_RCC_G

Definition at line **423** of file **stm3210e_eval.h**.

#define EVAL_SPIx_SCK_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line **422** of file **stm3210e_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_SCK_GPIO_PORT GPIOA /* PA.05*/

Definition at line **420** of file **stm3210e_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_SCK_PIN GPIO_PIN_5

Definition at line **421** of file **stm3210e_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_TIMEOUT_MAX 1000

Definition at line **435** of file **stm3210e_eval.h**.

STM3210E_EVAL BSP User Manual

Exported_Constants

STM3210E_EVAL Serial FLASH

Defines

#define **FLASH_OK** 0x00
#define **FLASH_ERROR** 0x01

Define Documentation

#define FLASH_ERROR 0x01

Definition at line 73 of file stm3210e_eval_serialflash.h.

Referenced by BSP_SERIAL_FLASH_EraseBulk(), BSP_SERIAL_FLASH_EraseSector(), BSP_SERIAL_FLASH_Init(), BSP_SERIAL_FLASH_ReadData(), BSP_SERIAL_FLASH_WriteData(), and BSP_SERIAL_FLASH_WritePage().

#define FLASH_OK 0x00

Definition at line 72 of file stm3210e_eval_serialflash.h.

Referenced by BSP_SERIAL_FLASH_EraseBulk(), BSP_SERIAL_FLASH_EraseSector(), BSP_SERIAL_FLASH_Init(), BSP_SERIAL_FLASH_ReadData(), BSP_SERIAL_FLASH_WriteData(), and BSP_SERIAL_FLASH_WritePage().

STM3210E_EVAL BSP User Manual

Functions

Bus Operations Functions

STM3210E-EVAL Common

Functions

static void	**FSMC_BANK1NORSRAM4_MspInit** (void) Initializes FSMC_BANK4 MSP.
static void	**FSMC_BANK1NORSRAM4_Init** (void) Initializes LCD IO.
static void	**FSMC_BANK1NORSRAM4_WriteData** (uint16_t Data) Writes register value.
static void	**FSMC_BANK1NORSRAM4_WriteReg** (uint8_t Reg) Writes register address.
static uint16_t	**FSMC_BANK1NORSRAM4_ReadData** (uint8_t Reg) Reads register value.
static void	**I2Cx_MspInit** (I2C_HandleTypeDef *hi2c) Eval I2Cx MSP Initialization.
static void	**I2Cx_Init** (void) Eval I2Cx Bus initialization.
static void	**I2Cx_WriteData** (uint16_t Addr, uint8_t Reg, uint8_t Value) Write a value in a register of the device through BUS.
static HAL_StatusTypeDef	**I2Cx_WriteBuffer** (uint16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuffer, uint16_t Length) Write a value in a register of the device through BUS.
static uint8_t	**I2Cx_ReadData** (uint16_t Addr, uint8_t Reg) Read a value in a register of the device through BUS.

static HAL_StatusTypeDef	**I2Cx_ReadBuffer** (uint16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuffer, uint16_t Length) Reads multiple data on the BUS.
static HAL_StatusTypeDef	**I2Cx_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.
static void	**I2Cx_Error** (void) Eval I2Cx error treatment function.
static void	**SPIx_MspInit** (SPI_HandleTypeDef *hspi) Initializes SPI MSP.
static HAL_StatusTypeDef	**SPIx_Init** (void) Initializes SPI HAL.
static uint8_t	**SPIx_Write** (uint8_t WriteValue) SPI Write a byte to device.
static uint8_t	**SPIx_Read** (void) SPI Read 1 byte from device.
static void	**SPIx_Error** (void) SPI error treatment function.

Function Documentation

static void FSMC_BANK1NORSRAM4_Init (void) [static]

Initializes LCD IO.

Return values:
None

Definition at line **597** of file **stm3210e_eval.c**.

References **FSMC_BANK1NORSRAM4_MspInit()**.

Referenced by **LCD_IO_Init()**.

static void FSMC_BANK1NORSRAM4_MspInit (void) [static]

Initializes FSMC_BANK4 MSP.

Return values:
None

Definition at line **552** of file **stm3210e_eval.c**.

Referenced by **FSMC_BANK1NORSRAM4_Init()**.

static uint16_t FSMC_BANK1NORSRAM4_ReadData (uint8_t Reg)

Reads register value.

Return values:
Read value

Definition at line **668** of file **stm3210e_eval.c**.

References **TFT_LCD**.

Referenced by **LCD_IO_ReadData()**.

static void FSMC_BANK1NORSRAM4_WriteData (uint16_t Data) [s

Writes register value.

Parameters:

Data,:

Return values:

None

Definition at line **647** of file **stm3210e_eval.c**.

References **TFT_LCD**.

Referenced by **LCD_IO_WriteData()**, and **LCD_IO_WriteMultipleData()**.

static void FSMC_BANK1NORSRAM4_WriteReg (uint8_t Reg) [sta

Writes register address.

Parameters:

Reg,:

Return values:

None

Definition at line **658** of file **stm3210e_eval.c**.

References **TFT_LCD**.

Referenced by **LCD_IO_WriteReg()**.

static void I2Cx_Error (void) [static]

Eval I2Cx error treatment function.

Return values:
None

Definition at line **865** of file **stm3210e_eval.c**.

References **heval_I2c**, and **I2Cx_Init()**.

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

static void I2Cx_Init (void) [static]

Eval I2Cx Bus initialization.

Return values:
None

Definition at line **736** of file **stm3210e_eval.c**.

References **EVAL_I2Cx**, **heval_I2c**, and **I2Cx_MspInit()**.

Referenced by **AUDIO_IO_Init()**, **I2Cx_Error()**, and **TSENSOR_IO_Init()**.

static HAL_StatusTypeDef I2Cx_IsDeviceReady (uint16_t DevAddr, uint32_t Trials) [static]

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 **855** of file **stm3210e_eval.c**.

References **heval_I2c**, and **I2cxTimeout**.

Referenced by **TSENSOR_IO_IsDeviceReady()**.

static void I2Cx_MspInit (I2C_HandleTypeDef * hi2c) [static]

Eval I2Cx MSP Initialization.

Parameters:

hi2c,: I2C handle

Return values:

None

Definition at line **685** of file **stm3210e_eval.c**.

References **EVAL_I2Cx**, **EVAL_I2Cx_CLK_ENABLE**, **EVAL_I2Cx_ER_IRQn**, **EVAL_I2Cx_EV_IRQn**, **EVAL_I2Cx_FORCE_RESET**, **EVAL_I2Cx_RELEASE_RESET**, **EVAL_I2Cx_SCL_GPIO_CLK_ENABLE**, **EVAL_I2Cx_SCL_GPIO_PORT**, **EVAL_I2Cx_SCL_PIN**, **EVAL_I2Cx_SDA_GPIO_CLK_ENABLE**, **EVAL_I2Cx_SDA_GPIO_PORT**, and **EVAL_I2Cx_SDA_PIN**.

Referenced by I2Cx_Init().

```
static HAL_StatusTypeDef I2Cx_ReadBuffer ( uint16_t  Addr,
                                           uint8_t   Reg,
                                           uint16_t  RegSize,
                                           uint8_t * pBuffer,
                                           uint16_t  Length
                                         )                    [static]
```

Reads multiple data on the BUS.

Parameters:

Addr,:	I2C Address
Reg,:	Reg Address
RegSize	: The target register size (can be 8BIT or 16BIT)
pBuffer,:	pointer to read data buffer
Length,:	length of the data

Return values:

0 if no problems to read multiple data

Definition at line **833** of file **stm3210e_eval.c**.

References **heval_I2c**, **I2Cx_Error()**, and **I2cxTimeout**.

Referenced by **TSENSOR_IO_Read()**.

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

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

Parameters:

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

Return values:
Data read at register @

Definition at line **807** of file **stm3210e_eval.c**.

References **heval_I2c**, **I2Cx_Error()**, and **I2cxTimeout**.

Referenced by **AUDIO_IO_Read()**.

static HAL_StatusTypeDef I2Cx_WriteBuffer (uint16_t Addr,
uint8_t Reg,
uint16_t RegSize,
uint8_t * pBuffer,
uint16_t Length
) [static]

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

Parameters:
Addr,: Device address on BUS Bus.
Reg,: The target register address to write
RegSize,: The target register size (can be 8BIT or 16BIT)
pBuffer,: The target register value to be written
Length,: buffer size to be written

Return values:
None

Definition at line **786** of file **stm3210e_eval.c**.

References **heval_I2c**, **I2Cx_Error()**, and **I2cxTimeout**.

Referenced by **TSENSOR_IO_Write()**.

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

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

Parameters:

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

Return values:

None

Definition at line **763** of file **stm3210e_eval.c**.

References **heval_I2c**, **I2Cx_Error()**, and **I2cxTimeout**.

Referenced by **AUDIO_IO_Write()**.

static void SPIx_Error (void) [static]

SPI error treatment function.

Return values:

None

Definition at line **977** of file **stm3210e_eval.c**.

References **heval_Spi**, and **SPIx_Init()**.

Referenced by **SPIx_Write()**.

HAL_StatusTypeDef SPIx_Init (void) [static]

Initializes SPI HAL.

Return values:
None

Definition at line 914 of file stm3210e_eval.c.

References EVAL_SPIx, heval_Spi, and SPIx_MspInit().

Referenced by FLASH_SPI_IO_Init(), and SPIx_Error().

static void SPIx_MspInit (SPI_HandleTypeDef * hspi) [static]

Initializes SPI MSP.

Return values:
None

Definition at line 882 of file stm3210e_eval.c.

References EVAL_SPIx_CLK_ENABLE, EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE, EVAL_SPIx_MISO_MOSI_GPIO_PORT, EVAL_SPIx_MISO_PIN, EVAL_SPIx_MOSI_PIN, EVAL_SPIx_SCK_GPIO_CLK_ENABLE, EVAL_SPIx_SCK_GPIO_PORT, and EVAL_SPIx_SCK_PIN.

Referenced by SPIx_Init().

static uint8_t SPIx_Read (void) [static]

SPI Read 1 byte from device.

Return values:

Read data

Definition at line **967** of file **stm3210e_eval.c**.

References **FLASH_SPI_DUMMY_BYTE**, and **SPIx_Write()**.

Referenced by **FLASH_SPI_IO_ReadByte()**.

static uint8_t SPIx_Write (uint8_t WriteValue) [static]

SPI Write a byte to device.

Parameters:

WriteValue to be written

Return values:

The value of the received byte.

Definition at line **945** of file **stm3210e_eval.c**.

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

Referenced by **FLASH_SPI_IO_ReadData()**, **FLASH_SPI_IO_ReadID()**, **FLASH_SPI_IO_WaitForWriteEnd()**, **FLASH_SPI_IO_WriteByte()**, **FLASH_SPI_IO_WriteEnable()**, and **SPIx_Read()**.

STM3210E_EVAL BSP User Manual

Private Functions

STM3210E-EVAL LCD

Functions

static void	**LCD_DrawPixel** (uint16_t Xpos, uint16_t Ypos, uint16_t RGBCode) Draws a pixel on LCD.
static void	**LCD_DrawChar** (uint16_t Xpos, uint16_t Ypos, const uint8_t *pChar) Draws a character on LCD.
static void	**LCD_SetDisplayWindow** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Sets display window.

Function Documentation

static void LCD_DrawChar (uint16_t Xpos,
uint16_t Ypos,
const uint8_t * pChar
) [static]

Draws a character on LCD.

Parameters:

Xpos,: Line where to display the character shape
Ypos,: Start column address
pChar,: Pointer to the character data

Return values:

None

Definition at line **1036** of file **stm3210e_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**, **bitmap**, **BSP_LCD_DrawBitmap()**, **OFFSET_BITMAP**, **LCD_DrawPropTypeDef::pFont**, and **LCD_DrawPropTypeDef::TextColor**.

Referenced by **BSP_LCD_DisplayChar()**.

static void LCD_DrawPixel (uint16_t Xpos,
uint16_t Ypos,
uint16_t RGBCode
) [static]

Draws a pixel on LCD.

Parameters:

Xpos,: X position
Ypos,: Y position
RGBCode,: Pixel color in RGB mode (5-6-5)

Return values:
None

Definition at line **1014** of file **stm3210e_eval_lcd.c**.

References **lcd_drv**, and **LCD_SwapXY**.

Referenced by **BSP_LCD_DrawCircle()**, **BSP_LCD_DrawEllipse()**, **BSP_LCD_DrawHLine()**, **BSP_LCD_DrawLine()**, and **BSP_LCD_DrawVLine()**.

---

static void LCD_SetDisplayWindow (uint16_t Xpos,
uint16_t Ypos,
uint16_t Width,
uint16_t Height
) [static]

Sets display window.

Parameters:
Xpos,: LCD X position
Ypos,: LCD Y position
Width,: LCD window width
Height,: LCD window height

Return values:
None

Definition at line **1107** of file **stm3210e_eval_lcd.c**.

References **lcd_drv**.

Referenced by **BSP_LCD_DrawBitmap()**, and **BSP_LCD_DrawVLine()**.

STM3210E_EVAL BSP User Manual

STM3210E_EVAL_LED

Exported Constants

Define for STM3210E_EVAL board. More...

Defines

```
#define LEDn 4
#define LED1_PIN GPIO_PIN_6 /* PF.06*/
#define LED1_GPIO_PORT GPIOF
#define LED1_GPIO_CLK_ENABLE() __HAL_RCC_GPIOF_CLK_EI
#define LED1_GPIO_CLK_DISABLE() __HAL_RCC_GPIOF_CLK_D
#define LED2_PIN GPIO_PIN_7 /* PF.07*/
#define LED2_GPIO_PORT GPIOF
#define LED2_GPIO_CLK_ENABLE() __HAL_RCC_GPIOF_CLK_EI
#define LED2_GPIO_CLK_DISABLE() __HAL_RCC_GPIOF_CLK_D
#define LED3_PIN GPIO_PIN_8 /* PF.08*/
#define LED3_GPIO_PORT GPIOF
#define LED3_GPIO_CLK_ENABLE() __HAL_RCC_GPIOF_CLK_EI
#define LED3_GPIO_CLK_DISABLE() __HAL_RCC_GPIOF_CLK_D
#define LED4_PIN GPIO_PIN_9 /* PF.09*/
#define LED4_GPIO_PORT GPIOF
#define LED4_GPIO_CLK_ENABLE() __HAL_RCC_GPIOF_CLK_EI
#define LED4_GPIO_CLK_DISABLE() __HAL_RCC_GPIOF_CLK_D
#define LEDx_GPIO_CLK_ENABLE(__LED__)
#define LEDx_GPIO_CLK_DISABLE(__LED__)
```

Detailed Description

Define for STM3210E_EVAL board.

Define Documentation

#define LED1_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOF_CLK_

Definition at line 161 of file stm3210e_eval.h.

#define LED1_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOF_CLK_

Definition at line 160 of file stm3210e_eval.h.

#define LED1_GPIO_PORT GPIOF

Definition at line 159 of file stm3210e_eval.h.

#define LED1_PIN GPIO_PIN_6 /* PF.06*/

Definition at line 158 of file stm3210e_eval.h.

#define LED2_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOF_CLK_

Definition at line 166 of file stm3210e_eval.h.

#define LED2_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOF_CLK_

Definition at line 165 of file stm3210e_eval.h.

#define LED2_GPIO_PORT GPIOF

Definition at line 164 of file stm3210e_eval.h.

#define LED2_PIN GPIO_PIN_7 /* PF.07*/

Definition at line 163 of file stm3210e_eval.h.

#define LED3_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOF_CLK_

Definition at line 172 of file stm3210e_eval.h.

#define LED3_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOF_CLK_

Definition at line 171 of file stm3210e_eval.h.

#define LED3_GPIO_PORT GPIOF

Definition at line 170 of file stm3210e_eval.h.

#define LED3_PIN GPIO_PIN_8 /* PF.08*/

Definition at line 169 of file stm3210e_eval.h.

#define LED4_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOF_CLK_

Definition at line 178 of file stm3210e_eval.h.

#define LED4_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOF_CLK_

Definition at line 177 of file stm3210e_eval.h.

#define LED4_GPIO_PORT GPIOF

Definition at line 176 of file stm3210e_eval.h.

#define LED4_PIN GPIO_PIN_9 /* PF.09*/

Definition at line 175 of file stm3210e_eval.h.

#define LEDn 4

Definition at line 156 of file stm3210e_eval.h.

#define LEDx_GPIO_CLK_DISABLE (__LED__)

Value:

```
(((__LED__) == LED1) ? LED1_GPIO_CLK_DISABLE() :\
                                                ((__LED
__) == LED2) ? LED2_GPIO_CLK_DISABLE() :\
                                                ((__LED
__) == LED3) ? LED3_GPIO_CLK_DISABLE() :\
                                                ((__LED
__) == LED4) ? LED4_GPIO_CLK_DISABLE() : 0 )
```

Definition at line 185 of file stm3210e_eval.h.

#define LEDx_GPIO_CLK_ENABLE (__LED__)

Value:

```
do { if ((__LED__) == LED1) LED1_GPIO_CLK_ENABLE(
); else\
                                                  if ((_
_LED__) == LED2) LED2_GPIO_CLK_ENABLE(); else \
                                                  if ((_
```

```
_LED__) == LED3) LED3_GPIO_CLK_ENABLE(); else\
                                           if ((_
_LED__) == LED4) LED4_GPIO_CLK_ENABLE();} while(0
)
```

Definition at line **180** of file **stm3210e_eval.h**.

Referenced by **BSP_LED_Init()**.

STM3210E_EVAL BSP User Manual

Private Defines

STM3210E-EVAL LCD

Defines

#define **POLY_X**(Z) ((int32_t)((pPoints + (Z))->X))
#define **POLY_Y**(Z) ((int32_t)((pPoints + (Z))->Y))
#define **MAX_HEIGHT_FONT** 17
#define **MAX_WIDTH_FONT** 24
#define **OFFSET_BITMAP** 54

Define Documentation

#define MAX_HEIGHT_FONT 17

Definition at line 93 of file stm3210e_eval_lcd.c.

#define MAX_WIDTH_FONT 24

Definition at line 94 of file stm3210e_eval_lcd.c.

#define OFFSET_BITMAP 54

Definition at line 95 of file stm3210e_eval_lcd.c.

Referenced by LCD_DrawChar().

#define POLY_X (Z) ((int32_t)((pPoints + (Z))->X))

Definition at line 90 of file stm3210e_eval_lcd.c.

Referenced by BSP_LCD_FillPolygon().

#define POLY_Y (Z) ((int32_t)((pPoints + (Z))->Y))

Definition at line 91 of file stm3210e_eval_lcd.c.

Referenced by BSP_LCD_FillPolygon().

STM3210E_EVAL BSP User Manual

Defines

Exported_Types

STM3210E-EVAL SD

Defines

#define	**SD_CardInfo** HAL_SD_CardInfoTypedef SD Card information structure.
#define	**MSD_OK** 0x00 SD status structure definition.
#define	**MSD_ERROR** 0x01

Define Documentation

#define MSD_ERROR 0x01

Definition at line 76 of file stm3210e_eval_sd.h.

Referenced by BSP_SD_Erase(), BSP_SD_Init(), BSP_SD_ReadBlocks(), BSP_SD_ReadBlocks_DMA(), BSP_SD_WriteBlocks(), and BSP_SD_WriteBlocks_DMA().

#define MSD_OK 0x00

SD status structure definition.

Definition at line 75 of file stm3210e_eval_sd.h.

Referenced by BSP_SD_Erase(), BSP_SD_Init(), BSP_SD_ReadBlocks(), BSP_SD_ReadBlocks_DMA(), BSP_SD_WriteBlocks(), and BSP_SD_WriteBlocks_DMA().

#define SD_CardInfo HAL_SD_CardInfoTypedef

SD Card information structure.

Definition at line 70 of file stm3210e_eval_sd.h.

STM3210E_EVAL BSP User Manual

Exported_Constants

STM3210E_EVAL NAND

Defines

#define	**NAND_OK** 0x00 NAND status structure definition.
#define	**NAND_DEVICE_ADDR** ((uint32_t)NAND_DEVICE1)
#define	**NAND_PAGE_SIZE** ((uint16_t)0x0200) /* 512 bytes per page w/o Spare Area */ FSMC NAND memory parameters.
#define	**NAND_BLOCK_SIZE** ((uint16_t)0x0020) /* 32x512 bytes pages per block */
#define	**NAND_ZONE_SIZE** ((uint16_t)0x0400) /* 1024 Block per zone */
#define	**NAND_SPARE_AREA_SIZE** ((uint16_t)0x0010) /* last 16 bytes as spare area */
#define	**NAND_MAX_ZONE** ((uint16_t)0x0004) /* 4 zones of 1024 block */

Define Documentation

#define NAND_BLOCK_SIZE ((uint16_t)0x0020) /* 32x512 bytes pa

Definition at line **89** of file **stm3210e_eval_nand.h**.

Referenced by **BSP_NAND_Init()**.

#define NAND_DEVICE_ADDR ((uint32_t)NAND_DEVICE1)

Definition at line **83** of file **stm3210e_eval_nand.h**.

#define NAND_MAX_ZONE ((uint16_t)0x0004) /* 4 zones of 1024 b

Definition at line **92** of file **stm3210e_eval_nand.h**.

Referenced by **BSP_NAND_Init()**.

#define NAND_OK 0x00

NAND status structure definition.

Definition at line **81** of file **stm3210e_eval_nand.h**.

Referenced by **BSP_NAND_Erase_Block()**, **BSP_NAND_Init()**, **BSP_NAND_Read_ID()**, **BSP_NAND_ReadData()**, and **BSP_NAND_WriteData()**.

#define NAND_PAGE_SIZE ((uint16_t)0x0200) /* 512 bytes per pag

FSMC NAND memory parameters.

Definition at line **88** of file **stm3210e_eval_nand.h**.

Referenced by **BSP_NAND_Init()**.

#define NAND_SPARE_AREA_SIZE ((uint16_t)0x0010) /* last 16 by

Definition at line **91** of file **stm3210e_eval_nand.h**.

Referenced by **BSP_NAND_Init()**.

#define NAND_ZONE_SIZE ((uint16_t)0x0400) /* 1024 Block per zo

Definition at line **90** of file **stm3210e_eval_nand.h**.

Referenced by **BSP_NAND_Init()**.

STM3210E_EVAL BSP User Manual

Private_Variables

STM3210E_EVAL NAND

Variables

static NAND_HandleTypeDef	**nandHandle**
static FSMC_NAND_PCC_TimingTypeDef	**Timing**

Variable Documentation

NAND_HandleTypeDef nandHandle [static]

Definition at line 118 of file stm3210e_eval_nand.c.

Referenced by BSP_NAND_Erase_Block(), BSP_NAND_Init(), BSP_NAND_Read_ID(), BSP_NAND_ReadData(), and BSP_NAND_WriteData().

FSMC_NAND_PCC_TimingTypeDef Timing [static]

Definition at line 119 of file stm3210e_eval_nand.c.

Referenced by BSP_NAND_Init().

STM3210E_EVAL BSP User Manual

Private_Variables

STM3210E_EVAL NOR

Variables

static NOR_HandleTypeDef	**norHandle**
static FSMC_NORSRAM_TimingTypeDef	**Timing**

Variable Documentation

NOR_HandleTypeDef norHandle [static]

Definition at line 122 of file stm3210e_eval_nor.c.

Referenced by BSP_NOR_Erase_Block(), BSP_NOR_Erase_Chip(), BSP_NOR_Init(), BSP_NOR_ProgramData(), BSP_NOR_Read_ID(), BSP_NOR_ReadData(), BSP_NOR_ReturnToReadMode(), and BSP_NOR_WriteData().

FSMC_NORSRAM_TimingTypeDef Timing [static]

Definition at line 123 of file stm3210e_eval_nor.c.

Referenced by BSP_NOR_Init().

STM3210E_EVAL BSP User Manual

Private Functions

STM3210E-EVAL SD

Functions

static void	**SD_MspInit** (void) Initializes the SD MSP.
HAL_SD_ErrorTypedef	**SD_DMAConfigRx** (SD_HandleTypeDef *hsd) SD_DMAConfigRx.
HAL_SD_ErrorTypedef	**SD_DMAConfigTx** (SD_HandleTypeDef *hsd) SD_DMAConfigTx.

Function Documentation

HAL_SD_ErrorTypedef SD_DMAConfigRx (SD_HandleTypeDef * hs

SD_DMAConfigRx.

Function Description

This function configure the DMA to receive data from the SD card

Return values:

SD_ERROR or SD_OK

Definition at line **465** of file **stm3210e_eval_sd.c**.

References **SD_DMAx_Rx_INSTANCE**, and **SD_DMAx_Rx_IRQn**.

Referenced by **BSP_SD_ReadBlocks_DMA()**.

HAL_SD_ErrorTypedef SD_DMAConfigTx (SD_HandleTypeDef * hs

SD_DMAConfigTx.

Function Description

This function configure the DMA to transmit data to the SD card

Return values:

SD_ERROR or SD_OK

Definition at line **513** of file **stm3210e_eval_sd.c**.

References **SD_DMAx_Tx_INSTANCE**, and **SD_DMAx_Tx_IRQn**.

Referenced by **BSP_SD_WriteBlocks_DMA()**.

static void SD_MspInit (void) [static]

Initializes the SD MSP.

Return values:

None

Definition at line **415** of file **stm3210e_eval_sd.c**.

References **__DMAx_TxRx_CLK_ENABLE**, **__SD_DETECT_GPIO_CLK_ENABLE**, **SD_DETECT_GPIO_PORT**, and **SD_DETECT_PIN**.

Referenced by **BSP_SD_Init()**.

STM3210E_EVAL BSP User Manual

Private_Variables

STM3210E_EVAL SRAM

Variables

SRAM_HandleTypeDef	**sramHandle**
static FSMC_NORSRAM_TimingTypeDef	**Timing**

Variable Documentation

SRAM_HandleTypeDef sramHandle

Definition at line **122** of file **stm3210e_eval_sram.c**.

Referenced by **BSP_SRAM_DMA_IRQHandler()**, **BSP_SRAM_Init()**, **BSP_SRAM_ReadData()**, **BSP_SRAM_ReadData_DMA()**, **BSP_SRAM_WriteData()**, **BSP_SRAM_WriteData_DMA()**, and **SRAM_MspInit()**.

FSMC_NORSRAM_TimingTypeDef Timing [static]

Definition at line **123** of file **stm3210e_eval_sram.c**.

Referenced by **BSP_SRAM_Init()**.

STM3210E_EVAL BSP User Manual

Private Variables

STM3210E_EVAL TSENSOR

Variables

static TSENSOR_DrvTypeDef *	**tsensor_drv**
__IO uint16_t	**TSENSORAddress** = 0

Variable Documentation

TSENSOR_DrvTypeDef* tsensor_drv [static]

Definition at line 84 of file stm3210e_eval_tsensor.c.

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

__IO uint16_t TSENSORAddress = 0

Definition at line 85 of file stm3210e_eval_tsensor.c.

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

STM3210E_EVAL BSP User Manual

Enumerations

Exported_Types

STM3210E_EVAL TSENSOR

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 **70** of file **stm3210e_eval_tsensor.h**.

STM3210E_EVAL BSP User Manual

Exported_Constants

STM3210E_EVAL TSENSOR

Defines

#define	**TSENSOR_I2C_ADDRESS_A01**	0x90
#define	**TSENSOR_I2C_ADDRESS_A02**	0x92
#define	**TSENSOR_MAX_TRIALS**	50

Define Documentation

#define TSENSOR_I2C_ADDRESS_A01 0x90

Definition at line 84 of file stm3210e_eval_tsensor.h.

Referenced by BSP_TSENSOR_Init().

#define TSENSOR_I2C_ADDRESS_A02 0x92

Definition at line 85 of file stm3210e_eval_tsensor.h.

Referenced by BSP_TSENSOR_Init().

#define TSENSOR_MAX_TRIALS 50

Definition at line 88 of file stm3210e_eval_tsensor.h.

Referenced by BSP_TSENSOR_Init().

STM3210E_EVAL BSP User Manual

Private Variables

STM3210E-EVAL SD

Variables

SD_HandleTypeDef	**uSdHandle**
static **SD_CardInfo**	**uSdCardInfo**

Variable Documentation

SD_CardInfo uSdCardInfo [static]

Definition at line **98** of file **stm3210e_eval_sd.c**.

Referenced by **BSP_SD_Init()**.

SD_HandleTypeDef uSdHandle

Definition at line **97** of file **stm3210e_eval_sd.c**.

Referenced by **BSP_SD_DMA_Rx_IRQHandler()**, **BSP_SD_DMA_Tx_IRQHandler()**, **BSP_SD_Erase()**, **BSP_SD_GetCardInfo()**, **BSP_SD_GetStatus()**, **BSP_SD_Init()**, **BSP_SD_IRQHandler()**, **BSP_SD_ReadBlocks()**, **BSP_SD_ReadBlocks_DMA()**, **BSP_SD_WriteBlocks()**, and **BSP_SD_WriteBlocks_DMA()**.

STM3210E_EVAL BSP User Manual

Drivers Directory Reference

Directories

directory	**BSP**

STM3210E_EVAL BSP User Manual

BSP Directory Reference

Directories

directory	**STM3210E_EVAL**

STM3210E_EVAL BSP User Manual

STM3210E_EVAL Directory Reference

Files

file **stm3210e_eval.c** [code]

This file provides a set of firmware functions to manage Leds, push-button and COM ports for STM3210E_EVAL.

file **stm3210e_eval.h** [code]

This file contains definitions for STM3210E_EVAL's LEDs, push-buttons and COM ports hardware resources.

file **stm3210e_eval_audio.c** [code]

This file provides the Audio driver for the STM3210E-Eval board.

file **stm3210e_eval_audio.h** [code]

This file contains the common defines and functions prototypes for **stm3210e_eval_audio.c** driver.

file **stm3210e_eval_lcd.c** [code]

This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM3210E-EVAL evaluation board.

file **stm3210e_eval_lcd.h** [code]

This file contains the common defines and functions prototypes for the **stm3210e_eval_lcd.c** driver.

file **stm3210e_eval_nand.c** [code]

This file includes a standard driver for the NAND512W3A2CN6E NAND flash memory device mounted on STM3210E-EVAL evaluation board.

file **stm3210e_eval_nand.h** [code]

This file contains the common defines and functions prototypes for the **stm3210e_eval_nand.c** driver.

file **stm3210e_eval_nor.c** [code]

This file includes a standard driver for the M29W128GL70ZA6E NOR flash memory device mounted on STM3210E-EVAL evaluation board.

file **stm3210e_eval_nor.h** [code]

This file contains the common defines and functions prototypes for the **stm3210e_eval_nor.c** driver.

file **stm3210e_eval_sd.c** [code]

This file includes the uSD card driver.

file **stm3210e_eval_sd.h** [code]

file **stm3210e_eval_serialflash.c** [code]

This file provides a set of functions needed to manage the SPI M25Pxx FLASH memory mounted on STM3210E_EVAL board. It implements a high level communication layer for read and write from/to this memory. The needed STM32 hardware

resources (SPI and GPIO) are defined in **stm3210e_eval.h** file, and the initialization is performed in **FLASH_SPI_IO_Init()** function declared in **stm3210e_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **FLASH_SPI_IO_Init()** function.

file **stm3210e_eval_serialflash.h** [code]

This file contains the common defines and functions prototypes for the stm3210e_eval_flash driver.

file **stm3210e_eval_sram.c** [code]

This file includes the SRAM driver for the IS61WV51216BLL-10M memory device mounted on STM3210E-EVAL evaluation board.

file **stm3210e_eval_sram.h** [code]

This file contains the common defines and functions prototypes for the **stm3210e_eval_sram.c** driver.

file **stm3210e_eval_tsensor.c** [code]

This file provides a set of functions needed to manage the I2C STLM75 temperature sensor mounted on STM3210E_EVAL board . It implements a high level communication layer for read and write from/to this sensor. The needed STM32L152xD hardware resources (I2C and GPIO) are defined in **stm3210e_eval.h** file, and the initialization is performed in **TSENSOR_IO_Init()** function declared in **stm3210e_eval.c** file. You can easily tailor this driver to any other development board,

by just adapting the defines for hardware resources and **TSENSOR_IO_Init()** function.

file **stm3210e_eval_tsensor.h** [code]

This file contains all the functions prototypes for the stm3210e_tsensor.c firmware driver.

STM3210E_EVAL BSP User Manual

stm3210e_eval.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210e_eval.h
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file contains definitions for STM3210E_EVAL's LEDs,
00008   *          push-buttons and COM ports hardware resources.
00009   ******************************************************************************
00010   * @attention
00011   *
00012   * <h2><center>&copy; COPYRIGHT(c) 2015 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 __STM3210E_EVAL_H
00041 #define __STM3210E_EVAL_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /** @addtogroup BSP
00048   * @{
00049   */
00050
00051 /** @addtogroup STM3210E_EVAL
00052   * @{
00053   */
00054
00055 /* Includes ------------------------------------------------------------------*/
00056 #include "stm32f1xx_hal.h"
00057
00058 /** @addtogroup STM3210E_EVAL_Common STM3210E-EVAL Common
00059   * @{
00060   */
00061
00062 /** @defgroup STM3210E_EVAL_Exported_Types Exported Types
00063   * @{
00064   */
00065
00066 /**
```

```
00067  * @brief LED Types Definition
00068  */
00069 typedef enum
00070 {
00071   LED1 = 0,
00072   LED2 = 1,
00073   LED3 = 2,
00074   LED4 = 3,
00075
00076   LED_GREEN  = LED1,
00077   LED_ORANGE = LED2,
00078   LED_RED    = LED3,
00079   LED_BLUE   = LED4
00080
00081 } Led_TypeDef;
00082
00083 /**
00084  * @brief BUTTON Types Definition
00085  */
00086 typedef enum
00087 {
00088   BUTTON_WAKEUP = 0,
00089   BUTTON_TAMPER = 1,
00090   BUTTON_KEY    = 2,
00091   BUTTON_SEL    = 3,
00092   BUTTON_LEFT   = 4,
00093   BUTTON_RIGHT  = 5,
00094   BUTTON_DOWN   = 6,
00095   BUTTON_UP     = 7,
00096
00097 } Button_TypeDef;
00098
00099 typedef enum
00100 {
00101   BUTTON_MODE_GPIO = 0,
00102   BUTTON_MODE_EXTI = 1,
00103   BUTTON_MODE_EVT  = 2
```

```
00104
00105 } ButtonMode_TypeDef;
00106
00107 /**
00108  * @brief JOYSTICK Types Definition
00109  */
00110 typedef enum
00111 {
00112   JOY_SEL   = 0,
00113   JOY_LEFT  = 1,
00114   JOY_RIGHT = 2,
00115   JOY_DOWN  = 3,
00116   JOY_UP    = 4,
00117   JOY_NONE  = 5
00118
00119 }JOYState_TypeDef;
00120
00121 typedef enum
00122 {
00123   JOY_MODE_GPIO = 0,
00124   JOY_MODE_EXTI = 1
00125
00126 }JOYMode_TypeDef;
00127
00128 /**
00129  * @brief COM Types Definition
00130  */
00131 typedef enum
00132 {
00133   COM1 = 0,
00134   COM2 = 1
00135
00136 } COM_TypeDef;
00137
00138 /**
00139   * @}
00140   */
```

```
00141
00142 /** @defgroup STM3210E_EVAL_Exported_Constan
ts Exported Constants
00143   * @{
00144   */
00145
00146 /**
00147   * @brief  Define for STM3210E_EVAL board
00148   */
00149 #if !defined (USE_STM3210E_EVAL)
00150  #define USE_STM3210E_EVAL
00151 #endif
00152
00153 /** @addtogroup STM3210E_EVAL_LED
00154   * @{
00155   */
00156 #define LEDn                                4
00157
00158 #define LED1_PIN                            GPI
O_PIN_6             /* PF.06*/
00159 #define LED1_GPIO_PORT                      GPI
OF
00160 #define LED1_GPIO_CLK_ENABLE()              __H
AL_RCC_GPIOF_CLK_ENABLE()
00161 #define LED1_GPIO_CLK_DISABLE()             __H
AL_RCC_GPIOF_CLK_DISABLE()
00162
00163 #define LED2_PIN                            GPI
O_PIN_7             /* PF.07*/
00164 #define LED2_GPIO_PORT                      GPI
OF
00165 #define LED2_GPIO_CLK_ENABLE()              __H
AL_RCC_GPIOF_CLK_ENABLE()
00166 #define LED2_GPIO_CLK_DISABLE()             __H
AL_RCC_GPIOF_CLK_DISABLE()
00167
00168
```

```
00169 #define LED3_PIN                                GPI
O_PIN_8            /* PF.08*/
00170 #define LED3_GPIO_PORT                          GPI
OF
00171 #define LED3_GPIO_CLK_ENABLE()                  __H
AL_RCC_GPIOF_CLK_ENABLE()
00172 #define LED3_GPIO_CLK_DISABLE()                 __H
AL_RCC_GPIOF_CLK_DISABLE()
00173
00174
00175 #define LED4_PIN                                GPI
O_PIN_9            /* PF.09*/
00176 #define LED4_GPIO_PORT                          GPI
OF
00177 #define LED4_GPIO_CLK_ENABLE()                  __H
AL_RCC_GPIOF_CLK_ENABLE()
00178 #define LED4_GPIO_CLK_DISABLE()                 __H
AL_RCC_GPIOF_CLK_DISABLE()
00179
00180 #define LEDx_GPIO_CLK_ENABLE(__LED__) do { i
f ((__LED__) == LED1) LED1_GPIO_CLK_ENABLE(); else\

00181                                                 i
f ((__LED__) == LED2) LED2_GPIO_CLK_ENABLE(); else
 \
00182                                                 i
f ((__LED__) == LED3) LED3_GPIO_CLK_ENABLE(); else\

00183                                                 i
f ((__LED__) == LED4) LED4_GPIO_CLK_ENABLE();} whi
le(0)
00184
00185 #define LEDx_GPIO_CLK_DISABLE(__LED__)   (((
__LED__) == LED1) ? LED1_GPIO_CLK_DISABLE() :\
00186                                                 ((
__LED__) == LED2) ? LED2_GPIO_CLK_DISABLE() :\
00187                                                 ((
```

```
__LED__) == LED3) ? LED3_GPIO_CLK_DISABLE() :\
00188                                          ((
__LED__) == LED4) ? LED4_GPIO_CLK_DISABLE() : 0 )
00189
00190 /**
00191   * @}
00192   */
00193
00194 /** @addtogroup STM3210E_EVAL_BUTTON
00195   * @{
00196   */
00197 #define JOYn                                5
00198 #define BUTTONn                             3 +
 JOYn
00199
00200 /**
00201  * @brief Tamper push-button
00202  */
00203 #define TAMPER_BUTTON_PIN
GPIO_PIN_13             /* PC.13*/
00204 #define TAMPER_BUTTON_GPIO_PORT
GPIOC
00205 #define TAMPER_BUTTON_GPIO_CLK_ENABLE()
__HAL_RCC_GPIOC_CLK_ENABLE()
00206 #define TAMPER_BUTTON_GPIO_CLK_DISABLE()
__HAL_RCC_GPIOC_CLK_DISABLE()
00207 #define TAMPER_BUTTON_EXTI_IRQn
EXTI15_10_IRQn
00208
00209 /**
00210  * @brief Key push-button
00211  */
00212 #define KEY_BUTTON_PIN
GPIO_PIN_8              /* PG.08*/
00213 #define KEY_BUTTON_GPIO_PORT
GPIOG
00214 #define KEY_BUTTON_GPIO_CLK_ENABLE()
```

```
__HAL_RCC_GPIOG_CLK_ENABLE()
00215 #define KEY_BUTTON_GPIO_CLK_DISABLE()
__HAL_RCC_GPIOG_CLK_DISABLE()
00216 #define KEY_BUTTON_EXTI_IRQn
EXTI9_5_IRQn
00217
00218 /**
00219  * @brief Wake-up push-button
00220  */
00221 #define WAKEUP_BUTTON_PIN
GPIO_PIN_0                 /* PA.00*/
00222 #define WAKEUP_BUTTON_GPIO_PORT
GPIOA
00223 #define WAKEUP_BUTTON_GPIO_CLK_ENABLE()
__HAL_RCC_GPIOA_CLK_ENABLE()
00224 #define WAKEUP_BUTTON_GPIO_CLK_DISABLE()
__HAL_RCC_GPIOA_CLK_DISABLE()
00225 #define WAKEUP_BUTTON_EXTI_IRQn
EXTI0_IRQn
00226
00227 /**
00228  * @brief Joystick Right push-button
00229  */
00230 #define RIGHT_JOY_PIN
GPIO_PIN_13                /* PG.13*/
00231 #define RIGHT_JOY_GPIO_PORT
GPIOG
00232 #define RIGHT_JOY_GPIO_CLK_ENABLE()
__HAL_RCC_GPIOG_CLK_ENABLE()
00233 #define RIGHT_JOY_GPIO_CLK_DISABLE()
__HAL_RCC_GPIOG_CLK_DISABLE()
00234 #define RIGHT_JOY_EXTI_IRQn
EXTI15_10_IRQn
00235
00236 /**
00237  * @brief Joystick Left push-button
00238  */
```

```
00239 #define LEFT_JOY_PIN
GPIO_PIN_14              /* PG.14*/
00240 #define LEFT_JOY_GPIO_PORT
GPIOG
00241 #define LEFT_JOY_GPIO_CLK_ENABLE()
__HAL_RCC_GPIOG_CLK_ENABLE()
00242 #define LEFT_JOY_GPIO_CLK_DISABLE()
__HAL_RCC_GPIOG_CLK_DISABLE()
00243 #define LEFT_JOY_EXTI_IRQn
EXTI15_10_IRQn
00244
00245 /**
00246  * @brief Joystick Up push-button
00247  */
00248 #define UP_JOY_PIN
GPIO_PIN_15              /* PG.15*/
00249 #define UP_JOY_GPIO_PORT
GPIOG
00250 #define UP_JOY_GPIO_CLK_ENABLE()
__HAL_RCC_GPIOG_CLK_ENABLE()
00251 #define UP_JOY_GPIO_CLK_DISABLE()
__HAL_RCC_GPIOG_CLK_DISABLE()
00252 #define UP_JOY_EXTI_IRQn
EXTI15_10_IRQn
00253
00254 /**
00255  * @brief Joystick Down push-button
00256  */
00257 #define DOWN_JOY_PIN
GPIO_PIN_3              /* PD.03*/
00258 #define DOWN_JOY_GPIO_PORT
GPIOD
00259 #define DOWN_JOY_GPIO_CLK_ENABLE()
__HAL_RCC_GPIOD_CLK_ENABLE()
00260 #define DOWN_JOY_GPIO_CLK_DISABLE()
__HAL_RCC_GPIOD_CLK_DISABLE()
00261 #define DOWN_JOY_EXTI_IRQn
```

```
EXTI3_IRQn
00262
00263 /**
00264  * @brief Joystick Sel push-button
00265  */
00266 #define SEL_JOY_PIN
GPIO_PIN_7              /* PG.07*/
00267 #define SEL_JOY_GPIO_PORT
GPIOG
00268 #define SEL_JOY_GPIO_CLK_ENABLE()
__HAL_RCC_GPIOG_CLK_ENABLE()
00269 #define SEL_JOY_GPIO_CLK_DISABLE()
__HAL_RCC_GPIOG_CLK_DISABLE()
00270 #define SEL_JOY_EXTI_IRQn
EXTI9_5_IRQn
00271
00272 #define BUTTONx_GPIO_CLK_ENABLE(__BUTTON__)
do { if ((__BUTTON__) == BUTTON_TAMPER) TAMPER_BUT
TON_GPIO_CLK_ENABLE(); else\
00273
     if ((__BUTTON__) == BUTTON_KEY) KEY_BUTTON_GP
IO_CLK_ENABLE(); else\
00274
     if ((__BUTTON__) == BUTTON_WAKEUP) WAKEUP_BUT
TON_GPIO_CLK_ENABLE(); else\
00275
     if ((__BUTTON__) == BUTTON_SEL) SEL_JOY_GPIO_
CLK_ENABLE(); else\
00276
     if ((__BUTTON__) == BUTTON_LEFT) LEFT_JOY_GPI
O_CLK_ENABLE(); else\
00277
     if ((__BUTTON__) == BUTTON_RIGHT) RIGHT_JOY_G
PIO_CLK_ENABLE(); else\
00278
     if ((__BUTTON__) == BUTTON_DOWN) DOWN_JOY_GPI
O_CLK_ENABLE(); else\
```

```
00279
     if ((__BUTTON__) == BUTTON_UP) UP_JOY_GPIO_CL
K_ENABLE();} while(0)
00280
00281 #define BUTTONx_GPIO_CLK_DISABLE(__BUTTON__)
    (((__BUTTON__) == BUTTON_TAMPER) ? TAMPER_BUTT
ON_GPIO_CLK_DISABLE()  :\
00282
     ((__BUTTON__) == BUTTON_KEY) ? KEY_BUTTON_GPI
O_CLK_DISABLE()  :\
00283
     ((__BUTTON__) == BUTTON_WAKEUP) ? WAKEUP_BUTT
ON_GPIO_CLK_DISABLE()  :\
00284
     ((__BUTTON__) == BUTTON_SEL) ? SEL_JOY_GPIO_C
LK_DISABLE()  :\
00285
     ((__BUTTON__) == BUTTON_LEFT) ? LEFT_JOY_GPIO
_CLK_DISABLE()  :\
00286
     ((__BUTTON__) == BUTTON_RIGHT) ? RIGHT_JOY_GP
IO_CLK_DISABLE()  :\
00287
     ((__BUTTON__) == BUTTON_DOWN) ? DOWN_JOY_GPIO
_CLK_DISABLE()  :\
00288
     ((__BUTTON__) == BUTTON_UP) ? UP_JOY_GPIO_CLK
_DISABLE() : 0 )
00289
00290 #define JOYx_GPIO_CLK_ENABLE(__JOY__)    do
{ if ((__JOY__) == JOY_SEL) SEL_JOY_GPIO_CLK_ENABL
E(); else\
00291
  if ((__JOY__) == JOY_LEFT) LEFT_JOY_GPIO_CLK_ENA
BLE(); else\
00292
  if ((__JOY__) == JOY_RIGHT) RIGHT_JOY_GPIO_CLK_E
```

```
NABLE(); else\
00293
  if ((__JOY__) == JOY_DOWN) DOWN_JOY_GPIO_CLK_ENA
BLE(); else\
00294
  if ((__JOY__) == JOY_UP) UP_JOY_GPIO_CLK_ENABLE(
);} while(0)
00295
00296 #define JOYx_GPIO_CLK_DISABLE(__JOY__)   (((
__JOY__) == JOY_SEL) ? SEL_JOY_GPIO_CLK_DISABLE()
:\
00297                                             ((
__JOY__) == JOY_LEFT) ? LEFT_JOY_GPIO_CLK_DISABLE(
) :\
00298                                             ((
__JOY__) == JOY_RIGHT) ? RIGHT_JOY_GPIO_CLK_DISABL
E() :\
00299                                             ((
__JOY__) == JOY_DOWN) ? DOWN_JOY_GPIO_CLK_DISABLE(
) :\
00300                                             ((
__JOY__) == JOY_UP) ? UP_JOY_GPIO_CLK_DISABLE() :
0 )
00301
00302 /**
00303   * @}
00304   */
00305
00306 /** @addtogroup STM3210E_EVAL_COM
00307   * @{
00308   */
00309 #define COMn                                2
00310
00311 /**
00312  * @brief Definition for COM port1, connecte
d to USART1
00313  */
```

```
00314 #define EVAL_COM1                           USA
RT1
00315 #define EVAL_COM1_CLK_ENABLE()              __H
AL_RCC_USART1_CLK_ENABLE()
00316 #define EVAL_COM1_CLK_DISABLE()             __H
AL_RCC_USART1_CLK_DISABLE()
00317
00318 #define EVAL_COM1_TX_PIN                    GPI
O_PIN_9              /* PA.09*/
00319 #define EVAL_COM1_TX_GPIO_PORT              GPI
OA
00320 #define EVAL_COM1_TX_GPIO_CLK_ENABLE()      __H
AL_RCC_GPIOA_CLK_ENABLE()
00321 #define EVAL_COM1_TX_GPIO_CLK_DISABLE()     __H
AL_RCC_GPIOA_CLK_DISABLE()
00322
00323 #define EVAL_COM1_RX_PIN                    GPI
O_PIN_10              /* PA.10*/
00324 #define EVAL_COM1_RX_GPIO_PORT              GPI
OA
00325 #define EVAL_COM1_RX_GPIO_CLK_ENABLE()      __H
AL_RCC_GPIOA_CLK_ENABLE()
00326 #define EVAL_COM1_RX_GPIO_CLK_DISABLE()     __H
AL_RCC_GPIOA_CLK_DISABLE()
00327
00328 #define EVAL_COM1_IRQn                      USA
RT1_IRQn
00329
00330 /**
00331   * @brief Definition for COM port2, connecte
d to USART2
00332   */
00333 #define EVAL_COM2                           USA
RT2
00334 #define EVAL_COM2_CLK_ENABLE()              __H
AL_RCC_USART2_CLK_ENABLE()
00335 #define EVAL_COM2_CLK_DISABLE()             __H
```

```
AL_RCC_USART2_CLK_DISABLE()
00336
00337 #define EVAL_COM2_TX_PIN                    GPI
O_PIN_2              /* PA.02*/
00338 #define EVAL_COM2_TX_GPIO_PORT              GPI
OA
00339 #define EVAL_COM2_TX_GPIO_CLK_ENABLE()   __H
AL_RCC_GPIOA_CLK_ENABLE()
00340 #define EVAL_COM2_TX_GPIO_CLK_DISABLE()  __H
AL_RCC_GPIOA_CLK_DISABLE()
00341
00342 #define EVAL_COM2_RX_PIN                    GPI
O_PIN_3              /* PA.03*/
00343 #define EVAL_COM2_RX_GPIO_PORT              GPI
OA
00344 #define EVAL_COM2_RX_GPIO_CLK_ENABLE()   __H
AL_RCC_GPIOA_CLK_ENABLE()
00345 #define EVAL_COM2_RX_GPIO_CLK_DISABLE()  __H
AL_RCC_GPIOA_CLK_DISABLE()
00346
00347 #define EVAL_COM2_IRQn                      USA
RT2_IRQn
00348
00349 #define COMx_CLK_ENABLE(__INDEX__)
    do { if((__INDEX__) == COM1) EVAL_COM1_CLK_ENA
BLE(); else\
00350
         if((__INDEX__) == COM2) EVAL_COM2_CLK_ENA
BLE();} while(0)
00351
00352 #define COMx_CLK_DISABLE(__INDEX__)
    (((__INDEX__) == COM1) ? EVAL_COM1_CLK_DISABLE
() :\
00353
     ((__INDEX__) == COM2) ? EVAL_COM2_CLK_DISABLE
() : 0)
00354
```

```
00355 #define COMx_TX_GPIO_CLK_ENABLE(__INDEX__)
    do { if((__INDEX__) == COM1) EVAL_COM1_TX_GPIO
_CLK_ENABLE(); else\
00356
         if((__INDEX__) == COM2) EVAL_COM2_TX_GPIO
_CLK_ENABLE();} while(0)
00357
00358 #define COMx_TX_GPIO_CLK_DISABLE(__INDEX__)
    (((__INDEX__) == COM1) ? EVAL_COM1_TX_GPIO_CLK
_DISABLE() :\
00359
     ((__INDEX__) == COM2) ? EVAL_COM2_TX_GPIO_CLK
_DISABLE() : 0)
00360
00361 #define COMx_RX_GPIO_CLK_ENABLE(__INDEX__)
    do { if((__INDEX__) == COM1) EVAL_COM1_RX_GPIO
_CLK_ENABLE(); else\
00362
         if((__INDEX__) == COM2) EVAL_COM2_RX_GPIO
_CLK_ENABLE();} while(0)
00363
00364 #define COMx_RX_GPIO_CLK_DISABLE(__INDEX__)
    (((__INDEX__) == COM1) ? EVAL_COM1_RX_GPIO_CLK
_DISABLE() :\
00365
     ((__INDEX__) == COM2) ? EVAL_COM2_RX_GPIO_CLK
_DISABLE() : 0)
00366
00367 /**
00368   * @}
00369   */
00370
00371 /** @addtogroup STM3210E_EVAL_BUS
00372   * @{
00373   */
00374
00375 /* Exported constant IO --------------------
```

```
----------------------------------*/
00376 /*####################### I2Cx ###############
#####################*/
00377 /* User can use this section to tailor I2Cx
instance used and associated
00378    resources */
00379 /* Definition for I2Cx Pins */
00380 #define EVAL_I2Cx_SCL_PIN
    GPIO_PIN_6         /* PB.06*/
00381 #define EVAL_I2Cx_SCL_GPIO_PORT
    GPIOB
00382 #define EVAL_I2Cx_SDA_PIN
    GPIO_PIN_7         /* PB.07*/
00383 #define EVAL_I2Cx_SDA_GPIO_PORT
    GPIOB
00384
00385 /* Definition for I2Cx clock resources */
00386 #define EVAL_I2Cx
    I2C1
00387 #define EVAL_I2Cx_CLK_ENABLE()
    __HAL_RCC_I2C1_CLK_ENABLE()
00388 #define EVAL_I2Cx_SDA_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOB_CLK_ENABLE()
00389 #define EVAL_I2Cx_SCL_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOB_CLK_ENABLE()
00390
00391 #define EVAL_I2Cx_FORCE_RESET()
    __HAL_RCC_I2C1_FORCE_RESET()
00392 #define EVAL_I2Cx_RELEASE_RESET()
    __HAL_RCC_I2C1_RELEASE_RESET()
00393
00394 /* Definition for I2Cx's NVIC */
00395 #define EVAL_I2Cx_EV_IRQn
    I2C1_EV_IRQn
00396 #define EVAL_I2Cx_EV_IRQHandler
    I2C1_EV_IRQHandler
00397 #define EVAL_I2Cx_ER_IRQn
```

```
    I2C1_ER_IRQn
00398 #define EVAL_I2Cx_ER_IRQHandler
    I2C1_ER_IRQHandler
00399
00400 /* I2C clock speed configuration (in Hz)
00401    WARNING:
00402    Make sure that this define is not already
 declared in other files (ie.
00403    stm3210e_eval.h file). It can be used in
parallel by other modules. */
00404 #ifndef BSP_I2C_SPEED
00405  #define BSP_I2C_SPEED
      100000
00406 #endif /* I2C_SPEED */
00407
00408
00409 /* Maximum Timeout values for flags waiting
loops. These timeouts are not based
00410    on accurate values, they just guarantee t
hat the application will not remain
00411    stuck if the I2C communication is corrupt
ed.
00412    You may modify these timeout values depen
ding on CPU frequency and application
00413    conditions (interrupts routines ...). */

00414 #define EVAL_I2Cx_TIMEOUT_MAX
    3000
00415
00416 /*######################## SPIx ###############
#####################*/
00417 #define EVAL_SPIx
    SPI1
00418 #define EVAL_SPIx_CLK_ENABLE()
    __HAL_RCC_SPI1_CLK_ENABLE()
00419
00420 #define EVAL_SPIx_SCK_GPIO_PORT
```

```
    GPIOA              /* PA.05*/
00421 #define EVAL_SPIx_SCK_PIN
    GPIO_PIN_5
00422 #define EVAL_SPIx_SCK_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOA_CLK_ENABLE()
00423 #define EVAL_SPIx_SCK_GPIO_CLK_DISABLE()
    __HAL_RCC_GPIOA_CLK_DISABLE()
00424
00425 #define EVAL_SPIx_MISO_MOSI_GPIO_PORT
    GPIOA
00426 #define EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE(
)   __HAL_RCC_GPIOA_CLK_ENABLE()
00427 #define EVAL_SPIx_MISO_MOSI_GPIO_CLK_DISABLE
()  __HAL_RCC_GPIOA_CLK_DISABLE()
00428 #define EVAL_SPIx_MISO_PIN
    GPIO_PIN_6         /* PA.06*/
00429 #define EVAL_SPIx_MOSI_PIN
    GPIO_PIN_7         /* PA.07*/
00430 /* Maximum Timeout values for flags waiting
loops. These timeouts are not based
00431    on accurate values, they just guarantee t
hat the application will not remain
00432    stuck if the SPI communication is corrupt
ed.
00433    You may modify these timeout values depen
ding on CPU frequency and application
00434    conditions (interrupts routines ...). */

00435 #define EVAL_SPIx_TIMEOUT_MAX
    1000
00436
00437 /**
00438   * @}
00439   */
00440
00441 /** @addtogroup STM3210E_EVAL_COMPONENT
00442   * @{
```

```
00443   */
00444
00445 /*###################### FLASH SPI ##########
##########################*/
00446 /**
00447   * @brief  M25P FLASH SPI Chip Select macro
 definition
00448   */
00449 #define FLASH_SPI_CS_LOW()       HAL_GPIO_Wr
itePin(FLASH_SPI_CS_GPIO_PORT, FLASH_SPI_CS_PIN, G
PIO_PIN_RESET)
00450 #define FLASH_SPI_CS_HIGH()      HAL_GPIO_Wr
itePin(FLASH_SPI_CS_GPIO_PORT, FLASH_SPI_CS_PIN, G
PIO_PIN_SET)
00451
00452 /**
00453   * @brief  M25P FLASH SPI Control Interface
 pins
00454   */
00455 #define FLASH_SPI_CS_PIN
       GPIO_PIN_2         /* PB.02*/
00456 #define FLASH_SPI_CS_GPIO_PORT
       GPIOB
00457 #define FLASH_SPI_CS_GPIO_CLK_ENABLE()
       __HAL_RCC_GPIOB_CLK_ENABLE()
00458 #define FLASH_SPI_CS_GPIO_CLK_DISABLE()
       __HAL_RCC_GPIOB_CLK_DISABLE()
00459
00460 /**
00461   * @brief  M25P FLASH SPI supported commands

00462   */
00463 #define FLASH_SPI_CMD_WRITE          0x02  /
*!< Write to Memory instruction */
00464 #define FLASH_SPI_CMD_WRSR           0x01  /
*!< Write Status Register instruction */
00465 #define FLASH_SPI_CMD_WREN           0x06  /
```

```
*!< Write enable instruction */
00466 #define FLASH_SPI_CMD_READ           0x03  /
*!< Read from Memory instruction */
00467 #define FLASH_SPI_CMD_RDSR           0x05  /
*!< Read Status Register instruction  */
00468 #define FLASH_SPI_CMD_RDID           0x9F  /
*!< Read identification */
00469 #define FLASH_SPI_CMD_SE             0xD8  /
*!< Sector Erase instruction */
00470 #define FLASH_SPI_CMD_BE             0xC7  /
*!< Bulk Erase instruction */
00471
00472 #define FLASH_SPI_WIP_FLAG           0x01  /
*!< Write In Progress (WIP) flag */
00473
00474 #define FLASH_SPI_DUMMY_BYTE         0xA5
00475 #define FLASH_SPI_PAGESIZE           0x100
00476
00477 #define FLASH_SPI_M25P128_ID         0x202018

00478 #define FLASH_SPI_M25P64_ID          0x202017

00479
00480
00481 /*###################### AUDIO ##############
############*/
00482 /**
00483   * @brief  AUDIO I2C Interface pins
00484   */
00485 #define AUDIO_I2C_ADDRESS
  0x27
00486
00487   /* Audio Reset Pin definition */
00488 #define AUDIO_RESET_GPIO_CLK_ENABLE()
  __HAL_RCC_GPIOG_CLK_ENABLE()
00489 #define AUDIO_RESET_PIN
  GPIO_PIN_11
```

```
00490 #define AUDIO_RESET_GPIO
  GPIOG
00491
00492 /**
00493   * @}
00494   */
00495
00496 /**
00497   * @}
00498   */
00499
00500
00501
00502 /** @addtogroup STM3210E_EVAL_Exported_Funct
ions
00503   * @{
00504   */
00505 uint32_t                BSP_GetVersion(void)
;
00506 void                    BSP_LED_Init(Led_Typ
eDef Led);
00507 void                    BSP_LED_On(Led_TypeD
ef Led);
00508 void                    BSP_LED_Off(Led_Type
Def Led);
00509 void                    BSP_LED_Toggle(Led_T
ypeDef Led);
00510 void                    BSP_PB_Init(Button_T
ypeDef Button, ButtonMode_TypeDef Button_Mode);
00511 uint32_t                BSP_PB_GetState(Butt
on_TypeDef Button);
00512 #ifdef HAL_UART_MODULE_ENABLED
00513 void                    BSP_COM_Init(COM_Typ
eDef COM, UART_HandleTypeDef* huart);
00514 #endif /* HAL_UART_MODULE_ENABLED */
00515 uint8_t                 BSP_JOY_Init(JOYMode
_TypeDef Joy_Mode);
```

```
00516 JOYState_TypeDef        BSP_JOY_GetState(void
);
00517
00518 /**
00519   * @}
00520   */
00521
00522
00523 /**
00524   * @}
00525   */
00526
00527 /**
00528   * @}
00529   */
00530
00531 /**
00532   * @}
00533   */
00534
00535 #ifdef __cplusplus
00536 }
00537 #endif
00538
00539 #endif /* __STM3210E_EVAL_H */
00540
00541 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
00542
```

STM3210E_EVAL BSP User Manual

stm3210e_eval_audio.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210e_eval_audio.h
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          stm3210e_eval_audio.c driver.
00009   ******************************************************************************
00010   * @attention
00011   *
00012   * <h2><center>&copy; COPYRIGHT(c) 2015 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 __STM3210E_EVAL_AUDIO_H
00041 #define __STM3210E_EVAL_AUDIO_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------------------------------------------*/
00048 /* Include audio component Driver */
00049 #include "../Components/ak4343/ak4343.h"
00050 #include "stm3210e_eval.h"
00051
00052 /** @addtogroup BSP
00053   * @{
00054   */
00055
00056 /** @addtogroup STM3210E_EVAL
00057   * @{
00058   */
00059
00060 /** @addtogroup STM3210E_EVAL_AUDIO
00061   * @{
00062   */
00063
00064
00065 /** @defgroup STM3210E_EVAL_AUDIO_Exported_Types AUDIO_Exported_Types
00066   * @{
00067   */
00068
```

```
00069 /**
00070   * @}
00071   */
00072
00073 /** @defgroup STM3210E_EVAL_AUDIO_OUT_Export
ed_Constants AUDIO_OUT_Exported_Constants
00074   * @{
00075   */
00076
00077
00078 /*-----------------------------------------
-----------------------------------
00079              AUDIO OUT CONFIGURATION
00080 -----------------------------------------
-----------------------------------*/
00081
00082 /* I2S peripheral configuration defines */
00083 #define I2SOUT                            SPI2
00084 #define I2SOUT_CLK_ENABLE()               __HA
L_RCC_SPI2_CLK_ENABLE()
00085 #define I2SOUT_SCK_SD_CLK_ENABLE()        __HA
L_RCC_GPIOB_CLK_ENABLE()
00086 #define I2SOUT_MCK_CLK_ENABLE()           __HA
L_RCC_GPIOC_CLK_ENABLE()
00087 #define I2SOUT_WS_CLK_ENABLE()            __HA
L_RCC_GPIOB_CLK_ENABLE()
00088 #define I2SOUT_WS_PIN                     GPIO
_PIN_12             /* PB.12*/
00089 #define I2SOUT_SCK_PIN                    GPIO
_PIN_13             /* PB.13*/
00090 #define I2SOUT_SD_PIN                     GPIO
_PIN_15             /* PB.15*/
00091 #define I2SOUT_MCK_PIN                    GPIO
_PIN_6              /* PC.06*/
00092 #define I2SOUT_SCK_SD_GPIO_PORT           GPIOB

00093 #define I2SOUT_WS_GPIO_PORT               GPIOB
```

```
00094 #define I2SOUT_MCK_GPIO_PORT             GPIOC

00095
00096 /* I2S DMA Channel definitions */
00097 #define I2SOUT_DMAx_CLK_ENABLE()         __HA
L_RCC_DMA1_CLK_ENABLE()
00098 #define I2SOUT_DMAx_CHANNEL              DMA1
_Channel5
00099 #define I2SOUT_DMAx_IRQ                  DMA1
_Channel5_IRQn
00100 #define I2SOUT_DMAx_PERIPH_DATA_SIZE     DMA_
PDATAALIGN_HALFWORD
00101 #define I2SOUT_DMAx_MEM_DATA_SIZE        DMA_
MDATAALIGN_HALFWORD
00102 #define DMA_MAX_SZE                      0xFF
FF
00103
00104 #define I2SOUT_IRQHandler                DMA1
_Channel5_IRQHandler
00105
00106 /* Select the interrupt preemption priority
and subpriority for the DMA interrupt */
00107 #define AUDIO_OUT_IRQ_PREPRIO          5   /*
 Select the preemption priority level(0 is the hig
hest) */
00108
00109 /*-----------------------------------------
-----------------------------------
00110              CONFIGURATION: Audio Driver Con
figuration parameters
00111 -----------------------------------------
-----------------------------------*/
00112
00113 /* Audio status definition */
00114 #define AUDIO_OK
  0
```

```
00115 #define AUDIO_ERROR
  1
00116 #define AUDIO_TIMEOUT
  2
00117
00118
00119 /*-----------------------------------------
-----------------------------------
00120                       OPTIONAL Configuration d
efines parameters
00121 -----------------------------------------
-----------------------------------*/
00122
00123
00124 /**
00125   * @}
00126   */
00127
00128 /** @defgroup STM3210E_EVAL_AUDIO_Exported_M
acros AUDIO_Exported_Macros
00129   * @{
00130   */
00131 #define DMA_MAX(_X_)                (((_X_)
<= DMA_MAX_SZE)? (_X_):DMA_MAX_SZE)
00132
00133 /**
00134   * @}
00135   */
00136
00137 /** @addtogroup STM3210E_EVAL_AUDIO_OUT_Expo
rted_Functions
00138   * @{
00139   */
00140 uint8_t BSP_AUDIO_OUT_Init(uint16_t OutputDe
vice, uint8_t Volume, uint32_t AudioFreq);
00141 uint8_t BSP_AUDIO_OUT_Play(uint16_t* pBuffer
, uint32_t Size);
```

```
00142 void    BSP_AUDIO_OUT_ChangeBuffer(uint16_t
*pData, uint16_t Size);
00143 uint8_t BSP_AUDIO_OUT_Pause(void);
00144 uint8_t BSP_AUDIO_OUT_Resume(void);
00145 uint8_t BSP_AUDIO_OUT_Stop(uint32_t Option);
00146 uint8_t BSP_AUDIO_OUT_SetVolume(uint8_t Volu
me);
00147 void    BSP_AUDIO_OUT_SetFrequency(uint32_t
AudioFreq);
00148 uint8_t BSP_AUDIO_OUT_SetMute(uint32_t Comma
nd);
00149 uint8_t BSP_AUDIO_OUT_SetOutputMode(uint8_t
Output);
00150
00151 /* User Callbacks: user has to implement the
se functions in his code if they are needed. */
00152 /* This function is called when the requeste
d data has been completely transferred.*/
00153 void    BSP_AUDIO_OUT_TransferComplete_CallB
ack(void);
00154
00155 /* This function is called when half of the
requested buffer has been transferred. */
00156 void    BSP_AUDIO_OUT_HalfTransfer_CallBack(
void);
00157
00158 /* This function is called when an Interrupt
 due to transfer error on or peripheral
00159    error occurs. */
00160 void    BSP_AUDIO_OUT_Error_CallBack(void);
00161
00162 /**
00163   * @}
00164   */
00165
00166
00167
```

```
00168 /**
00169   * @}
00170   */
00171
00172 /**
00173   * @}
00174   */
00175
00176 /**
00177   * @}
00178   */
00179
00180 #ifdef __cplusplus
00181 }
00182 #endif
00183
00184 #endif /* __STM3210E_EVAL_AUDIO_H */
00185
00186 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210E_EVAL BSP User Manual

stm3210e_eval_audio.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210e_eval_audio.c
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file provides the Audio driver for the STM3210E-Eval
00008   *          board.
00009   @verbatim
00010   ==============================================================================
00011                        ##### How to use this driver #####
00012   ==============================================================================
00013     [..]
00014   (#) This driver supports STM32F103xE devices on STM3210E-Eval Kit:
00015        (++) to play an audio file (all functions names start by BSP_AUDIO_OUT_xxx)
```

```
00016
00017   [..]
00018    (#) PLAY A FILE:
00019        (++) Call the function BSP_AUDIO_OUT_
Init(
00020               OutputDevice: physical output
mode (OUTPUT_DEVICE_SPEAKER,
00021                                  OUTPUT_DEVICE_HEA
DPHONE, OUTPUT_DEVICE_BOTH)
00022               Volume: initial volume to be s
et (0 is min (mute), 100 is max (100%)
00023               AudioFreq: Audio frequency in
Hz (8000, 16000, 22500, 32000 ...)
00024               this parameter is relative to
the audio file/stream type.
00025                )
00026            This function configures all the
hardware required for the audio application
00027            (codec, I2C, I2S, GPIOs, DMA and
interrupt if needed). This function returns 0
00028            if configuration is OK.
00029            If the returned value is differen
t from 0 or the function is stuck then the
00030            communication with the codec (try
 to un-plug the power or reset device in this case
).
00031               (+++) OUTPUT_DEVICE_SPEAKER: o
nly speaker will be set as output for the
00032                     audio stream.
00033               (+++) OUTPUT_DEVICE_HEADPHONE:
 only headphones will be set as output for
00034                     the audio stream.
00035               (+++) OUTPUT_DEVICE_BOTH: both
 Speaker and Headphone are used as outputs
00036                     for the audio stream at
the same time.
00037        (++) Call the function BSP_AUDIO_OUT_
```

```
Play(
00038                   pBuffer: pointer to the audi
o data file address
00039                   Size: size of the buffer to
be sent in Bytes
00040                  )
00041                to start playing (for the firs
t time) from the audio file/stream.
00042         (++) Call the function BSP_AUDIO_OUT_
Pause() to pause playing
00043         (++) Call the function BSP_AUDIO_OUT_
Resume() to resume playing.
00044              Note. After calling BSP_AUDIO_OU
T_Pause() function for pause,
00045              only BSP_AUDIO_OUT_Resume() shou
ld be called for resume
00046              (it is not allowed to call BSP_A
UDIO_OUT_Play() in this case).
00047              Note. This function should be ca
lled only when the audio file is played
00048              or paused (not stopped).
00049         (++) For each mode, you may need to i
mplement the relative callback functions
00050              into your code.
00051              The Callback functions are named
 BSP_AUDIO_OUT_XXXCallBack() and only
00052              their prototypes are declared in
 the stm3210e_eval_audio.h file.
00053              (refer to the example for more d
etails on the callbacks implementations)
00054         (++) To Stop playing, to modify the v
olume level, the frequency or to mute,
00055              use the functions BSP_AUDIO_OUT_
Stop(), BSP_AUDIO_OUT_SetVolume(),
00056              AUDIO_OUT_SetFrequency() BSP_AUD
IO_OUT_SetOutputMode and BSP_AUDIO_OUT_SetMute().
00057         (++) The driver API and the callback
```

```
functions are at the end of the
00058             stm3210e_eval_audio.h file.
00059
00060        (++) This driver provide the High Aud
io Layer: consists of the function API
00061             exported in the stm3210e_eval_au
dio.h file (BSP_AUDIO_OUT_Init(),
00062             BSP_AUDIO_OUT_Play() ...)
00063        (++) This driver provide also the Med
ia Access Layer (MAL): which consists
00064             of functions allowing to access
the media containing/providing the
00065             audio file/stream. These functio
ns are also included as local functions into
00066             the stm3210e_eval_audio.c file (
I2SOUT_Init()...)
00067
00068   [..]
00069                        ##### Known Limitations
 #####
00070   ============================================
===================================
00071    (#) When using the Speaker, if the audio
file quality is not high enough, the
00072        speaker output may produce high and u
ncomfortable noise level. To avoid
00073        this issue, to use speaker output pro
perly, try to increase audio file
00074        sampling rate (typically higher than
48KHz).
00075        This operation will lead to larger fi
le size.
00076
00077    (#) Communication with the audio codec (t
hrough I2C) may be corrupted if it
00078        is interrupted by some user interrupt
 routines (in this case, interrupts
```

```
00079         could be disabled just before the sta
rt of communication then re-enabled
00080         when it is over). Note that this comm
unication is only done at the
00081         configuration phase (BSP_AUDIO_OUT_In
it() or BSP_AUDIO_OUT_Stop())
00082         and when Volume control modification
is performed (BSP_AUDIO_OUT_SetVolume()
00083         or BSP_AUDIO_OUT_SetMute()or BSP_AUDI
O_OUT_SetOutputMode()).
00084         When the audio data is played, no com
munication is required with the audio codec.
00085
00086    (#) Parsing of audio file is not implemen
ted (in order to determine audio file
00087         properties: Mono/Stereo, Data size, F
ile size, Audio Frequency, Audio Data
00088         header size ...). The configuration i
s fixed for the given audio file.
00089
00090    (#) Mono audio streaming is not supported
 (in order to play mono audio streams,
00091         each data should be sent twice on the
 I2S or should be duplicated on the
00092         source buffer. Or convert the stream
in stereo before playing).
00093
00094    (#) Supports only 16-bit audio data size.
00095
00096   @endverbatim
00097   ******************************************
**********************************
00098   * @attention
00099   *
00100   * <h2><center>&copy; COPYRIGHT(c) 2015 STM
icroelectronics</center></h2>
00101   *
```

```
00102   * Redistribution and use in source and bin
ary forms, with or without modification,
00103   * are permitted provided that the followin
g conditions are met:
00104   *   1. Redistributions of source code must
 retain the above copyright notice,
00105   *      this list of conditions and the fol
lowing disclaimer.
00106   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00107   *      this list of conditions and the fol
lowing disclaimer in the documentation
00108   *      and/or other materials provided wit
h the distribution.
00109   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00110   *      may be used to endorse or promote p
roducts derived from this software
00111   *      without specific prior written perm
ission.
00112   *
00113   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00114   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00115   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00116   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00117   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00118   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
00119   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00120   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
```

```
00121   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00122   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00123   *
00124   ****************************************
**********************************
00125   */
00126
00127 /* Includes ------------------------------
-----------------------------------*/
00128 #include "stm32f1xx_hal.h"      /* Include H
AL driver definitions to have access to I2S litera
ls in audio driver */
00129 #include "stm3210e_eval_audio.h"
00130
00131 /* Private variables ---------------------
-----------------------------------*/
00132
00133 /** @addtogroup BSP
00134   * @{
00135   */
00136
00137 /** @addtogroup STM3210E_EVAL
00138   * @{
00139   */
00140
00141 /** @addtogroup STM3210E_EVAL_AUDIO
00142   * @brief This file includes the low layer
audio driver available on STM3210E-Eval
00143   * eval board.
00144   * @{
00145   */
00146
00147 /** @defgroup STM3210E_EVAL_AUDIO_Private_Ty
pes AUDIO_Private_Types
00148   * @{
```

```
00149   */
00150 /**
00151   * @}
00152   */
00153
00154 /** @defgroup STM3210E_EVAL_AUDIO_Private_De
fines AUDIO_Private_Defines
00155   * @{
00156   */
00157
00158 /**
00159   * @}
00160   */
00161
00162 /** @defgroup STM3210E_EVAL_AUDIO_Private_Ma
cros AUDIO Private_Macros
00163   * @{
00164   */
00165 /**
00166   * @}
00167   */
00168
00169 /** @defgroup STM3210E_EVAL_AUDIO_Private_Va
riables AUDIO_Private_Variables
00170   * @{
00171   */
00172 #define AUDIO_SAMPLE_SENDDUMMYDATA_SIZE
  8
00173
00174 const uint16_t audio_sample_SendDummyData[]
=
00175   {0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0
x0000, 0x0000, 0x0000};
00176
00177 static __IO uint32_t SendDummyData = DISABLE
;   /* Variable to switch between sending dummy da
ta or audio data on bus I2S */
```

```
00178
00179
00180 /*### PLAY ###*/
00181 static AUDIO_DrvTypeDef          *pAudioDrv;
00182 I2S_HandleTypeDef                hAudioOutI2s
;
00183
00184 /**
00185   * @}
00186   */
00187
00188 /** @defgroup STM3210E_EVAL_AUDIO_Private_Fu
nction_Prototypes AUDIO_Private_Function_Prototypes

00189   * @{
00190   */
00191 static void  I2SOUT_MspInit(void);
00192 static void  I2SOUT_Init(uint32_t AudioFreq)
;
00193 static uint8_t  I2SOUT_SendDummyData_Start(u
int16_t* pBuffer, uint32_t Size);
00194 static uint8_t  I2SOUT_SendDummyData_Stop(vo
id);
00195
00196 /**
00197   * @}
00198   */
00199
00200 /** @defgroup STM3210E_EVAL_AUDIO_OUT_Export
ed_Functions AUDIO_OUT_Exported_Functions
00201   * @{
00202   */
00203
00204 /**
00205   * @brief  Configure the audio peripherals.
00206   * @param  OutputDevice: OUTPUT_DEVICE_SPEA
KER, OUTPUT_DEVICE_HEADPHONE,
```

```
00207   *                           OUTPUT_DEVICE_BOTH.
00208   * @param  Volume: Initial volume level (fr
om 0 (Mute) to 100 (Max))
00209   * @param  AudioFreq: Audio frequency used
to play the audio stream.
00210   * @retval 0 if correct communication, else
 wrong communication
00211   */
00212 uint8_t BSP_AUDIO_OUT_Init(uint16_t OutputDe
vice, uint8_t Volume, uint32_t AudioFreq)
00213 {
00214   uint8_t ret = AUDIO_ERROR;
00215
00216   /* I2S data transfer preparation:
                                    */
00217   /* Prepare the Media to be used for the au
dio transfer from memory to I2S   */
00218   /* peripheral.
                                    */
00219   /* Note: This function must be called befo
re audio codec init because       */
00220   /*       I2S initialisation parameter "MCL
KOutput" determines whether the   */
00221   /*       audio Codec is driven by I2S cloc
k (I2S_MCLKOUTPUT_ENABLE)         */
00222   /*       or by its own internal PLL (I2S_M
CLKOUTPUT_DISABLE).               */
00223   /* Configure the I2S peripheral */
00224
00225   /* Switch to send dummy data on bus I2S (s
witch used in I2SOUT_Init() */
00226   SendDummyData = ENABLE;
00227
00228   I2SOUT_Init(AudioFreq);
00229
00230   /* Assign value to codec driver parameter
```

```
I2S_MCLKOutput in function of     */
00231   /* I2S initialisation parameter "MCLKOutpu
t".                                */
00232   if (hAudioOutI2s.Init.MCLKOutput == I2S_MC
LKOUTPUT_DISABLE)
00233   {
00234     ak4343_MCLKOutput(0);
00235   }
00236   else
00237   {
00238     ak4343_MCLKOutput(1);
00239   }
00240
00241   /* Initialize the audio driver structure */

00242   pAudioDrv = &ak4343_drv;
00243
00244   /* Generate I2S master clock MCK */
00245   /* (Start the audio player to send dummy d
ata on I2S bus) */
00246   if(I2SOUT_SendDummyData_Start((uint16_t *)
audio_sample_SendDummyData, AUDIO_SAMPLE_SENDDUMMY
DATA_SIZE) == 0)
00247   {
00248     if (pAudioDrv->Init(AUDIO_I2C_ADDRESS, O
utputDevice, Volume, AudioFreq) == 0)
00249     {
00250       ret = AUDIO_OK;
00251     }
00252     else
00253     {
00254       ret = AUDIO_ERROR;
00255     }
00256
00257     I2SOUT_SendDummyData_Stop();
00258
00259     /* Switch to send audio data on bus I2S
```

```
*/
00260     SendDummyData = DISABLE;
00261
00262     I2SOUT_Init(AudioFreq);
00263   }
00264   else
00265   {
00266     ret = AUDIO_ERROR;
00267   }
00268
00269
00270   return ret;
00271 }
00272
00273 /**
00274   * @brief  Starts playing audio stream from
 a data buffer for a determined size.
00275   * @param  pBuffer: Pointer to the buffer
00276   * @param  Size: Number of audio data BYTES.

00277   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00278   */
00279 uint8_t BSP_AUDIO_OUT_Play(uint16_t* pBuffer
, uint32_t Size)
00280 {
00281   /* Call the audio Codec Play function */
00282   if(pAudioDrv->Play(AUDIO_I2C_ADDRESS, pBuf
fer, Size) != 0)
00283   {
00284     return AUDIO_ERROR;
00285   }
00286   else
00287   {
00288     /* Update the Media layer and enable it
for play */
00289     HAL_I2S_Transmit_DMA(&hAudioOutI2s, pBuf
```

```
fer, DMA_MAX(Size));
00290
00291     return AUDIO_OK;
00292   }
00293 }
00294
00295 /**
00296   * @brief Sends n-Bytes on the I2S interfac
e.
00297   * @param pData: pointer on data address
00298   * @param Size: number of data to be written

00299   * @retval None
00300   */
00301 void BSP_AUDIO_OUT_ChangeBuffer(uint16_t *pD
ata, uint16_t Size)
00302 {
00303   HAL_I2S_Transmit_DMA(&hAudioOutI2s, pData,
 Size);
00304 }
00305
00306 /**
00307   * @brief  This function Pauses the audio f
ile stream. In case
00308   *         of using DMA, the DMA Pause feat
ure is used.
00309   * @note When calling BSP_AUDIO_OUT_Pause()
 function for pause, only
00310   *          BSP_AUDIO_OUT_Resume() function
 should be called for resume (use of BSP_AUDIO_OUT
_Play()
00311   *          function for resume could lead
to unexpected behavior).
00312   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00313   */
00314 uint8_t BSP_AUDIO_OUT_Pause(void)
```

```
00315 {
00316   /* Call the Audio Codec Pause/Resume funct
ion */
00317   if(pAudioDrv->Pause(AUDIO_I2C_ADDRESS) !=
0)
00318   {
00319     return AUDIO_ERROR;
00320   }
00321   else
00322   {
00323     /* Call the Media layer pause function */

00324     HAL_I2S_DMAPause(&hAudioOutI2s);
00325
00326     /* Return AUDIO_OK if all operations are
 OK */
00327     return AUDIO_OK;
00328   }
00329 }
00330
00331 /**
00332   * @brief  This function  Resumes the audio
 file stream.
00333   * @note When calling BSP_AUDIO_OUT_Pause()
 function for pause, only
00334   *           BSP_AUDIO_OUT_Resume() function
 should be called for resume (use of BSP_AUDIO_OUT
_Play()
00335   *           function for resume could lead
to unexpected behavior).
00336   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00337   */
00338 uint8_t BSP_AUDIO_OUT_Resume(void)
00339 {
00340   /* Call the Audio Codec Pause/Resume funct
ion */
```

```
00341   if(pAudioDrv->Resume(AUDIO_I2C_ADDRESS) !=
 0)
00342   {
00343     return AUDIO_ERROR;
00344   }
00345   else
00346   {
00347     /* Call the Media layer resume function
*/
00348     HAL_I2S_DMAResume(&hAudioOutI2s);
00349     /* Return AUDIO_OK if all operations are
 OK */
00350     return AUDIO_OK;
00351   }
00352 }
00353
00354 /**
00355   * @brief  Stops audio playing and Power do
wn the Audio Codec.
00356   * @param  Option: could be one of the foll
owing parameters
00357   *           - CODEC_PDWN_HW: completely sh
ut down the codec (physically).
00358   *                              Then need to
reconfigure the Codec after power on.
00359   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00360   */
00361 uint8_t BSP_AUDIO_OUT_Stop(uint32_t Option)
00362 {
00363   /* Call DMA Stop to disable DMA stream bef
ore stopping codec */
00364   HAL_I2S_DMAStop(&hAudioOutI2s);
00365
00366   /* Call Audio Codec Stop function */
00367   if(pAudioDrv->Stop(AUDIO_I2C_ADDRESS, Opti
on) != 0)
```

```
00368   {
00369     return AUDIO_ERROR;
00370   }
00371   else
00372   {
00373     if(Option == CODEC_PDWN_HW)
00374     {
00375       /* Wait at least 100us */
00376       HAL_Delay(1);
00377
00378       /* Power Down the codec */
00379       HAL_GPIO_WritePin(AUDIO_RESET_GPIO, AU
DIO_RESET_PIN, GPIO_PIN_RESET);
00380
00381     }
00382     /* Return AUDIO_OK when all operations a
re correctly done */
00383     return AUDIO_OK;
00384   }
00385 }
00386
00387 /**
00388   * @brief  Controls the current audio volum
e level.
00389   * @param  Volume: Volume level to be set i
n percentage from 0% to 100% (0 for
00390   *         Mute and 100 for Max volume leve
l).
00391   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00392   */
00393 uint8_t BSP_AUDIO_OUT_SetVolume(uint8_t Volu
me)
00394 {
00395   /* Call the codec volume control function
with converted volume value */
00396   if(pAudioDrv->SetVolume(AUDIO_I2C_ADDRESS,
```

```
 Volume) != 0)
00397   {
00398     return AUDIO_ERROR;
00399   }
00400   else
00401   {
00402     /* Return AUDIO_OK when all operations a
re correctly done */
00403     return AUDIO_OK;
00404   }
00405 }
00406
00407 /**
00408   * @brief  Enables or disables the MUTE mod
e by software
00409   * @param  Cmd: could be AUDIO_MUTE_ON to m
ute sound or AUDIO_MUTE_OFF to
00410   *         unmute the codec and restore pre
vious volume level.
00411   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00412   */
00413 uint8_t BSP_AUDIO_OUT_SetMute(uint32_t Cmd)
00414 {
00415   /* Call the Codec Mute function */
00416   if(pAudioDrv->SetMute(AUDIO_I2C_ADDRESS, C
md) != 0)
00417   {
00418     return AUDIO_ERROR;
00419   }
00420   else
00421   {
00422     /* Return AUDIO_OK when all operations a
re correctly done */
00423     return AUDIO_OK;
00424   }
00425 }
```

```
00426
00427 /**
00428   * @brief  Switch dynamically (while audio
file is played) the output target
00429   *         (speaker or headphone).
00430   * @param  Output: specifies the audio outp
ut target: OUTPUT_DEVICE_SPEAKER,
00431   *         OUTPUT_DEVICE_HEADPHONE, OUTPUT_
DEVICE_BOTH
00432   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00433   */
00434 uint8_t BSP_AUDIO_OUT_SetOutputMode(uint8_t
Output)
00435 {
00436   /* Call the Codec output Device function */

00437   if(pAudioDrv->SetOutputMode(AUDIO_I2C_ADDR
ESS, Output) != 0)
00438   {
00439     return AUDIO_ERROR;
00440   }
00441   else
00442   {
00443     /* Return AUDIO_OK when all operations a
re correctly done */
00444     return AUDIO_OK;
00445   }
00446 }
00447
00448 /**
00449   * @brief  Update the audio frequency.
00450   * @param  AudioFreq: Audio frequency used
to play the audio stream.
00451   * @retval None
00452   * @note This API should be called after th
e BSP_AUDIO_OUT_Init() to adjust the
```

```
00453   * audio frequency.
00454   */
00455 void BSP_AUDIO_OUT_SetFrequency(uint32_t Aud
ioFreq)
00456 {
00457   /* Update the I2S audio frequency configur
ation */
00458   I2SOUT_Init(AudioFreq);
00459 }
00460
00461 /**
00462   * @brief Tx Transfer completed callbacks
00463   * @param hi2s: I2S handle
00464   * @retval None
00465   */
00466 void HAL_I2S_TxCpltCallback(I2S_HandleTypeDe
f *hi2s)
00467 {
00468   if(hi2s->Instance == I2SOUT)
00469   {
00470     /* Configuration change depending on sen
ding dummy data or audio data on bus I2S */
00471     if (SendDummyData != ENABLE)
00472     {
00473       /* Call the user function which will m
anage directly transfer complete */
00474       BSP_AUDIO_OUT_TransferComplete_CallBack
();
00475     }
00476     else
00477     {
00478       /* No action */
00479     }
00480   }
00481
00482 }
00483
```

```
00484 /**
00485   * @brief Tx Transfer Half completed callba
cks
00486   * @param hi2s: I2S handle
00487   * @retval None
00488   */
00489 void HAL_I2S_TxHalfCpltCallback(I2S_HandleTy
peDef *hi2s)
00490 {
00491   if(hi2s->Instance == I2SOUT)
00492   {
00493     /* Manage the remaining file size and ne
w address offset: This function
00494     should be coded by user (its prototype i
s already declared in stm3210e_eval_audio.h) */
00495     BSP_AUDIO_OUT_HalfTransfer_CallBack();
00496   }
00497 }
00498
00499 /**
00500   * @brief I2S error callbacks
00501   * @param hi2s: I2S handle
00502   * @retval None
00503   */
00504 void HAL_I2S_ErrorCallback(I2S_HandleTypeDef
 *hi2s)
00505 {
00506   /* Manage the error generated on DMA FIFO:
 This function
00507      should be coded by user (its prototype
is already declared in stm3210e_eval_audio.h) */
00508   if(hi2s->Instance == I2SOUT)
00509   {
00510     BSP_AUDIO_OUT_Error_CallBack();
00511   }
00512 }
00513
```

```
/**
  * @brief  Manages the DMA full Transfer complete event.
  * @retval None
  */
__weak void BSP_AUDIO_OUT_TransferComplete_CallBack(void)
{
}

/**
  * @brief  Manages the DMA Half Transfer complete event.
  * @retval None
  */
__weak void BSP_AUDIO_OUT_HalfTransfer_CallBack(void)
{
}

/**
  * @brief  Manages the DMA FIFO error event.

  * @retval None
  */
__weak void BSP_AUDIO_OUT_Error_CallBack(void)
{
}

/*****************************************************************************
                                Static Function
*****************************************************************************/

/**
```

```
00543   * @brief AUDIO OUT I2S MSP Init
00544   * @retval None
00545   */
00546 static void I2SOUT_MspInit(void)
00547 {
00548   static DMA_HandleTypeDef  hdma_i2stx;
00549   GPIO_InitTypeDef          gpioinitstruct =
 {0};
00550   I2S_HandleTypeDef         *hi2s = &hAudioO
utI2s;
00551
00552   /* Enable I2SOUT clock */
00553   I2SOUT_CLK_ENABLE();
00554
00555   /*** Configure the GPIOs ***/
00556   /* Enable I2S GPIO clocks */
00557   I2SOUT_SCK_SD_CLK_ENABLE();
00558   I2SOUT_WS_CLK_ENABLE();
00559
00560   /* I2SOUT pins configuration: WS, SCK and
SD pins -----------------------------*/
00561   gpioinitstruct.Pin         = I2SOUT_SCK_PIN
 | I2SOUT_SD_PIN;
00562   gpioinitstruct.Mode        = GPIO_MODE_AF_
PP;
00563   gpioinitstruct.Pull        = GPIO_NOPULL;
00564   gpioinitstruct.Speed       = GPIO_SPEED_FR
EQ_MEDIUM;
00565   HAL_GPIO_Init(I2SOUT_SCK_SD_GPIO_PORT, &gp
ioinitstruct);
00566
00567   gpioinitstruct.Pin         = I2SOUT_WS_PIN
 ;
00568   HAL_GPIO_Init(I2SOUT_WS_GPIO_PORT, &gpioin
itstruct);
00569
00570   /* I2SOUT pins configuration: MCK pin */
```

```
00571   I2SOUT_MCK_CLK_ENABLE();
00572   gpioinitstruct.Pin         = I2SOUT_MCK_PIN
;
00573   HAL_GPIO_Init(I2SOUT_MCK_GPIO_PORT, &gpioi
nitstruct);
00574
00575   /* Enable the I2S DMA clock */
00576   I2SOUT_DMAx_CLK_ENABLE();
00577
00578   if(hi2s->Instance == I2SOUT)
00579   {
00580     /* Configure the hdma_i2stx handle param
eters */
00581     hdma_i2stx.Init.Direction           = DM
A_MEMORY_TO_PERIPH;
00582     hdma_i2stx.Init.PeriphInc           = DM
A_PINC_DISABLE;
00583     hdma_i2stx.Init.MemInc              = DM
A_MINC_ENABLE;
00584     hdma_i2stx.Init.PeriphDataAlignment = I2
SOUT_DMAx_PERIPH_DATA_SIZE;
00585     hdma_i2stx.Init.MemDataAlignment    = I2
SOUT_DMAx_MEM_DATA_SIZE;
00586     /* Configuration change depending on sen
ding dummy data or audio data on bus I2S */
00587     if (SendDummyData != ENABLE)
00588     {
00589       hdma_i2stx.Init.Mode              = DM
A_NORMAL;
00590     }
00591     else
00592     {
00593       hdma_i2stx.Init.Mode              = DM
A_CIRCULAR;
00594     }
00595     hdma_i2stx.Init.Priority            = DM
A_PRIORITY_HIGH;
```

```
00596
00597     hdma_i2stx.Instance                = I2
SOUT_DMAx_CHANNEL;
00598
00599     /* Associate the DMA handle */
00600     __HAL_LINKDMA(hi2s, hdmatx, hdma_i2stx);
00601
00602     /* Deinitialize the Channel for new tran
sfer */
00603     HAL_DMA_DeInit(&hdma_i2stx);
00604
00605     /* Configure the DMA Channel */
00606     HAL_DMA_Init(&hdma_i2stx);
00607   }
00608
00609   /* I2S DMA IRQ Channel configuration */
00610   HAL_NVIC_SetPriority(I2SOUT_DMAx_IRQ, AUDI
O_OUT_IRQ_PREPRIO, 0);
00611   HAL_NVIC_EnableIRQ(I2SOUT_DMAx_IRQ);
00612 }
00613
00614 /**
00615   * @brief  Initializes the Audio Codec audi
o interface (I2S)
00616   * @param  AudioFreq: Audio frequency to be
 configured for the I2S peripheral.
00617   */
00618 static void I2SOUT_Init(uint32_t AudioFreq)
00619 {
00620   /* Initialize the hAudioOutI2s Instance pa
rameter */
00621   hAudioOutI2s.Instance         = I2SOUT;
00622
00623  /* Disable I2S block */
00624   __HAL_I2S_DISABLE(&hAudioOutI2s);
00625
00626   /* Perform MSP initialization at first fun
```

```
ction call */
00627   if(HAL_I2S_GetState(&hAudioOutI2s) == HAL_
I2S_STATE_RESET)
00628   {
00629     I2SOUT_MspInit();
00630   }
00631
00632   /* I2SOUT peripheral configuration */
00633   hAudioOutI2s.Init.AudioFreq   = AudioFreq;
00634   hAudioOutI2s.Init.CPOL        = I2S_CPOL_L
OW;
00635   hAudioOutI2s.Init.DataFormat  = I2S_DATAFO
RMAT_16B;
00636   hAudioOutI2s.Init.MCLKOutput  = I2S_MCLKOU
TPUT_ENABLE;
00637   hAudioOutI2s.Init.Mode        = I2S_MODE_M
ASTER_TX;
00638   hAudioOutI2s.Init.Standard    = I2S_STANDA
RD;
00639
00640   /* Initialize the I2S peripheral with the
structure above */
00641   HAL_I2S_Init(&hAudioOutI2s);
00642
00643 }
00644
00645 /**
00646   * @brief  Starts sending dummy data as aud
io stream: function used to
00647             generate I2S master clock MCK fo
r audio codec device.
00648   * @param  pBuffer: Pointer to the buffer
00649   * @param  Size: Number of audio data BYTES.

00650   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00651   */
```

```
00652 static uint8_t I2SOUT_SendDummyData_Start(ui
nt16_t* pBuffer, uint32_t Size)
00653 {
00654   /* Update the Media layer and enable it fo
r play */
00655   HAL_I2S_Transmit_DMA(&hAudioOutI2s, pBuffe
r, DMA_MAX(Size));
00656
00657   return AUDIO_OK;
00658
00659 }
00660
00661 /**
00662   * @brief  Stops sending dummy data as audi
o stream: function used to stop to
00663             generate I2S master clock MCK fo
r audio codec device.
00664   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00665   */
00666 static uint8_t I2SOUT_SendDummyData_Stop(void
)
00667 {
00668   /* Call DMA Stop to disable DMA stream bef
ore stopping codec */
00669   HAL_I2S_DMAStop(&hAudioOutI2s);
00670
00671   return AUDIO_OK;
00672
00673 }
00674
00675
00676 /**
00677   * @}
00678   */
00679
00680 /**
```

```
00681   * @}
00682   */
00683
00684 /**
00685   * @}
00686   */
00687
00688 /**
00689   * @}
00690   */
00691
00692 /**
00693   * @}
00694   */
00695
00696 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210E_EVAL BSP User Manual

stm3210e_eval_nand.h

Go to the documentation of this file.

```
00001 /**
00002   ********************************************************************************
00003   * @file    stm3210e_eval_nand.h
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm3210e_eval_nand.c driver.
00009   ********************************************************************************
00010   * @attention
00011   *
00012   * <h2><center>&copy; COPYRIGHT(c) 2015 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 __STM3210E_EVAL_NAND_H
00041 #define __STM3210E_EVAL_NAND_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------
-----------------------------------*/
00048 #include "stm32f1xx_hal.h"
00049
00050 /** @addtogroup BSP
00051   * @{
00052   */
00053
00054 /** @addtogroup STM3210E_EVAL
00055   * @{
00056   */
00057
00058 /** @addtogroup STM3210E_EVAL_NAND
00059   * @{
00060   */
00061
00062 /* Exported types ------------------------
-----------------------------------*/
00063
00064 /** @defgroup STM3210E_EVAL_NAND_Exported_Ty
pes Exported_Types
00065   * @{
00066   */
```

```
00067
00068 /**
00069   * @}
00070   */
00071
00072
00073 /* Exported constants ---------------------
---------------------------------*/
00074 /** @defgroup STM3210E_EVAL_NAND_Exported_Co
nstants Exported_Constants
00075   * @{
00076   */
00077
00078 /**
00079   * @brief  NAND status structure definition

00080   */
00081 #define   NAND_OK         0x00
00082
00083 #define NAND_DEVICE_ADDR  ((uint32_t)NAND_DE
VICE1)
00084
00085 /**
00086   * @brief  FSMC NAND memory parameters
00087   */
00088 #define NAND_PAGE_SIZE             ((uint16_
t)0x0200) /* 512 bytes per page w/o Spare Area */
00089 #define NAND_BLOCK_SIZE            ((uint16_
t)0x0020) /* 32x512 bytes pages per block */
00090 #define NAND_ZONE_SIZE             ((uint16_
t)0x0400) /* 1024 Block per zone */
00091 #define NAND_SPARE_AREA_SIZE       ((uint16_
t)0x0010) /* last 16 bytes as spare area */
00092 #define NAND_MAX_ZONE              ((uint16_
t)0x0004) /* 4 zones of 1024 block */
00093
00094 /**
```

```
00095   * @}
00096   */
00097
00098 /* Exported macro ------------------------
-----------------------------------*/
00099
00100 /** @defgroup STM3210E_EVAL_NAND_Exported_Ma
cro Exported_Macro
00101   * @{
00102   */
00103
00104 /**
00105   * @}
00106   */
00107
00108 /* Exported functions --------------------
-----------------------------------*/
00109
00110 /** @addtogroup STM3210E_EVAL_NAND_Exported_
Functions
00111   * @{
00112   */
00113 uint8_t BSP_NAND_Init(void);
00114 uint8_t BSP_NAND_ReadData(NAND_AddressTypeDe
f BlockAddress, uint8_t *pData, uint32_t uwDataSiz
e);
00115 uint8_t BSP_NAND_WriteData(NAND_AddressTypeD
ef BlockAddress, uint8_t *pData, uint32_t uwDataSi
ze);
00116 uint8_t BSP_NAND_Erase_Block(NAND_AddressTyp
eDef BlockAddress);
00117 uint8_t BSP_NAND_Erase_Chip(void);
00118 uint8_t BSP_NAND_Read_ID(NAND_IDTypeDef *pNA
ND_ID);
00119
00120 /**
00121   * @}
```

```
00122   */
00123
00124 /**
00125   * @}
00126   */
00127
00128 /**
00129   * @}
00130   */
00131
00132 /**
00133   * @}
00134   */
00135
00136 #ifdef __cplusplus
00137 }
00138 #endif
00139
00140 #endif /* __STM3210E_EVAL_NAND_H */
00141
00142 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
00143
```

STM3210E_EVAL BSP User Manual

stm3210e_eval_nand.c

Go to the documentation of this file.

```
00001 /**
00002   ********************************************************************************
00003   * @file    stm3210e_eval_nand.c
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file includes a standard driver for the NAND512W3A2CN6E NAND flash memory
00008   *          device mounted on STM3210E-EVAL evaluation board.
00009   ********************************************************************************
00010   * @attention
00011   *
00012   * <h2><center>&copy; COPYRIGHT(c) 2015 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 /* File Info : ----------------------------
-----------------------------------
00040                                       User NOTES

00041 1. How To use this driver:
00042 ---------------------------
00043    - This driver is used to drive the NAND51
2W3A2CN6E NAND flash external memory mounted
00044      on STM3210E-EVAL evaluation board.
00045    - This driver does not need a specific co
mponent driver for the NAND device
00046      to be included with.
00047
00048 2. Driver description:
00049 ---------------------
00050   + Initialization steps:
00051      o Initialize the NAND external memory u
sing the BSP_NAND_Init() function. This
00052        function includes the MSP layer hardw
are resources initialization and the
00053        FMC controller configuration to inter
face with the external NAND memory.
00054
00055   + NAND flash operations
00056      o NAND external memory can be accessed
with read/write operations once it is
00057        initialized.
00058        Read/write operation can be performed
 with AHB access using the functions
00059        BSP_NAND_ReadData()/BSP_NAND_WriteDat
a(). The BSP_NAND_WriteData() performs write opera
tion
```

```
00060         of an amount of data by unit (halfwor
d).
00061       o The function BSP_NAND_Read_ID() retur
ns the chip IDs stored in the structure
00062         "NAND_IDTypeDef". (see the NAND IDs i
n the memory data sheet)
00063       o Perform erase block operation using t
he function BSP_NAND_Erase_Block() and by
00064         specifying the block address.
00065
00066 ------------------------------------------
-----------------------------------*/
00067
00068 /* Includes ---------------------------------
-----------------------------------*/
00069 #include "stm3210e_eval_nand.h"
00070
00071 /** @addtogroup BSP
00072   * @{
00073   */
00074
00075 /** @addtogroup STM3210E_EVAL
00076   * @{
00077   */
00078
00079 /** @defgroup STM3210E_EVAL_NAND STM3210E_EV
AL NAND
00080   * @{
00081   */
00082
00083 /* Private typedef -------------------------
-----------------------------------*/
00084
00085 /** @defgroup STM3210E_EVAL_NAND_Private_Typ
es_Definitions Private_Types_Definitions
00086   * @{
00087   */
```

```
00088
00089 /**
00090   * @}
00091   */
00092
00093 /* Private define ------------------------
-----------------------------------*/
00094
00095 /** @defgroup STM3210E_EVAL_NAND_Private_Def
ines Private_Defines
00096   * @{
00097   */
00098
00099 /**
00100   * @}
00101   */
00102
00103 /* Private macro -----------------------------
-----------------------------------*/
00104
00105 /** @defgroup STM3210E_EVAL_NAND_Private_Mac
ros Private_Macros
00106   * @{
00107   */
00108
00109 /**
00110   * @}
00111   */
00112
00113 /* Private variables ------------------------
-----------------------------------*/
00114
00115 /** @defgroup STM3210E_EVAL_NAND_Private_Var
iables Private_Variables
00116   * @{
00117   */
00118 static NAND_HandleTypeDef nandHandle;
```

```
00119 static FSMC_NAND_PCC_TimingTypeDef Timing;
00120
00121 /**
00122   * @}
00123   */
00124
00125 /* Private function prototypes -------------
-----------------------------------*/
00126
00127 /** @defgroup STM3210E_EVAL_NAND_Private_Fun
ction_Prototypes Private_Function_Prototypes
00128   * @{
00129   */
00130 static void NAND_MspInit(void);
00131
00132 /**
00133   * @}
00134   */
00135
00136 /* Private functions -----------------------
-----------------------------------*/
00137
00138 /** @defgroup STM3210E_EVAL_NAND_Exported_Fu
nctions Exported_Functions
00139   * @{
00140   */
00141
00142 /**
00143   * @brief  Initializes the NAND device.
00144   * @retval NAND memory status
00145   */
00146 uint8_t BSP_NAND_Init(void)
00147 {
00148   nandHandle.Instance  = FSMC_NAND_DEVICE;
00149
00150   /*NAND Configuration */
00151   Timing.SetupTime     = 0;
```

```
00152   Timing.WaitSetupTime = 2;
00153   Timing.HoldSetupTime = 1;
00154   Timing.HiZSetupTime  = 0;
00155
00156   nandHandle.Init.NandBank        = FSMC_NAN
D_BANK2;
00157   nandHandle.Init.Waitfeature     = FSMC_NAN
D_PCC_WAIT_FEATURE_ENABLE;
00158   nandHandle.Init.MemoryDataWidth = FSMC_NAN
D_PCC_MEM_BUS_WIDTH_8;
00159   nandHandle.Init.EccComputation  = FSMC_NAN
D_ECC_ENABLE;
00160   nandHandle.Init.ECCPageSize     = FSMC_NAN
D_ECC_PAGE_SIZE_512BYTE;
00161   nandHandle.Init.TCLRSetupTime   = 0;
00162   nandHandle.Init.TARSetupTime    = 0;
00163
00164   nandHandle.Info.BlockNbr      = NAND_MAX_Z
ONE;
00165   nandHandle.Info.BlockSize     = NAND_BLOCK
_SIZE;
00166   nandHandle.Info.ZoneSize      = NAND_ZONE_
SIZE;
00167   nandHandle.Info.PageSize      = NAND_PAGE_
SIZE;
00168   nandHandle.Info.SpareAreaSize = NAND_SPARE
_AREA_SIZE;
00169
00170   /* NAND controller initialization */
00171   NAND_MspInit();
00172
00173   if(HAL_NAND_Init(&nandHandle, &Timing, &Ti
ming) != HAL_OK)
00174   {
00175     return NAND_ERROR;
00176   }
00177   else
```

```
00178   {
00179     return NAND_OK;
00180   }
00181 }
00182
00183 /**
00184   * @brief  Reads an amount of data from the
 NAND device.
00185   * @param  BlockAddress: Block address to R
ead
00186   * @param  pData: Pointer to data to be read

00187   * @param  uwNumPage: Number of Pages to re
ad to Block
00188   * @retval NAND memory status
00189   */
00190 uint8_t BSP_NAND_ReadData(NAND_AddressTypeDe
f BlockAddress, uint8_t* pData, uint32_t uwNumPage
)
00191 {
00192   /* Read data from NAND */
00193   if (HAL_NAND_Read_Page(&nandHandle, &Block
Address, pData, uwNumPage) != HAL_OK)
00194   {
00195     return NAND_ERROR;
00196   }
00197
00198   return NAND_OK;
00199 }
00200
00201 /**
00202   * @brief  Writes an amount of data to the
NAND device.
00203   * @param  BlockAddress: Block address to W
rite
00204   * @param  pData: Pointer to data to be wri
tten
```

```
00205   * @param  uwNumPage: Number of Pages to wr
ite to Block
00206   * @retval NAND memory status
00207   */
00208 uint8_t BSP_NAND_WriteData(NAND_AddressTypeD
ef BlockAddress, uint8_t* pData, uint32_t uwNumPag
e)
00209 {
00210   /* Write data to NAND */
00211   if (HAL_NAND_Write_Page(&nandHandle, &Bloc
kAddress, pData, uwNumPage) != HAL_OK)
00212   {
00213     return NAND_ERROR;
00214   }
00215
00216   return NAND_OK;
00217 }
00218
00219 /**
00220   * @brief  Erases the specified block of th
e NAND device.
00221   * @param  BlockAddress: Block address to E
rase
00222   * @retval NAND memory status
00223   */
00224 uint8_t BSP_NAND_Erase_Block(NAND_AddressTyp
eDef BlockAddress)
00225 {
00226   /* Send NAND erase block operation */
00227   if (HAL_NAND_Erase_Block(&nandHandle, &Blo
ckAddress) != HAL_OK)
00228   {
00229     return NAND_ERROR;
00230   }
00231
00232   return NAND_OK;
00233 }
```

```
00234
00235 /**
00236   * @brief  Reads NAND flash IDs.
00237   * @param  pNAND_ID : Pointer to NAND ID st
ructure
00238   * @retval NAND memory status
00239   */
00240 uint8_t BSP_NAND_Read_ID(NAND_IDTypeDef *pNA
ND_ID)
00241 {
00242   if(HAL_NAND_Read_ID(&nandHandle, pNAND_ID)
 != HAL_OK)
00243   {
00244     return NAND_ERROR;
00245   }
00246   else
00247   {
00248     return NAND_OK;
00249   }
00250 }
00251
00252 /**
00253   * @brief  Initializes the NAND MSP.
00254   * @retval None
00255   */
00256 static void NAND_MspInit(void)
00257 {
00258   GPIO_InitTypeDef gpioinitstruct = {0};
00259
00260   /* Enable FSMC clock */
00261   __HAL_RCC_FSMC_CLK_ENABLE();
00262
00263   /* Enable GPIOs clock */
00264   __HAL_RCC_GPIOD_CLK_ENABLE();
00265   __HAL_RCC_GPIOE_CLK_ENABLE();
00266   __HAL_RCC_GPIOF_CLK_ENABLE();
00267   __HAL_RCC_GPIOG_CLK_ENABLE();
```

```
00268
00269   /* Common GPIO configuration */
00270   gpioinitstruct.Mode      = GPIO_MODE_AF_PP
;
00271   gpioinitstruct.Pull      = GPIO_PULLUP;
00272   gpioinitstruct.Speed     = GPIO_SPEED_FREQ
_HIGH;
00273
00274 /*-- GPIO Configuration ------------------
---------------------------------*/
00275 /*!< CLE, ALE, D0->D3, NOE, NWE and NCE2  NA
ND pin configuration  */
00276   gpioinitstruct.Pin =  GPIO_PIN_11 | GPIO_P
IN_12 | GPIO_PIN_14 | GPIO_PIN_15 |
00277                                        GPIO_PIN_0
| GPIO_PIN_1 | GPIO_PIN_4 | GPIO_PIN_5 |
00278                                        GPIO_PIN_7;
00279   HAL_GPIO_Init(GPIOD, &gpioinitstruct);
00280
00281 /*!< D4->D7 NAND pin configuration  */
00282   gpioinitstruct.Pin = GPIO_PIN_7 | GPIO_PIN
_8 | GPIO_PIN_9 | GPIO_PIN_10;
00283
00284   HAL_GPIO_Init(GPIOE, &gpioinitstruct);
00285
00286
00287 /*!< NWAIT NAND pin configuration */
00288   gpioinitstruct.Pin = GPIO_PIN_6;
00289   gpioinitstruct.Mode      = GPIO_MODE_INPUT
;
00290
00291   HAL_GPIO_Init(GPIOD, &gpioinitstruct);
00292
00293 /*!< INT2 NAND pin configuration */
00294   gpioinitstruct.Pin = GPIO_PIN_6;
00295   HAL_GPIO_Init(GPIOG, &gpioinitstruct);
00296 }
```

```
00297
00298 /**
00299   * @}
00300   */
00301
00302 /**
00303   * @}
00304   */
00305
00306 /**
00307   * @}
00308   */
00309
00310 /**
00311   * @}
00312   */
00313
00314 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210E_EVAL BSP User Manual

stm3210e_eval_nor.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210e_eval_nor.h
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm3210e_eval_nor.c driver.
00009   ******************************************************************************
00010   * @attention
00011   *
00012   * <h2><center>&copy; COPYRIGHT(c) 2015 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 __STM3210E_EVAL_NOR_H
00041 #define __STM3210E_EVAL_NOR_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------------------------------------------*/
00048 #include "stm32f1xx_hal.h"
00049
00050 /** @addtogroup BSP
00051   * @{
00052   */
00053
00054 /** @addtogroup STM3210E_EVAL
00055   * @{
00056   */
00057
00058 /** @addtogroup STM3210E_EVAL_NOR
00059   * @{
00060   */
00061
00062 /* Exported types ------------------------------------------------------------*/
00063
00064 /** @defgroup STM3210E_EVAL_NOR_Exported_Types Exported_Types
00065   * @{
00066   */
00067
```

```
00068 /**
00069   * @}
00070   */
00071
00072 /* Exported constants ----------------------
------------------------------------*/
00073 /** @defgroup STM3210E_EVAL_NOR_Exported_Con
stants Exported_Constants
00074   * @{
00075   */
00076
00077 /**
00078   * @brief  NOR status structure definition

00079   */
00080 #define   NOR_STATUS_OK         0x00
00081 #define   NOR_STATUS_ERROR      0x01
00082
00083 #define NOR_DEVICE_ADDR  ((uint32_t)0x640000
00)
00084
00085 /* #define NOR_MEMORY_WIDTH    FSMC_NORSRAM_
MEM_BUS_WIDTH_8  */
00086 #define NOR_MEMORY_WIDTH    FSMC_NORSRAM_MEM
_BUS_WIDTH_16
00087
00088 #define NOR_BURSTACCESS    FSMC_BURST_ACCESS
_MODE_DISABLE
00089 /* #define NOR_BURSTACCESS    FSMC_BURST_ACC
ESS_MODE_ENABLE*/
00090
00091 #define NOR_WRITEBURST    FSMC_WRITE_BURST_D
ISABLE
00092 /* #define NOR_WRITEBURST   FSMC_WRITE_BURST
_ENABLE */
00093
00094 /* NOR operations Timeout definitions */
```

```
00095 #define BLOCKERASE_TIMEOUT   ((uint32_t)0x00
A00000)  /* NOR block erase timeout */
00096 #define CHIPERASE_TIMEOUT    ((uint32_t)0x30
000000)  /* NOR chip erase timeout  */
00097 #define PROGRAM_TIMEOUT      ((uint32_t)0x00
004400)  /* NOR program timeout     */
00098
00099 /* NOR Ready/Busy signal GPIO definitions */
00100 #define NOR_READY_BUSY_PIN    GPIO_PIN_6
00101 #define NOR_READY_BUSY_GPIO   GPIOD
00102 #define NOR_READY_STATE       GPIO_PIN_SET
00103 #define NOR_BUSY_STATE        GPIO_PIN_RESET

00104
00105 /**
00106   * @}
00107   */
00108
00109 /* Exported macro ------------------------
-----------------------------------*/
00110
00111 /** @defgroup STM3210E_EVAL_NOR_Exported_Mac
ro Exported_Macro
00112   * @{
00113   */
00114
00115 /**
00116   * @}
00117   */
00118
00119 /* Exported functions ---------------------
-----------------------------------*/
00120
00121 /** @addtogroup STM3210E_EVAL_NOR_Exported_F
unctions
00122   * @{
00123   */
```

```
00124 uint8_t BSP_NOR_Init(void);
00125 uint8_t BSP_NOR_ReadData(uint32_t uwStartAdd
ress, uint16_t *pData, uint32_t uwDataSize);
00126 uint8_t BSP_NOR_WriteData(uint32_t uwStartAd
dress, uint16_t *pData, uint32_t uwDataSize);
00127 uint8_t BSP_NOR_ProgramData(uint32_t uwStart
Address, uint16_t *pData, uint32_t uwDataSize);
00128 uint8_t BSP_NOR_Erase_Block(uint32_t BlockAd
dress);
00129 uint8_t BSP_NOR_Erase_Chip(void);
00130 uint8_t BSP_NOR_Read_ID(NOR_IDTypeDef *pNOR_
ID);
00131 void BSP_NOR_ReturnToReadMode(void);
00132
00133 /**
00134   * @}
00135   */
00136
00137 /**
00138   * @}
00139   */
00140
00141 /**
00142   * @}
00143   */
00144
00145 /**
00146   * @}
00147   */
00148
00149 #ifdef __cplusplus
00150 }
00151 #endif
00152
00153 #endif /* __STM3210E_EVAL_NOR_H */
00154
00155 /************************ (C) COPYRIGHT STMi
```

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

STM3210E_EVAL BSP User Manual

stm3210e_eval_nor.c

Go to the documentation of this file.

```
00001 /**
00002   ********************************************************************************
00003   * @file    stm3210e_eval_nor.c
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file includes a standard driver for the M29W128GL70ZA6E NOR flash memory
00008   *          device mounted on STM3210E-EVAL evaluation board.
00009   ********************************************************************************
00010   * @attention
00011   *
00012   * <h2><center>&copy; COPYRIGHT(c) 2015 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 /* File Info : ----------------------------
-----------------------------------
00040                                       User NOTES

00041 1. How To use this driver:
00042 --------------------------
00043    - This driver is used to drive the M29W12
8GL NOR flash external memory mounted
00044      on STM3210E-EVAL evaluation board.
00045    - This driver does not need a specific co
mponent driver for the NOR device
00046      to be included with.
00047
00048 2. Driver description:
00049 ---------------------
00050   + Initialization steps:
00051      o Initialize the NOR external memory us
ing the BSP_NOR_Init() function. This
00052        function includes the MSP layer hardw
are resources initialization and the
00053        FSMC controller configuration to inte
rface with the external NOR memory.
00054
00055   + NOR flash operations
00056      o NOR external memory can be accessed w
ith read/write operations once it is
00057        initialized.
00058        Read/write operation can be performed
 with AHB access using the functions
00059        BSP_NOR_ReadData()/BSP_NOR_WriteData(
). The BSP_NOR_WriteData() performs write operation
```

```
00060         of an amount of data by unit (halfwor
d). You can also perform a program data
00061         operation of an amount of data using
the function BSP_NOR_ProgramData().
00062       o The function BSP_NOR_Read_ID() return
s the chip IDs stored in the structure
00063         "NOR_IDTypeDef". (see the NOR IDs in
the memory data sheet)
00064       o Perform erase block operation using t
he function BSP_NOR_Erase_Block() and by
00065         specifying the block address. You can
 perform an erase operation of the whole
00066         chip by calling the function BSP_NOR_
Erase_Chip().
00067       o After other operations, the function
BSP_NOR_ReturnToReadMode() allows the NOR
00068         flash to return to read mode to perfo
rm read operations on it.
00069
00070 ------------------------------------------
-----------------------------------*/
00071
00072 /* Includes ---------------------------------
-----------------------------------*/
00073 #include "stm3210e_eval_nor.h"
00074
00075 /** @addtogroup BSP
00076   * @{
00077   */
00078
00079 /** @addtogroup STM3210E_EVAL
00080   * @{
00081   */
00082
00083 /** @defgroup STM3210E_EVAL_NOR STM3210E_EVA
L NOR
00084   * @{
```

```
00085   */
00086
00087 /* Private typedef -----------------------
-----------------------------------*/
00088
00089 /** @defgroup STM3210E_EVAL_NOR_Private_Type
s_Definitions Private_Types_Definitions
00090   * @{
00091   */
00092
00093 /**
00094   * @}
00095   */
00096
00097 /* Private define ------------------------
-----------------------------------*/
00098
00099 /** @defgroup STM3210E_EVAL_NOR_Private_Defi
nes Private_Defines
00100   * @{
00101   */
00102
00103 /**
00104   * @}
00105   */
00106
00107 /* Private macro -------------------------
-----------------------------------*/
00108
00109 /** @defgroup STM3210E_EVAL_NOR_Private_Macr
os Private_Macros
00110   * @{
00111   */
00112
00113 /**
00114   * @}
00115   */
```

```
00116
00117 /* Private variables ---------------------
-----------------------------------*/
00118
00119 /** @defgroup STM3210E_EVAL_NOR_Private_Vari
ables Private_Variables
00120   * @{
00121   */
00122 static NOR_HandleTypeDef norHandle;
00123 static FSMC_NORSRAM_TimingTypeDef Timing;
00124
00125 /**
00126   * @}
00127   */
00128
00129 /* Private function prototypes -----------
-----------------------------------*/
00130
00131 /** @defgroup STM3210E_EVAL_NOR_Private_Func
tion_Prototypes Private_Function_Prototypes
00132   * @{
00133   */
00134
00135 static void NOR_MspInit(void);
00136
00137 /**
00138   * @}
00139   */
00140
00141 /* Private functions ---------------------
-----------------------------------*/
00142
00143 /** @defgroup STM3210E_EVAL_NOR_Exported_Fun
ctions Exported_Functions
00144   * @{
00145   */
00146
```

```
00147 /**
00148   * @brief  Initializes the NOR device.
00149   * @retval NOR memory status
00150   */
00151 uint8_t BSP_NOR_Init(void)
00152 {
00153   norHandle.Instance  = FSMC_NORSRAM_DEVICE;
00154   norHandle.Extended  = FSMC_NORSRAM_EXTENDE
D_DEVICE;
00155
00156   /* NOR device configuration */
00157   Timing.AddressSetupTime      = 2;
00158   Timing.AddressHoldTime       = 1;
00159   Timing.DataSetupTime         = 5;
00160   Timing.BusTurnAroundDuration = 0;
00161   Timing.CLKDivision           = 2;
00162   Timing.DataLatency           = 2;
00163   Timing.AccessMode            = FSMC_ACCESS
_MODE_B;
00164
00165   norHandle.Init.NSBank             = FSMC_N
ORSRAM_BANK2;
00166   norHandle.Init.DataAddressMux     = FSMC_D
ATA_ADDRESS_MUX_DISABLE;
00167   norHandle.Init.MemoryType         = FSMC_M
EMORY_TYPE_NOR;
00168   norHandle.Init.MemoryDataWidth    = NOR_ME
MORY_WIDTH;
00169   norHandle.Init.BurstAccessMode    = NOR_BU
RSTACCESS;
00170   norHandle.Init.WaitSignalPolarity = FSMC_W
AIT_SIGNAL_POLARITY_LOW;
00171   norHandle.Init.WrapMode           = FSMC_W
RAP_MODE_DISABLE;
00172   norHandle.Init.WaitSignalActive   = FSMC_W
AIT_TIMING_BEFORE_WS;
00173   norHandle.Init.WriteOperation     = FSMC_W
```

```
RITE_OPERATION_ENABLE;
00174   norHandle.Init.WaitSignal          = FSMC_W
AIT_SIGNAL_DISABLE;
00175   norHandle.Init.ExtendedMode        = FSMC_E
XTENDED_MODE_DISABLE;
00176   norHandle.Init.AsynchronousWait    = FSMC_A
SYNCHRONOUS_WAIT_DISABLE;
00177   norHandle.Init.WriteBurst          = NOR_WR
ITEBURST;
00178
00179   /* NOR controller initialization */
00180   NOR_MspInit();
00181
00182   if(HAL_NOR_Init(&norHandle, &Timing, &Timi
ng) != HAL_OK)
00183   {
00184     return NOR_STATUS_ERROR;
00185   }
00186   else
00187   {
00188     return NOR_STATUS_OK;
00189   }
00190 }
00191
00192 /**
00193   * @brief  Reads an amount of data from the
 NOR device.
00194   * @param  uwStartAddress: Read start addre
ss
00195   * @param  pData: Pointer to data to be read

00196   * @param  uwDataSize: Size of data to read

00197   * @retval NOR memory status
00198   */
00199 uint8_t BSP_NOR_ReadData(uint32_t uwStartAdd
ress, uint16_t* pData, uint32_t uwDataSize)
```

```
00200 {
00201   if(HAL_NOR_ReadBuffer(&norHandle, NOR_DEVI
CE_ADDR + uwStartAddress, pData, uwDataSize) != HA
L_OK)
00202   {
00203     return NOR_STATUS_ERROR;
00204   }
00205   else
00206   {
00207     return NOR_STATUS_OK;
00208   }
00209 }
00210
00211 /**
00212   * @brief  Returns the NOR memory to read m
ode.
00213   * @retval None
00214   */
00215 void BSP_NOR_ReturnToReadMode(void)
00216 {
00217    HAL_NOR_ReturnToReadMode(&norHandle);
00218 }
00219
00220 /**
00221   * @brief  Writes an amount of data to the
NOR device.
00222   * @param  uwStartAddress: Write start addr
ess
00223   * @param  pData: Pointer to data to be wri
tten
00224   * @param  uwDataSize: Size of data to writ
e
00225   * @retval NOR memory status
00226   */
00227 uint8_t BSP_NOR_WriteData(uint32_t uwStartAd
dress, uint16_t* pData, uint32_t uwDataSize)
00228 {
```

```
00229   uint32_t index = uwDataSize;
00230 
00231   while(index > 0)
00232   {
00233     /* Write data to NOR */
00234     HAL_NOR_Program(&norHandle, (uint32_t *)(
NOR_DEVICE_ADDR + uwStartAddress), pData);
00235 
00236     /* Read NOR device status */
00237     if(HAL_NOR_GetStatus(&norHandle, NOR_DEV
ICE_ADDR, PROGRAM_TIMEOUT) != HAL_NOR_STATUS_SUCCE
SS)
00238     {
00239       return NOR_STATUS_ERROR;
00240     }
00241 
00242     /* Update the counters */
00243     index--;
00244     uwStartAddress += 2;
00245     pData++;
00246   }
00247 
00248   return NOR_STATUS_OK;
00249 }
00250 
00251 /**
00252   * @brief  Programs an amount of data to th
e NOR device.
00253   * @param  uwStartAddress: Write start addr
ess
00254   * @param  pData: Pointer to data to be wri
tten
00255   * @param  uwDataSize: Size of data to writ
e
00256   * @retval NOR memory status
00257   */
00258 uint8_t BSP_NOR_ProgramData(uint32_t uwStart
```

```
Address, uint16_t* pData, uint32_t uwDataSize)
00259 {
00260   /* Send NOR program buffer operation */
00261   HAL_NOR_ProgramBuffer(&norHandle, uwStartA
ddress, pData, uwDataSize);
00262
00263   /* Return the NOR memory status */
00264   if(HAL_NOR_GetStatus(&norHandle, NOR_DEVIC
E_ADDR, PROGRAM_TIMEOUT) != HAL_NOR_STATUS_SUCCESS
)
00265   {
00266     return NOR_STATUS_ERROR;
00267   }
00268   else
00269   {
00270     return NOR_STATUS_OK;
00271   }
00272 }
00273
00274 /**
00275   * @brief  Erases the specified block of th
e NOR device.
00276   * @param  BlockAddress: Block address to e
rase
00277   * @retval NOR memory status
00278   */
00279 uint8_t BSP_NOR_Erase_Block(uint32_t BlockAd
dress)
00280 {
00281   /* Send NOR erase block operation */
00282   HAL_NOR_Erase_Block(&norHandle, BlockAddre
ss, NOR_DEVICE_ADDR);
00283
00284   /* Return the NOR memory status */
00285   if(HAL_NOR_GetStatus(&norHandle, NOR_DEVIC
E_ADDR, BLOCKERASE_TIMEOUT) != HAL_NOR_STATUS_SUCC
ESS)
```

```
00286   {
00287     return NOR_STATUS_ERROR;
00288   }
00289   else
00290   {
00291     return NOR_STATUS_OK;
00292   }
00293 }
00294
00295 /**
00296   * @brief  Erases the entire NOR chip.
00297   * @retval NOR memory status
00298   */
00299 uint8_t BSP_NOR_Erase_Chip(void)
00300 {
00301   /* Send NOR Erase chip operation */
00302   HAL_NOR_Erase_Chip(&norHandle, NOR_DEVICE_
ADDR);
00303
00304   /* Return the NOR memory status */
00305   if(HAL_NOR_GetStatus(&norHandle, NOR_DEVIC
E_ADDR, CHIPERASE_TIMEOUT) != HAL_NOR_STATUS_SUCCE
SS)
00306   {
00307     return NOR_STATUS_ERROR;
00308   }
00309   else
00310   {
00311     return NOR_STATUS_OK;
00312   }
00313 }
00314
00315 /**
00316   * @brief  Reads NOR flash IDs.
00317   * @param  pNOR_ID : Pointer to NOR ID stru
cture
00318   * @retval NOR memory status
```

```
00319   */
00320 uint8_t BSP_NOR_Read_ID(NOR_IDTypeDef *pNOR_
ID)
00321 {
00322   if(HAL_NOR_Read_ID(&norHandle, pNOR_ID) !=
 HAL_OK)
00323   {
00324     return NOR_STATUS_ERROR;
00325   }
00326   else
00327   {
00328     return NOR_STATUS_OK;
00329   }
00330 }
00331
00332 /**
00333   * @brief  Initializes the NOR MSP.
00334   * @retval None
00335   */
00336 static void NOR_MspInit(void)
00337 {
00338   GPIO_InitTypeDef gpioinitstruct = {0};
00339
00340   /* Enable FSMC clock */
00341   __HAL_RCC_FSMC_CLK_ENABLE();
00342
00343   /* Enable GPIOs clock */
00344   __HAL_RCC_GPIOD_CLK_ENABLE();
00345   __HAL_RCC_GPIOE_CLK_ENABLE();
00346   __HAL_RCC_GPIOF_CLK_ENABLE();
00347   __HAL_RCC_GPIOG_CLK_ENABLE();
00348
00349   /* Common GPIO configuration */
00350   gpioinitstruct.Mode      = GPIO_MODE_AF_PP
;
00351   gpioinitstruct.Pull      = GPIO_PULLUP;
00352   gpioinitstruct.Speed     = GPIO_SPEED_FREQ
```

```
_HIGH;
00353
00354   /*-- GPIO Configuration ------------------
-------------------------------------*/
00355   /*!< NOR Data lines configuration */
00356   gpioinitstruct.Pin = GPIO_PIN_0 | GPIO_PIN
_1 | GPIO_PIN_8 | GPIO_PIN_9 |
00357                                        GPIO_PIN_10
| GPIO_PIN_14 | GPIO_PIN_15;
00358   HAL_GPIO_Init(GPIOD, &gpioinitstruct);
00359
00360   gpioinitstruct.Pin = GPIO_PIN_7 | GPIO_PIN
_8 | GPIO_PIN_9 | GPIO_PIN_10 |
00361                                        GPIO_PIN_11
| GPIO_PIN_12 | GPIO_PIN_13 |
00362                                        GPIO_PIN_14
| GPIO_PIN_15;
00363   HAL_GPIO_Init(GPIOE, &gpioinitstruct);
00364
00365   /*!< NOR Address lines configuration */
00366   gpioinitstruct.Pin = GPIO_PIN_0 | GPIO_PIN
_1 | GPIO_PIN_2 | GPIO_PIN_3 |
00367                                        GPIO_PIN_4 |
 GPIO_PIN_5 | GPIO_PIN_12 | GPIO_PIN_13 |
00368                                        GPIO_PIN_14
| GPIO_PIN_15;
00369   HAL_GPIO_Init(GPIOF, &gpioinitstruct);
00370
00371   gpioinitstruct.Pin = GPIO_PIN_0 | GPIO_PIN
_1 | GPIO_PIN_2 |
00372                                        GPIO_PIN_3 |
 GPIO_PIN_4 | GPIO_PIN_5;
00373   HAL_GPIO_Init(GPIOG, &gpioinitstruct);
00374
00375   gpioinitstruct.Pin = GPIO_PIN_11 | GPIO_PI
N_12 | GPIO_PIN_13;
00376   HAL_GPIO_Init(GPIOD, &gpioinitstruct);
```

```
00377
00378   gpioinitstruct.Pin = GPIO_PIN_3 | GPIO_PIN
_4 | GPIO_PIN_5 | GPIO_PIN_6;
00379   HAL_GPIO_Init(GPIOE, &gpioinitstruct);
00380
00381   /*!< NOE and NWE configuration */
00382   gpioinitstruct.Pin = GPIO_PIN_4 | GPIO_PIN
_5;
00383   HAL_GPIO_Init(GPIOD, &gpioinitstruct);
00384
00385   /*!< NE2 configuration */
00386   gpioinitstruct.Pin = GPIO_PIN_9 | GPIO_PIN
_12;
00387   HAL_GPIO_Init(GPIOG, &gpioinitstruct);
00388
00389   /*!< Configure PD6 for NOR memory Ready/Bu
sy signal */
00390   gpioinitstruct.Pin = GPIO_PIN_6;
00391   gpioinitstruct.Mode = GPIO_MODE_INPUT;
00392   HAL_GPIO_Init(GPIOD, &gpioinitstruct);
00393
00394 }
00395
00396 /**
00397   * @brief  NOR BSP Wait for Ready/Busy sign
al.
00398   * @param  hnor: Pointer to NOR handle
00399   * @param  Timeout: Timeout duration
00400   * @retval None
00401   */
00402 void HAL_NOR_MspWait(NOR_HandleTypeDef *hnor
, uint32_t Timeout)
00403 {
00404   uint32_t timeout = Timeout;
00405
00406   /* Polling on Ready/Busy signal */
00407   while((HAL_GPIO_ReadPin(NOR_READY_BUSY_GPIO
```

```
, NOR_READY_BUSY_PIN) != NOR_BUSY_STATE) && (timeo
ut > 0))
00408   {
00409     timeout--;
00410   }
00411
00412   timeout = Timeout;
00413
00414   /* Polling on Ready/Busy signal */
00415   while((HAL_GPIO_ReadPin(NOR_READY_BUSY_GPIO
, NOR_READY_BUSY_PIN) != NOR_READY_STATE) && (time
out > 0))
00416   {
00417     timeout--;
00418   }
00419 }
00420
00421 /**
00422   * @}
00423   */
00424
00425 /**
00426   * @}
00427   */
00428
00429 /**
00430   * @}
00431   */
00432
00433 /**
00434   * @}
00435   */
00436
00437 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP
User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

stm3210e_eval_sd.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210e_eval_sd.c
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file includes the uSD card driver.
00008   ******************************************************************************
00009   * @attention
00010   *
00011   * <h2><center>&copy; COPYRIGHT(c) 2015 STMicroelectronics</center></h2>
00012   *
00013   * Redistribution and use in source and binary forms, with or without modification,
00014   * are permitted provided that the following conditions are met:
00015   *   1. Redistributions of source code must retain the above copyright notice,
```

```
00016   *      this list of conditions and the fol
lowing disclaimer.
00017   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00018   *      this list of conditions and the fol
lowing disclaimer in the documentation
00019   *      and/or other materials provided wit
h the distribution.
00020   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00021   *      may be used to endorse or promote p
roducts derived from this software
00022   *      without specific prior written perm
ission.
00023   *
00024   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00025   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00026   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00027   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00028   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00029   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 PROCUREMENT OF SUBSTITUTE GOODS OR
00030   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00031   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00032   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00033   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00034   *
00035   ****************************************
```

```
**********************************
00036   */
00037
00038 /* Define to prevent recursive inclusion ---
-------------------------------------*/
00039 #ifndef __STM3210E_EVAL_SD_H
00040 #define __STM3210E_EVAL_SD_H
00041
00042 #ifdef __cplusplus
00043  extern "C" {
00044 #endif
00045
00046 /* Includes ------------------------------
-------------------------------------*/
00047 #include "stm32f1xx_hal.h"
00048
00049 /** @addtogroup BSP
00050   * @{
00051   */
00052
00053 /** @addtogroup STM3210E_EVAL
00054   * @{
00055   */
00056
00057 /** @addtogroup STM3210E_EVAL_SD
00058   * @{
00059   */
00060
00061 /* Exported types ------------------------
-------------------------------------*/
00062
00063 /** @defgroup STM3210E_EVAL_SD_Exported_Type
s Exported_Types
00064   * @{
00065   */
00066
00067 /**
```

```
00068   * @brief SD Card information structure
00069   */
00070 #define SD_CardInfo HAL_SD_CardInfoTypedef
00071
00072 /**
00073   * @brief  SD status structure definition
00074   */
00075 #define MSD_OK          0x00
00076 #define MSD_ERROR       0x01
00077
00078 /**
00079   * @}
00080   */
00081
00082
00083 /* Exported constants ---------------------
---------------------------------*/
00084
00085 /** @defgroup STM3210E_EVAL_SD_Exported_Cons
tants Exported_Constants
00086   * @{
00087   */
00088 #define SD_DETECT_PIN                     GPI
O_PIN_11
00089 #define SD_DETECT_GPIO_PORT               GPI
OF
00090 #define __SD_DETECT_GPIO_CLK_ENABLE()    __H
AL_RCC_GPIOF_CLK_ENABLE()
00091 #define SD_DETECT_IRQn                    EXT
I15_10_IRQn
00092
00093 #define SD_DATATIMEOUT           ((uint32_t)
100000000)
00094
00095 #define SD_PRESENT               ((uint8_t)0
x01)
00096 #define SD_NOT_PRESENT           ((uint8_t)0
```

```
x00)
00097
00098 /* DMA definitions for SD DMA transfer */
00099 #define __DMAx_TxRx_CLK_ENABLE            __
HAL_RCC_DMA2_CLK_ENABLE
00100 #define SD_DMAx_Tx_INSTANCE               DM
A2_Channel4
00101 #define SD_DMAx_Rx_INSTANCE               DM
A2_Channel4
00102 #define SD_DMAx_Tx_IRQn                   DM
A2_Channel4_5_IRQn
00103 #define SD_DMAx_Rx_IRQn                   DM
A2_Channel4_5_IRQn
00104 #define SD_DMAx_Tx_IRQHandler             DM
A2_Channel4_5_IRQHandler
00105 #define SD_DMAx_Rx_IRQHandler             DM
A2_Channel4_5_IRQHandler
00106
00107 /**
00108   * @}
00109   */
00110
00111
00112 /* Exported functions ---------------------
-----------------------------------*/
00113
00114 /** @addtogroup STM3210E_EVAL_SD_Exported_Fu
nctions
00115   * @{
00116   */
00117 uint8_t BSP_SD_Init(void);
00118 uint8_t BSP_SD_ITConfig(void);
00119 void    BSP_SD_DetectIT(void);
00120 void    BSP_SD_DetectCallback(void);
00121 uint8_t BSP_SD_ReadBlocks(uint32_t *pData, u
int64_t ReadAddr, uint32_t BlockSize, uint32_t Num
OfBlocks);
```

```
00122 uint8_t BSP_SD_WriteBlocks(uint32_t *pData,
uint64_t WriteAddr, uint32_t BlockSize, uint32_t N
umOfBlocks);
00123 uint8_t BSP_SD_ReadBlocks_DMA(uint32_t *pDat
a, uint64_t ReadAddr, uint32_t BlockSize, uint32_t
 NumOfBlocks);
00124 uint8_t BSP_SD_WriteBlocks_DMA(uint32_t *pDa
ta, uint64_t WriteAddr, uint32_t BlockSize, uint32
_t NumOfBlocks);
00125 uint8_t BSP_SD_Erase(uint64_t StartAddr, uin
t64_t EndAddr);
00126 void    BSP_SD_IRQHandler(void);
00127 void    BSP_SD_DMA_Tx_IRQHandler(void);
00128 void    BSP_SD_DMA_Rx_IRQHandler(void);
00129 HAL_SD_TransferStateTypedef BSP_SD_GetStatus(
void);
00130 void    BSP_SD_GetCardInfo(HAL_SD_CardInfoTy
pedef *CardInfo);
00131 uint8_t BSP_SD_IsDetected(void);
00132
00133 /**
00134   * @}
00135   */
00136
00137 /**
00138   * @}
00139   */
00140
00141 /**
00142   * @}
00143   */
00144
00145 /**
00146   * @}
00147   */
00148
00149 #ifdef __cplusplus
```

```
00150 }
00151 #endif
00152
00153 #endif /* __STM3210E_EVAL_SD_H */
00154
00155 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210E_EVAL BSP User Manual

stm3210e_eval_sd.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210e_eval_sd.c
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file includes the uSD card driver.
00008   @verbatim
00009   ==============================================================================
00010                        ##### How to use this driver #####
00011   ==============================================================================
00012   (#) This driver is used to drive the micro SD external card mounted on STM3210E-EVAL
00013       evaluation board.
00014
00015   (#) This driver does not need a specific component driver for the micro SD device
```

```
00016      to be included with.
00017
00018   (#) Initialization steps:
00019         (++) Initialize the micro SD card usi
ng the BSP_SD_Init() function. This
00020              function includes the MSP layer
hardware resources initialization and the
00021              SDIO interface configuration to
interface with the external micro SD. It
00022              also includes the micro SD initi
alization sequence.
00023         (++) To check the SD card presence yo
u can use the function BSP_SD_IsDetected() which
00024              returns the detection status
00025         (++) If SD presence detection interru
pt mode is desired, you must configure the
00026              SD detection interrupt mode by c
alling the function BSP_SD_ITConfig(). The interru
pt
00027              is generated as an external inte
rrupt whenever the micro SD card is
00028              plugged/unplugged in/from the ev
aluation board. The SD detection interrupt
00029              is handeled by calling the funct
ion BSP_SD_DetectIT() which is called in the IRQ
00030              handler file, the user callback
is implemented in the function BSP_SD_DetectCallba
ck().
00031         (++) The function BSP_SD_GetCardInfo(
) is used to get the micro SD card information
00032              which is stored in the structure
 "HAL_SD_CardInfoTypedef".
00033
00034   (#) Micro SD card operations
00035         (++) The micro SD card can be accesse
d with read/write block(s) operations once
00036              it is reay for access. The acces
```

```
s cand be performed whether using the polling
00037              mode by calling the functions BS
P_SD_ReadBlocks()/BSP_SD_WriteBlocks(), or by DMA
00038              transfer using the functions BSP
_SD_ReadBlocks_DMA()/BSP_SD_WriteBlocks_DMA()
00039        (++) The DMA transfer complete is use
d with interrupt mode. Once the SD transfer
00040              is complete, the SD interrupt is
 handeled using the function BSP_SD_IRQHandler(),
00041              the DMA Tx/Rx transfer complete
are handeled using the functions
00042              BSP_SD_DMA_Tx_IRQHandler()/BSP_S
D_DMA_Rx_IRQHandler(). The corresponding user call
backs
00043              are implemented by the user at a
pplication level.
00044        (++) The SD erase block(s) is perform
ed using the function BSP_SD_Erase() with specifyi
ng
00045              the number of blocks to erase.
00046        (++) The SD runtime status is returne
d when calling the function BSP_SD_GetStatus().
00047    [..]
00048   @endverbatim
00049   ******************************************
**********************************
00050   * @attention
00051   *
00052   * <h2><center>&copy; COPYRIGHT(c) 2015 STM
icroelectronics</center></h2>
00053   *
00054   * Redistribution and use in source and bin
ary forms, with or without modification,
00055   * are permitted provided that the followin
g conditions are met:
00056   *   1. Redistributions of source code must
 retain the above copyright notice,
```

```
00057   *      this list of conditions and the fol
lowing disclaimer.
00058   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00059   *      this list of conditions and the fol
lowing disclaimer in the documentation
00060   *      and/or other materials provided wit
h the distribution.
00061   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00062   *      may be used to endorse or promote p
roducts derived from this software
00063   *      without specific prior written perm
ission.
00064   *
00065   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00066   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00067   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00068   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00069   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00070   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
00071   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00072   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00073   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00074   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00075   *
00076   ******************************************
```

```
**********************************
00077   */
00078
00079 /* Includes ------------------------------
-----------------------------------*/
00080 #include "stm3210e_eval_sd.h"
00081
00082 /** @addtogroup BSP
00083   * @{
00084   */
00085
00086 /** @addtogroup STM3210E_EVAL
00087   * @{
00088   */
00089
00090 /** @defgroup STM3210E_EVAL_SD STM3210E-EVAL
 SD
00091   * @{
00092   */
00093
00094 /** @defgroup STM3210E_SD_Private_Variables
Private Variables
00095   * @{
00096   */
00097 SD_HandleTypeDef uSdHandle;
00098 static SD_CardInfo uSdCardInfo;
00099 /**
00100   * @}
00101   */
00102
00103 /** @defgroup STM3210E_SD_Private_Functions
Private Functions
00104   * @{
00105   */
00106 static void SD_MspInit(void);
00107 HAL_SD_ErrorTypedef SD_DMAConfigRx(SD_Handle
TypeDef *hsd);
```

```
00108 HAL_SD_ErrorTypedef SD_DMAConfigTx(SD_Handle
TypeDef *hsd);
00109
00110 /**
00111   * @}
00112   */
00113
00114 /** @defgroup STM3210E_SD_Exported_Functions
 Exported_Functions
00115   * @{
00116   */
00117
00118 /**
00119   * @brief  Initializes the SD card device.
00120   * @retval SD status.
00121   */
00122 uint8_t BSP_SD_Init(void)
00123 {
00124   uint8_t state = MSD_OK;
00125
00126   /* uSD device interface configuration */
00127   uSdHandle.Instance = SDIO;
00128
00129   uSdHandle.Init.ClockEdge           = SDIO_
CLOCK_EDGE_RISING;
00130   uSdHandle.Init.ClockBypass         = SDIO_
CLOCK_BYPASS_DISABLE;
00131   uSdHandle.Init.ClockPowerSave      = SDIO_
CLOCK_POWER_SAVE_DISABLE;
00132   uSdHandle.Init.BusWide             = SDIO_
BUS_WIDE_1B;
00133   uSdHandle.Init.HardwareFlowControl = SDIO_
HARDWARE_FLOW_CONTROL_DISABLE;
00134   uSdHandle.Init.ClockDiv            = SDIO_
TRANSFER_CLK_DIV;
00135
00136   /* Check if the SD card is plugged in the
```

```
slot */
00137   if(BSP_SD_IsDetected() != SD_PRESENT)
00138   {
00139     return MSD_ERROR;
00140   }
00141
00142   /* HAL SD initialization */
00143   SD_MspInit();
00144   if(HAL_SD_Init(&uSdHandle, &uSdCardInfo) !
= SD_OK)
00145   {
00146     state = MSD_ERROR;
00147   }
00148
00149   /* Configure SD Bus width */
00150   if(state == MSD_OK)
00151   {
00152     /* Enable wide operation */
00153     if(HAL_SD_WideBusOperation_Config(&uSdHa
ndle, SDIO_BUS_WIDE_4B) != SD_OK)
00154     {
00155       state = MSD_ERROR;
00156     }
00157     else
00158     {
00159       state = MSD_OK;
00160     }
00161   }
00162
00163   return  state;
00164 }
00165
00166 /**
00167   * @brief  Configures Interrupt mode for SD
 detection pin.
00168   * @retval Returns 0
00169   */
```

```
00170 uint8_t BSP_SD_ITConfig(void)
00171 {
00172   GPIO_InitTypeDef gpioinitstruct = {0};
00173
00174   /* Configure Interrupt mode for SD detecti
on pin */
00175   gpioinitstruct.Mode      = GPIO_MODE_IT_RI
SING_FALLING;
00176   gpioinitstruct.Pull      = GPIO_PULLUP;
00177   gpioinitstruct.Speed     = GPIO_SPEED_FREQ
_HIGH;
00178   gpioinitstruct.Pin       = SD_DETECT_PIN;
00179   HAL_GPIO_Init(SD_DETECT_GPIO_PORT, &gpioin
itstruct);
00180
00181   /* NVIC configuration for SDIO interrupts
*/
00182   HAL_NVIC_SetPriority(SD_DETECT_IRQn, 5, 0)
;
00183   HAL_NVIC_EnableIRQ(SD_DETECT_IRQn);
00184
00185   return 0;
00186 }
00187
00188 /**
00189  * @brief  Detects if SD card is correctly p
lugged in the memory slot or not.
00190  * @retval Returns if SD is detected or not
00191  */
00192 uint8_t BSP_SD_IsDetected(void)
00193 {
00194   __IO uint8_t status = SD_PRESENT;
00195
00196   /* Check SD card detect pin */
00197   if(HAL_GPIO_ReadPin(SD_DETECT_GPIO_PORT, S
D_DETECT_PIN) != GPIO_PIN_RESET)
00198   {
```

```
00199     status = SD_NOT_PRESENT;
00200   }
00201
00202   return status;
00203 }
00204
00205 /** @brief  SD detect IT treatment
00206   * @retval None
00207   */
00208 void BSP_SD_DetectIT(void)
00209 {
00210   /* SD detect IT callback */
00211   BSP_SD_DetectCallback();
00212
00213 }
00214
00215
00216 /** @brief  SD detect IT detection callback
00217   * @retval None
00218   */
00219 __weak void BSP_SD_DetectCallback(void)
00220 {
00221   /* NOTE: This function Should not be modif
ied, when the callback is needed,
00222   the BSP_SD_DetectCallback could be impleme
nted in the user file
00223   */
00224
00225 }
00226
00227 /**
00228   * @brief  Reads block(s) from a specified
address in an SD card, in polling mode.
00229   * @param  pData: Pointer to the buffer tha
t will contain the data to transmit
00230   * @param  ReadAddr: Address from where dat
a is to be read
```

```
00231   * @param  BlockSize: SD card data block si
ze, that should be 512
00232   * @param  NumOfBlocks: Number of SD blocks
 to read
00233   * @retval SD status
00234   */
00235 uint8_t BSP_SD_ReadBlocks(uint32_t *pData, u
int64_t ReadAddr, uint32_t BlockSize, uint32_t Num
OfBlocks)
00236 {
00237   if(HAL_SD_ReadBlocks(&uSdHandle, pData, Re
adAddr, BlockSize, NumOfBlocks) != SD_OK)
00238   {
00239     return MSD_ERROR;
00240   }
00241   else
00242   {
00243     return MSD_OK;
00244   }
00245 }
00246
00247 /**
00248   * @brief  Writes block(s) to a specified a
ddress in an SD card, in polling mode.
00249   * @param  pData: Pointer to the buffer tha
t will contain the data to transmit
00250   * @param  WriteAddr: Address from where da
ta is to be written
00251   * @param  BlockSize: SD card data block si
ze, that should be 512
00252   * @param  NumOfBlocks: Number of SD blocks
 to write
00253   * @retval SD status
00254   */
00255 uint8_t BSP_SD_WriteBlocks(uint32_t *pData,
uint64_t WriteAddr, uint32_t BlockSize, uint32_t N
umOfBlocks)
```

```
00256 {
00257   if(HAL_SD_WriteBlocks(&uSdHandle, pData, W
riteAddr, BlockSize, NumOfBlocks) != SD_OK)
00258   {
00259     return MSD_ERROR;
00260   }
00261   else
00262   {
00263     return MSD_OK;
00264   }
00265 }
00266
00267 /**
00268   * @brief  Reads block(s) from a specified
address in an SD card, in DMA mode.
00269   * @param  pData: Pointer to the buffer tha
t will contain the data to transmit
00270   * @param  ReadAddr: Address from where dat
a is to be read
00271   * @param  BlockSize: SD card data block si
ze, that should be 512
00272   * @param  NumOfBlocks: Number of SD blocks
 to read
00273   * @retval SD status
00274   */
00275 uint8_t BSP_SD_ReadBlocks_DMA(uint32_t *pDat
a, uint64_t ReadAddr, uint32_t BlockSize, uint32_t
 NumOfBlocks)
00276 {
00277   uint8_t state = MSD_OK;
00278
00279   /* Invalidate the dma tx handle*/
00280   uSdHandle.hdmatx = NULL;
00281
00282   /* Prepare the dma channel for a read oper
ation */
00283   state = ((SD_DMAConfigRx(&uSdHandle) == SD
```

```
_OK) ? MSD_OK : MSD_ERROR);
00284
00285   if(state == MSD_OK)
00286   {
00287     /* Read block(s) in DMA transfer mode */
00288     state = ((HAL_SD_ReadBlocks_DMA(&uSdHand
le, pData, ReadAddr, BlockSize, NumOfBlocks) == SD
_OK) ? MSD_OK : MSD_ERROR);
00289
00290     /* Wait until transfer is complete */
00291     if(state == MSD_OK)
00292     {
00293       state = ((HAL_SD_CheckReadOperation(&u
SdHandle, (uint32_t)SD_DATATIMEOUT) == SD_OK) ? MS
D_OK : MSD_ERROR);
00294     }
00295   }
00296
00297   return state;
00298 }
00299
00300 /**
00301   * @brief  Writes block(s) to a specified a
ddress in an SD card, in DMA mode.
00302   * @param  pData: Pointer to the buffer tha
t will contain the data to transmit
00303   * @param  WriteAddr: Address from where da
ta is to be written
00304   * @param  BlockSize: SD card data block si
ze, that should be 512
00305   * @param  NumOfBlocks: Number of SD blocks
 to write
00306   * @retval SD status
00307   */
00308 uint8_t BSP_SD_WriteBlocks_DMA(uint32_t *pDa
ta, uint64_t WriteAddr, uint32_t BlockSize, uint32
_t NumOfBlocks)
```

```
00309 {
00310   uint8_t state = MSD_OK;
00311
00312   /* Invalidate the dma rx handle*/
00313   uSdHandle.hdmarx = NULL;
00314
00315   /* Prepare the dma channel for a read oper
ation */
00316   state = ((SD_DMAConfigTx(&uSdHandle) == SD
_OK) ? MSD_OK : MSD_ERROR);
00317
00318   if(state == MSD_OK)
00319   {
00320     /* Write block(s) in DMA transfer mode */

00321     state = ((HAL_SD_WriteBlocks_DMA(&uSdHan
dle, pData, WriteAddr, BlockSize, NumOfBlocks) ==
SD_OK) ? MSD_OK : MSD_ERROR);
00322
00323     /* Wait until transfer is complete */
00324     if(state == MSD_OK)
00325     {
00326       state = ((HAL_SD_CheckWriteOperation(&
uSdHandle, (uint32_t)SD_DATATIMEOUT) == SD_OK) ? M
SD_OK : MSD_ERROR);
00327     }
00328   }
00329
00330   return state;
00331 }
00332
00333 /**
00334   * @brief  Erases the specified memory area
 of the given SD card.
00335   * @param  StartAddr: Start byte address
00336   * @param  EndAddr: End byte address
00337   * @retval SD status
```

```
00338   */
00339 uint8_t BSP_SD_Erase(uint64_t StartAddr, uin
t64_t EndAddr)
00340 {
00341   if(HAL_SD_Erase(&uSdHandle, StartAddr, End
Addr) != SD_OK)
00342   {
00343     return MSD_ERROR;
00344   }
00345   else
00346   {
00347     return MSD_OK;
00348   }
00349 }
00350
00351 /**
00352   * @brief  Handles SD card interrupt reques
t.
00353   * @retval None
00354   */
00355 void BSP_SD_IRQHandler(void)
00356 {
00357   HAL_SD_IRQHandler(&uSdHandle);
00358 }
00359
00360 /**
00361   * @brief  Handles SD DMA Tx transfer inter
rupt request.
00362   * @retval None
00363   */
00364 void BSP_SD_DMA_Tx_IRQHandler(void)
00365 {
00366   HAL_DMA_IRQHandler(uSdHandle.hdmatx);
00367 }
00368
00369 /**
00370   * @brief  Handles SD DMA Rx transfer inter
```

```
rupt request.
00371   * @retval None
00372   */
00373 void BSP_SD_DMA_Rx_IRQHandler(void)
00374 {
00375   HAL_DMA_IRQHandler(uSdHandle.hdmarx);
00376 }
00377
00378 /**
00379   * @brief  Gets the current SD card data st
atus.
00380   * @retval Data transfer state.
00381   *          This value can be one of the fo
llowing values:
00382   *            @arg  SD_TRANSFER_OK: No data
 transfer is acting
00383   *            @arg  SD_TRANSFER_BUSY: Data
transfer is acting
00384   *            @arg  SD_TRANSFER_ERROR: Data
 transfer error
00385   */
00386 HAL_SD_TransferStateTypedef BSP_SD_GetStatus(
void)
00387 {
00388   return(HAL_SD_GetStatus(&uSdHandle));
00389 }
00390
00391 /**
00392   * @brief  Get SD information about specifi
c SD card.
00393   * @param  CardInfo: Pointer to HAL_SD_Card
InfoTypedef structure
00394   * @retval None
00395   */
00396 void BSP_SD_GetCardInfo(HAL_SD_CardInfoTyped
ef *CardInfo)
00397 {
```

```
00398   /* Get SD card Information */
00399   HAL_SD_Get_CardInfo(&uSdHandle, CardInfo);
00400 }
00401
00402 /**
00403   * @}
00404   */
00405
00406
00407 /** @addtogroup STM3210E_SD_Private_Functions

00408   * @{
00409   */
00410
00411 /**
00412   * @brief  Initializes the SD MSP.
00413   * @retval None
00414   */
00415 static void SD_MspInit(void)
00416 {
00417   GPIO_InitTypeDef gpioinitstruct = {0};
00418
00419   /* Enable SDIO clock */
00420   __HAL_RCC_SDIO_CLK_ENABLE();
00421
00422   /* Enable DMA2 clocks */
00423   __DMAx_TxRx_CLK_ENABLE();
00424
00425   /* Enable GPIOs clock */
00426   __HAL_RCC_GPIOC_CLK_ENABLE();
00427   __HAL_RCC_GPIOD_CLK_ENABLE();
00428   __SD_DETECT_GPIO_CLK_ENABLE();
00429
00430   /* Common GPIO configuration */
00431   gpioinitstruct.Mode      = GPIO_MODE_AF_PP
;
00432   gpioinitstruct.Pull      = GPIO_PULLUP;
```

```
00433   gpioinitstruct.Speed     = GPIO_SPEED_FREQ
_HIGH;
00434
00435   /* GPIOC configuration */
00436   gpioinitstruct.Pin = GPIO_PIN_8 | GPIO_PIN
_9 | GPIO_PIN_10 | GPIO_PIN_11 | GPIO_PIN_12;
00437
00438   HAL_GPIO_Init(GPIOC, &gpioinitstruct);
00439
00440   /* GPIOD configuration */
00441   gpioinitstruct.Pin = GPIO_PIN_2;
00442   HAL_GPIO_Init(GPIOD, &gpioinitstruct);
00443
00444   /* SD Card detect pin configuration */
00445   gpioinitstruct.Mode      = GPIO_MODE_INPUT
;
00446   gpioinitstruct.Pull      = GPIO_PULLUP;
00447   gpioinitstruct.Speed     = GPIO_SPEED_FREQ
_HIGH;
00448   gpioinitstruct.Pin       = SD_DETECT_PIN;
00449   HAL_GPIO_Init(SD_DETECT_GPIO_PORT, &gpioin
itstruct);
00450
00451   /* NVIC configuration for SDIO interrupts
*/
00452   HAL_NVIC_SetPriority(SDIO_IRQn, 0, 0);
00453   HAL_NVIC_EnableIRQ(SDIO_IRQn);
00454
00455   /* DMA initialization should be done here
but , as there is only one channel for RX and TX i
t is configured and done directly when required*/
00456 }
00457
00458 /**
00459   * @brief SD_DMAConfigRx
00460   * @par Function Description
00461   *   This function configure the DMA to rec
```

```
eive data from the SD card
00462   * @retval
00463   *  SD_ERROR or SD_OK
00464   */
00465 HAL_SD_ErrorTypedef SD_DMAConfigRx(SD_Handle
TypeDef *hsd)
00466 {
00467   static DMA_HandleTypeDef hdma_rx;
00468   HAL_StatusTypeDef status = HAL_ERROR;
00469
00470   if(hsd->hdmarx == NULL)
00471   {
00472     /* Configure DMA Rx parameters */
00473     hdma_rx.Init.Direction           = DMA_P
ERIPH_TO_MEMORY;
00474     hdma_rx.Init.PeriphInc           = DMA_P
INC_DISABLE;
00475     hdma_rx.Init.MemInc              = DMA_M
INC_ENABLE;
00476     hdma_rx.Init.PeriphDataAlignment = DMA_P
DATAALIGN_WORD;
00477     hdma_rx.Init.MemDataAlignment    = DMA_M
DATAALIGN_WORD;
00478     hdma_rx.Init.Priority            = DMA_P
RIORITY_VERY_HIGH;
00479
00480     hdma_rx.Instance = SD_DMAx_Rx_INSTANCE;
00481
00482     /* Associate the DMA handle */
00483     __HAL_LINKDMA(hsd, hdmarx, hdma_rx);
00484
00485     /* Stop any ongoing transfer and reset t
he state*/
00486     HAL_DMA_Abort(&hdma_rx);
00487
00488     /* Deinitialize the Channel for new tran
sfer */
```

```
00489     HAL_DMA_DeInit(&hdma_rx);
00490
00491     /* Configure the DMA Channel */
00492     status = HAL_DMA_Init(&hdma_rx);
00493
00494     /* NVIC configuration for DMA transfer c
omplete interrupt */
00495     HAL_NVIC_SetPriority(SD_DMAx_Rx_IRQn, 1,
 0);
00496     HAL_NVIC_EnableIRQ(SD_DMAx_Rx_IRQn);
00497   }
00498   else
00499   {
00500     status = HAL_OK;
00501   }
00502
00503   return (status != HAL_OK? SD_ERROR : SD_OK
);
00504 }
00505
00506 /**
00507   * @brief SD_DMAConfigTx
00508   * @par Function Description
00509   *   This function configure the DMA to tra
nsmit data to the SD card
00510   * @retval
00511   *  SD_ERROR or SD_OK
00512   */
00513 HAL_SD_ErrorTypedef SD_DMAConfigTx(SD_Handle
TypeDef *hsd)
00514 {
00515   static DMA_HandleTypeDef hdma_tx;
00516   HAL_StatusTypeDef status;
00517
00518   if(hsd->hdmatx == NULL)
00519   {
00520     /* Configure DMA Tx parameters */
```

```
00521     hdma_tx.Init.Direction           = DMA_M
EMORY_TO_PERIPH;
00522     hdma_tx.Init.PeriphInc           = DMA_P
INC_DISABLE;
00523     hdma_tx.Init.MemInc              = DMA_M
INC_ENABLE;
00524     hdma_tx.Init.PeriphDataAlignment = DMA_P
DATAALIGN_WORD;
00525     hdma_tx.Init.MemDataAlignment    = DMA_M
DATAALIGN_WORD;
00526     hdma_tx.Init.Priority            = DMA_P
RIORITY_VERY_HIGH;
00527
00528     hdma_tx.Instance = SD_DMAx_Tx_INSTANCE;
00529
00530     /* Associate the DMA handle */
00531     __HAL_LINKDMA(hsd, hdmatx, hdma_tx);
00532
00533     /* Stop any ongoing transfer and reset t
he state*/
00534     HAL_DMA_Abort(&hdma_tx);
00535
00536     /* Deinitialize the Channel for new tran
sfer */
00537     HAL_DMA_DeInit(&hdma_tx);
00538
00539     /* Configure the DMA Channel */
00540     status = HAL_DMA_Init(&hdma_tx);
00541
00542     /* NVIC configuration for DMA transfer c
omplete interrupt */
00543     HAL_NVIC_SetPriority(SD_DMAx_Tx_IRQn, 1,
 0);
00544     HAL_NVIC_EnableIRQ(SD_DMAx_Tx_IRQn);
00545   }
00546   else
00547   {
```

```
00548     status = HAL_OK;
00549   }
00550 
00551   return (status != HAL_OK? SD_ERROR : SD_OK
);
00552 }
00553 
00554 /**
00555   * @}
00556   */
00557 
00558 /**
00559   * @}
00560   */
00561 
00562 /**
00563   * @}
00564   */
00565 
00566 /**
00567   * @}
00568   */
00569 
00570 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210E_EVAL BSP User Manual

stm3210e_eval_serialflash.h

Go to the documentation of this file.

```
00001 /**
00002   ********************************************************************************
00003   * @file    stm3210e_eval_serialflash.h
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm3210e_eval_flash driver.
00009   ********************************************************************************
00010   * @attention
00011   *
00012   * <h2><center>&copy; COPYRIGHT(c) 2015 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 __STM3210E_EVAL_SERIAL_FLASH_H
00041 #define __STM3210E_EVAL_SERIAL_FLASH_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------------------------------------------*/
00048 #include "stm3210e_eval.h"
00049
00050 /** @addtogroup BSP
00051   * @{
00052   */
00053
00054 /** @addtogroup STM3210E_EVAL
00055   * @{
00056   */
00057
00058 /** @addtogroup STM3210E_EVAL_SERIAL_FLASH
00059   * @{
00060   */
00061
00062 /** @defgroup STM3210E_EVAL_SERIAL_FLASH_Exported_Types Exported_Types
00063   * @{
00064   */
00065 /**
00066   * @}
00067   */
00068
```

```
00069 /** @defgroup STM3210E_EVAL_SERIAL_FLASH_Exp
orted_Constants Exported_Constants
00070   * @{
00071   */
00072 #define   FLASH_OK         0x00
00073 #define   FLASH_ERROR      0x01
00074 /**
00075   * @}
00076   */
00077
00078 /** @defgroup STM3210E_EVAL_SERIAL_FLASH_Exp
orted_Macros Exported_Macros
00079   * @{
00080   */
00081 /**
00082   * @}
00083   */
00084
00085
00086 /** @addtogroup STM3210E_EVAL_SERIAL_FLASH_E
xported_Functions
00087   * @{
00088   */
00089 uint8_t  BSP_SERIAL_FLASH_Init(void);
00090 uint8_t  BSP_SERIAL_FLASH_EraseSector(uint32
_t SectorAddr);
00091 uint8_t  BSP_SERIAL_FLASH_EraseBulk(void);
00092 uint8_t  BSP_SERIAL_FLASH_WritePage(uint32_t
 uwStartAddress, uint8_t* pData, uint32_t uwDataSi
ze);
00093 uint8_t  BSP_SERIAL_FLASH_WriteData(uint32_t
 uwStartAddress, uint8_t* pData, uint32_t uwDataSi
ze);
00094 uint8_t  BSP_SERIAL_FLASH_ReadData( uint32_t
 uwStartAddress, uint8_t* pData, uint32_t uwDataSi
ze);
00095 uint32_t BSP_SERIAL_FLASH_ReadID(void);
```

```
00096
00097 /* Link function for EEPROM peripheral over
SPI */
00098 HAL_StatusTypeDef         FLASH_SPI_IO_Init(
void);
00099 uint8_t                   FLASH_SPI_IO_Write
Byte(uint8_t Data);
00100 uint8_t                   FLASH_SPI_IO_ReadB
yte(void);
00101 HAL_StatusTypeDef         FLASH_SPI_IO_ReadD
ata(uint32_t MemAddress, uint8_t* pBuffer, uint32_
t BufferSize);
00102 void                      FLASH_SPI_IO_Write
Enable(void);
00103 HAL_StatusTypeDef         FLASH_SPI_IO_WaitF
orWriteEnd(void);
00104 uint32_t                  FLASH_SPI_IO_ReadID
(void);
00105 /**
00106   * @}
00107   */
00108
00109
00110 #ifdef __cplusplus
00111 }
00112 #endif
00113
00114 #endif /* __STM3210E_EVAL_SERIAL_FLASH_H */
00115 /**
00116   * @}
00117   */
00118
00119 /**
00120   * @}
00121   */
00122
00123 /**
```

```
00124   * @}
00125   */
00126
00127 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210E_EVAL BSP User Manual

stm3210e_eval_serialflash.c

Go to the documentation of this file.

```
00001 /**
00002   ********************************************************************************
00003   * @file    stm3210e_eval_serialflash.c
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file provides a set of functions needed to manage the SPI M25Pxx
00008   *          FLASH memory mounted on STM3210E_EVAL board.
00009   *          It implements a high level communication layer for read and write
00010   *          from/to this memory. The needed STM32 hardware resources (SPI and
00011   *          GPIO) are defined in stm3210e_eval.h file, and the initialization is
00012   *          performed in FLASH_SPI_IO_Init() function declared in stm3210e_eval.c
00013   *          file.
00014   *          You can easily tailor this driver to any other development board,
```

```
00015   *          by just adapting the defines fo
r hardware resources and
00016   *          FLASH_SPI_IO_Init() function.
00017   *
00018   *   +-----------------------------------
--------------------------------+
00019   *   |                          Pin assignmen
t                              |
00020   *   +-----------------------------------
--+---------------+-------------+
00021   *   |  STM32 SPI Pins
  |   BSP_FLASH   |    Pin      |
00022   *   +-----------------------------------
--+---------------+-------------+
00023   *   | BSP_SERIAL_FLASH_CS_PIN
  | ChipSelect(/S)|    1        |
00024   *   | BSP_SERIAL_FLASH_SPI_MISO_PIN / MISO
  |   DataOut(Q)  |    2        |
00025   *   |
  |   VCC         |    3 (3.3 V)|
00026   *   |
  |   GND         |    4 (0 V)  |
00027   *   | BSP_SERIAL_FLASH_SPI_MOSI_PIN / MOSI
  |   DataIn(D)   |    5        |
00028   *   | BSP_SERIAL_FLASH_SPI_SCK_PIN / SCLK
  |   Clock(C)    |    6        |
00029   *   |
  |    VCC        |    7 (3.3 V)|
00030   *   |
  |    VCC        |    8 (3.3 V)|
00031   *   +-----------------------------------
--+---------------+-------------+
00032   ******************************************
**********************************
00033   * @attention
00034   *
00035   * <h2><center>&copy; COPYRIGHT(c) 2015 STM
```

```
icroelectronics</center></h2>
00036   *
00037   * Redistribution and use in source and bin
ary forms, with or without modification,
00038   * are permitted provided that the followin
g conditions are met:
00039   *   1. Redistributions of source code must
 retain the above copyright notice,
00040   *      this list of conditions and the fol
lowing disclaimer.
00041   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00042   *      this list of conditions and the fol
lowing disclaimer in the documentation
00043   *      and/or other materials provided wit
h the distribution.
00044   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00045   *      may be used to endorse or promote p
roducts derived from this software
00046   *      without specific prior written perm
ission.
00047   *
00048   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00049   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00050   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00051   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00052   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00053   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
00054   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
```

```
00055   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00056   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00057   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00058   *
00059   ******************************************
**********************************
00060   */
00061
00062 /* Includes ------------------------------
-----------------------------------*/
00063 #include "stm3210e_eval_serialflash.h"
00064
00065 /** @addtogroup BSP
00066   * @{
00067   */
00068
00069 /** @addtogroup STM3210E_EVAL
00070   * @{
00071   */
00072
00073 /** @defgroup STM3210E_EVAL_SERIAL_FLASH STM
3210E_EVAL Serial FLASH
00074   * @brief    This file includes the M25Pxxx
 SPI FLASH driver of
00075   *           STM3210E-EVAL board.
00076   * @{
00077   */
00078
00079
00080
00081 /** @defgroup STM3210E_EVAL_SERIAL_FLASH_Exp
orted_Functions Exported_Functions
00082   * @{
00083   */
```

```
00084
00085 /**
00086   * @brief  Initializes peripherals used by
the Serial FLASH device.
00087   * @retval FLASH_OK (0x00) if operation is
correctly performed, else
00088   *         return FLASH_ERROR (0x01).
00089   */
00090 uint8_t BSP_SERIAL_FLASH_Init(void)
00091 {
00092   if(FLASH_SPI_IO_Init() != HAL_OK)
00093   {
00094     return FLASH_ERROR;
00095   }
00096   else
00097   {
00098     return FLASH_OK;
00099   }
00100 }
00101
00102 /**
00103   * @brief  Erases the specified FLASH secto
r.
00104   * @param  SectorAddr: address of the secto
r to erase.
00105   * @retval FLASH_OK (0x00) if operation is
correctly performed, else
00106   *         return FLASH_ERROR (0x01).
00107   */
00108 uint8_t BSP_SERIAL_FLASH_EraseSector(uint32_
t SectorAddr)
00109 {
00110   /*!< Sector Erase */
00111   /*!< Select the FLASH  and send "Write Ena
ble" instruction */
00112   FLASH_SPI_IO_WriteEnable();
00113
```

```
00114   /*!< Send Sector Erase instruction */
00115   FLASH_SPI_IO_WriteByte(FLASH_SPI_CMD_SE);
00116   /*!< Send SectorAddr high nibble address b
yte */
00117   FLASH_SPI_IO_WriteByte((SectorAddr & 0xFF0
000) >> 16);
00118   /*!< Send SectorAddr medium nibble address
 byte */
00119   FLASH_SPI_IO_WriteByte((SectorAddr & 0xFF0
0) >> 8);
00120   /*!< Send SectorAddr low nibble address by
te */
00121   FLASH_SPI_IO_WriteByte(SectorAddr & 0xFF);
00122
00123   /*!< Wait the end of Flash writing and Des
elect the FLASH*/
00124   if(FLASH_SPI_IO_WaitForWriteEnd()!= HAL_OK
)
00125   {
00126     return FLASH_ERROR;
00127   }
00128   else
00129   {
00130     return FLASH_OK;
00131   }
00132 }
00133
00134 /**
00135   * @brief  Erases the entire FLASH.
00136   * @retval FLASH_OK (0x00) if operation is
correctly performed, else
00137   *         return FLASH_ERROR (0x01).
00138   */
00139 uint8_t BSP_SERIAL_FLASH_EraseBulk(void)
00140 {
00141   /*!< Bulk Erase */
00142   /*!< Select the FLASH  and send "Write Ena
```

```
ble" instruction */
00143   FLASH_SPI_IO_WriteEnable();
00144
00145   /*!< Send Bulk Erase instruction  */
00146   FLASH_SPI_IO_WriteByte(FLASH_SPI_CMD_BE);
00147
00148   /*!< Wait the end of Flash writing and Des
elect the FLASH*/
00149   if(FLASH_SPI_IO_WaitForWriteEnd()!= HAL_OK
)
00150   {
00151     return FLASH_ERROR;
00152   }
00153   else
00154   {
00155     return FLASH_OK;
00156   }
00157 }
00158
00159 /**
00160   * @brief  Writes more than one byte to the
 FLASH with a single WRITE cycle
00161   *         (Page WRITE sequence).
00162   * @note   The number of byte can't exceed
the FLASH page size (FLASH_SPI_PAGESIZE).
00163   * @param  pData: pointer to the buffer  co
ntaining the data to be written
00164   *         to the FLASH.
00165   * @param  uwStartAddress: FLASH's internal
 address to write to.
00166   * @param  uwDataSize: number of bytes to w
rite to the FLASH, must be equal
00167   *         or less than "FLASH_SPI_PAGESIZE
" value.
00168   * @retval FLASH_OK (0x00) if operation is
correctly performed, else
00169   *         return FLASH_ERROR (0x01).
```

```
00170   */
00171 uint8_t BSP_SERIAL_FLASH_WritePage(uint32_t
uwStartAddress, uint8_t* pData, uint32_t uwDataSiz
e)
00172 {
00173   /*!< Select the FLASH  and send "Write Ena
ble" instruction */
00174   FLASH_SPI_IO_WriteEnable();
00175
00176   /*!< Send "Write to Memory " instruction */

00177   FLASH_SPI_IO_WriteByte(FLASH_SPI_CMD_WRITE
);
00178   /*!< Send uwStartAddress high nibble addre
ss byte to write to */
00179   FLASH_SPI_IO_WriteByte((uwStartAddress & 0
xFF0000) >> 16);
00180   /*!< Send uwStartAddress medium nibble add
ress byte to write to */
00181   FLASH_SPI_IO_WriteByte((uwStartAddress & 0
xFF00) >> 8);
00182   /*!< Send uwStartAddress low nibble addres
s byte to write to */
00183   FLASH_SPI_IO_WriteByte(uwStartAddress & 0x
FF);
00184
00185   /*!< while there is data to be written on
the FLASH */
00186   while (uwDataSize--)
00187   {
00188     /*!< Send the current byte */
00189     FLASH_SPI_IO_WriteByte(*pData);
00190     /*!< Point on the next byte to be writte
n */
00191     pData++;
00192   }
00193
```

```
00194   /*!< Wait the end of Flash writing */
00195   if (FLASH_SPI_IO_WaitForWriteEnd()!= HAL_O
K)
00196   {
00197     return FLASH_ERROR;
00198   }
00199   else
00200   {
00201     return FLASH_OK;
00202   }
00203 }
00204
00205 /**
00206   * @brief  Writes block of data to the FLAS
H. In this function, the number of
00207   *         WRITE cycles are reduced, using
Page WRITE sequence.
00208   * @param  pData: pointer to the buffer  co
ntaining the data to be written
00209   *         to the FLASH.
00210   * @param  uwStartAddress: FLASH's internal
 address to write to.
00211   * @param  uwDataSize: number of bytes to w
rite to the FLASH.
00212   * @retval FLASH_OK (0x00) if operation is
correctly performed, else
00213   *         return FLASH_ERROR (0x01).
00214   */
00215 uint8_t BSP_SERIAL_FLASH_WriteData(uint32_t
uwStartAddress, uint8_t* pData, uint32_t uwDataSiz
e)
00216 {
00217   uint8_t NumOfPage = 0, NumOfSingle = 0, Ad
dr = 0, count = 0, temp = 0, status = 0;
00218
00219   Addr = uwStartAddress % FLASH_SPI_PAGESIZE
;
```

```
00220   count = FLASH_SPI_PAGESIZE - Addr;
00221   NumOfPage =  uwDataSize / FLASH_SPI_PAGESI
ZE;
00222   NumOfSingle = uwDataSize % FLASH_SPI_PAGES
IZE;
00223
00224   if (Addr == 0) /*!< uwStartAddress is FLAS
H_SPI_PAGESIZE aligned  */
00225   {
00226     if (NumOfPage == 0) /*!< uwDataSize < FL
ASH_SPI_PAGESIZE */
00227     {
00228       status = BSP_SERIAL_FLASH_WritePage(uw
StartAddress, pData, uwDataSize);
00229     }
00230     else /*!< uwDataSize > FLASH_SPI_PAGESIZ
E */
00231     {
00232       while (NumOfPage--)
00233       {
00234         status = BSP_SERIAL_FLASH_WritePage(
uwStartAddress, pData, FLASH_SPI_PAGESIZE);
00235         uwStartAddress +=  FLASH_SPI_PAGESIZE
;
00236         pData += FLASH_SPI_PAGESIZE;
00237       }
00238
00239       status = BSP_SERIAL_FLASH_WritePage(uw
StartAddress, pData, NumOfSingle);
00240     }
00241   }
00242   else /*!< uwStartAddress is not FLASH_SPI_
PAGESIZE aligned  */
00243   {
00244     if (NumOfPage == 0) /*!< uwDataSize < FL
ASH_SPI_PAGESIZE */
00245     {
```

```
00246       if (NumOfSingle > count) /*!< (uwDataS
ize + uwStartAddress) > FLASH_SPI_PAGESIZE */
00247       {
00248         temp = NumOfSingle - count;
00249 
00250         status = BSP_SERIAL_FLASH_WritePage(
uwStartAddress, pData, count);
00251         uwStartAddress +=  count;
00252         pData += count;
00253 
00254         status = BSP_SERIAL_FLASH_WritePage(
uwStartAddress, pData, temp);
00255       }
00256       else
00257       {
00258         status = BSP_SERIAL_FLASH_WritePage(
uwStartAddress, pData, uwDataSize);
00259       }
00260     }
00261     else /*!< uwDataSize > BSP_SERIAL_FLASH_
PAGESIZE */
00262     {
00263       uwDataSize -= count;
00264       NumOfPage =  uwDataSize / FLASH_SPI_PA
GESIZE;
00265       NumOfSingle = uwDataSize % FLASH_SPI_P
AGESIZE;
00266 
00267       status = BSP_SERIAL_FLASH_WritePage(uw
StartAddress, pData, count);
00268       uwStartAddress +=  count;
00269       pData += count;
00270 
00271       while (NumOfPage--)
00272       {
00273         status = BSP_SERIAL_FLASH_WritePage(
uwStartAddress, pData, FLASH_SPI_PAGESIZE);
```

```
00274         uwStartAddress +=  FLASH_SPI_PAGESIZE
;
00275         pData += FLASH_SPI_PAGESIZE;
00276       }
00277
00278       if (NumOfSingle != 0)
00279       {
00280         status = BSP_SERIAL_FLASH_WritePage(
uwStartAddress, pData, NumOfSingle);
00281       }
00282     }
00283   }
00284   if (status!= HAL_OK)
00285   {
00286     return FLASH_ERROR;
00287   }
00288   else
00289   {
00290     return FLASH_OK;
00291   }
00292 }
00293
00294 /**
00295   * @brief  Reads a block of data from the F
LASH.
00296   * @param  pData: pointer to the buffer tha
t receives the data read from the FLASH.
00297   * @param  uwStartAddress: FLASH's internal
 address to read from.
00298   * @param  uwDataSize: number of bytes to r
ead from the FLASH.
00299   * @retval FLASH_OK (0x00) if operation is
correctly performed, else
00300   *         return FLASH_ERROR (0x01).
00301   */
00302 uint8_t BSP_SERIAL_FLASH_ReadData(uint32_t u
wStartAddress, uint8_t* pData, uint32_t uwDataSize
```

```
)
00303 {
00304   if(FLASH_SPI_IO_ReadData(uwStartAddress, p
Data, uwDataSize)!= HAL_OK)
00305   {
00306     return FLASH_ERROR;
00307   }
00308   else
00309   {
00310     return FLASH_OK;
00311   }
00312 }
00313
00314
00315 /**
00316   * @brief  Reads FLASH identification.
00317   * @retval FLASH identification
00318   */
00319 uint32_t BSP_SERIAL_FLASH_ReadID(void)
00320 {
00321   return(FLASH_SPI_IO_ReadID());
00322 }
00323
00324
00325 /**
00326   * @}
00327   */
00328
00329 /**
00330   * @}
00331   */
00332
00333 /**
00334   * @}
00335   */
00336
00337 /**
```

```
00338   * @}
00339   */
00340
00341 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210E_EVAL BSP User Manual

stm3210e_eval_sram.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210e_eval_sram.h
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm3210e_eval_sram.c driver.
00009   ******************************************************************************
00010   * @attention
00011   *
00012   * <h2><center>&copy; COPYRIGHT(c) 2015 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 __STM3210E_EVAL_SRAM_H
00041 #define __STM3210e_EVAL_SRAM_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------------------------------------------*/
00048 #include "stm32f1xx_hal.h"
00049
00050 /** @addtogroup BSP
00051   * @{
00052   */
00053
00054 /** @addtogroup STM3210E_EVAL
00055   * @{
00056   */
00057
00058 /** @addtogroup STM3210E_EVAL_SRAM
00059   * @{
00060   */
00061
00062 /* Exported types ------------------------------------------------------------*/
00063
00064 /** @defgroup STM3210E_EVAL_SRAM_Exported_Types Exported_Types
00065   * @{
00066   */
```

```
00067
00068 /**
00069   * @}
00070   */
00071
00072
00073 /* Exported constants ---------------------
---------------------------------*/
00074 /** @defgroup STM3210E_EVAL_SRAM_Exported_Co
nstants Exported_Constants
00075   * @{
00076   */
00077
00078 /**
00079   * @brief  SRAM status structure definition

00080   */
00081 #define   SRAM_OK         0x00
00082 #define   SRAM_ERROR      0x01
00083
00084 #define SRAM_DEVICE_ADDR  ((uint32_t)0x68000
000)
00085 #define SRAM_DEVICE_SIZE  ((uint32_t)0x20000
0)  /* SRAM device size in MBytes */
00086
00087 /* #define SRAM_MEMORY_WIDTH    FSMC_NORSRAM
_MEM_BUS_WIDTH_8  */
00088 #define SRAM_MEMORY_WIDTH    FSMC_NORSRAM_ME
M_BUS_WIDTH_16
00089
00090 #define SRAM_BURSTACCESS    FSMC_BURST_ACCES
S_MODE_DISABLE
00091 /* #define SRAM_BURSTACCESS    FSMC_BURST_AC
CESS_MODE_ENABLE*/
00092
00093 #define SRAM_WRITEBURST    FSMC_WRITE_BURST_
DISABLE
```

```
00094 /* #define SRAM_WRITEBURST   FSMC_WRITE_BURS
T_ENABLE */
00095
00096 /* DMA definitions for SRAM DMA transfer */
00097 #define __SRAM_DMAx_CLK_ENABLE             __
HAL_RCC_DMA2_CLK_ENABLE
00098 #define SRAM_DMAx_STREAM                   DM
A2_Channel1
00099 #define SRAM_DMAx_IRQn                     DM
A2_Channel1_IRQn
00100 #define SRAM_DMAx_IRQHandler               DM
A2_Channel1_IRQHandler
00101
00102 /**
00103   * @}
00104   */
00105
00106 /* Exported macro ----------------------------
-----------------------------------*/
00107
00108 /** @defgroup STM3210E_EVAL_SRAM_Exported_Ma
cro Exported_Macro
00109   * @{
00110   */
00111
00112 /**
00113   * @}
00114   */
00115
00116 /* Exported functions ----------------------
-----------------------------------*/
00117
00118 /** @addtogroup STM3210E_EVAL_SRAM_Exported_
Functions
00119   * @{
00120   */
00121 uint8_t BSP_SRAM_Init(void);
```

```
00122 uint8_t BSP_SRAM_ReadData(uint32_t uwStartAd
dress, uint16_t *pData, uint32_t uwDataSize);
00123 uint8_t BSP_SRAM_ReadData_DMA(uint32_t uwSta
rtAddress, uint16_t *pData, uint32_t uwDataSize);
00124 uint8_t BSP_SRAM_WriteData(uint32_t uwStartA
ddress, uint16_t *pData, uint32_t uwDataSize);
00125 uint8_t BSP_SRAM_WriteData_DMA(uint32_t uwSt
artAddress, uint16_t *pData, uint32_t uwDataSize);
00126 void    BSP_SRAM_DMA_IRQHandler(void);
00127
00128 /**
00129   * @}
00130   */
00131
00132 /**
00133   * @}
00134   */
00135
00136 /**
00137   * @}
00138   */
00139
00140 /**
00141   * @}
00142   */
00143
00144 #ifdef __cplusplus
00145 }
00146 #endif
00147
00148 #endif /* __STM3210E_EVAL_SRAM_H */
00149
00150 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
00151
```

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP
User Manual by  1.7.6.1

STM3210E_EVAL BSP User Manual

stm3210e_eval_sram.c

Go to the documentation of this file.

```
00001 /**
00002   ********************************************************************************
00003   * @file    stm3210e_eval_sram.c
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file includes the SRAM driver for the IS61WV51216BLL-10M memory
00008   *          device mounted on STM3210E-EVAL evaluation board.
00009   ********************************************************************************
00010   * @attention
00011   *
00012   * <h2><center>&copy; COPYRIGHT(c) 2015 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 /* File Info : -----------------------------------------------------------------
00040                                       User NOTES

00041 1. How To use this driver:
00042 --------------------------
00043    - This driver is used to drive the IS61WV51216BLL-10M SRAM external memory mounted
00044      on STM3210E-EVAL evaluation board.
00045    - This driver does not need a specific component driver for the SRAM device
00046      to be included with.
00047
00048 2. Driver description:
00049 ---------------------
00050   + Initialization steps:
00051      o Initialize the SRAM external memory using the BSP_SRAM_Init() function. This
00052        function includes the MSP layer hardware resources initialization and the
00053        FSMC controller configuration to interface with the external SRAM memory.
00054
00055   + SRAM read/write operations
00056      o SRAM external memory can be accessed with read/write operations once it is
00057        initialized.
00058        Read/write operation can be performed with AHB access using the functions
00059        BSP_SRAM_ReadData()/BSP_SRAM_WriteData(), or by DMA transfer using the functions
00060        BSP_SRAM_ReadData_DMA()/BSP_SRAM_Writ
```

```
eData_DMA().
00061      o The AHB access is performed with 16-b
it width transaction, the DMA transfer
00062        configuration is fixed at single (no
burst) halfword transfer
00063        (see the SRAM_MspInit() static functi
on).
00064      o User can implement his own functions
for read/write access with his desired
00065        configurations.
00066      o If interrupt mode is used for DMA tra
nsfer, the function BSP_SRAM_DMA_IRQHandler()
00067        is called in IRQ handler file, to ser
ve the generated interrupt once the DMA
00068        transfer is complete.
00069
00070 ------------------------------------------
-----------------------------------*/
00071
00072 /* Includes ------------------------------
-----------------------------------*/
00073 #include "stm3210e_eval_sram.h"
00074
00075 /** @addtogroup BSP
00076   * @{
00077   */
00078
00079 /** @addtogroup STM3210E_EVAL
00080   * @{
00081   */
00082
00083 /** @defgroup STM3210E_EVAL_SRAM STM3210E_EV
AL SRAM
00084   * @{
00085   */
00086
00087 /* Private typedef -----------------------
```

```
------------------------------------*/
00088
00089 /** @defgroup STM3210E_EVAL_SRAM_Private_Typ
es_Definitions Private_Types_Definitions
00090   * @{
00091   */
00092
00093 /**
00094   * @}
00095   */
00096
00097 /* Private define ---------------------------
-----------------------------------*/
00098
00099 /** @defgroup STM3210E_EVAL_SRAM_Private_Def
ines Private_Defines
00100   * @{
00101   */
00102
00103 /**
00104   * @}
00105   */
00106
00107 /* Private macro ----------------------------
-----------------------------------*/
00108
00109 /** @defgroup STM3210E_EVAL_SRAM_Private_Mac
ros Private_Macros
00110   * @{
00111   */
00112
00113 /**
00114   * @}
00115   */
00116
00117 /* Private variables ------------------------
-----------------------------------*/
```

```
00118
00119 /** @defgroup STM3210E_EVAL_SRAM_Private_Var
iables Private_Variables
00120   * @{
00121   */
00122 SRAM_HandleTypeDef sramHandle;
00123 static FSMC_NORSRAM_TimingTypeDef Timing;
00124
00125 /**
00126   * @}
00127   */
00128
00129 /* Private function prototypes -------------
-----------------------------------*/
00130
00131 /** @defgroup STM3210E_EVAL_SRAM_Private_Fun
ction_Prototypes Private_Function_Prototypes
00132   * @{
00133   */
00134 static void SRAM_MspInit(void);
00135
00136 /**
00137   * @}
00138   */
00139
00140 /* Private functions -----------------------
-----------------------------------*/
00141
00142 /** @defgroup STM3210E_EVAL_SRAM_Exported_Fu
nctions Exported_Functions
00143   * @{
00144   */
00145
00146 /**
00147   * @brief  Initializes the SRAM device.
00148   * @retval SRAM status
00149   */
```

```
00150 uint8_t BSP_SRAM_Init(void)
00151 {
00152   sramHandle.Instance  = FSMC_NORSRAM_DEVICE
;
00153   sramHandle.Extended  = FSMC_NORSRAM_EXTEND
ED_DEVICE;
00154
00155   /* SRAM device configuration */
00156   Timing.AddressSetupTime      = 2;
00157   Timing.AddressHoldTime       = 1;
00158   Timing.DataSetupTime         = 2;
00159   Timing.BusTurnAroundDuration = 1;
00160   Timing.CLKDivision           = 2;
00161   Timing.DataLatency           = 2;
00162   Timing.AccessMode            = FSMC_ACCESS
_MODE_A;
00163
00164   sramHandle.Init.NSBank             = FSMC_
NORSRAM_BANK3;
00165   sramHandle.Init.DataAddressMux     = FSMC_
DATA_ADDRESS_MUX_DISABLE;
00166   sramHandle.Init.MemoryType         = FSMC_
MEMORY_TYPE_SRAM;
00167   sramHandle.Init.MemoryDataWidth    = SRAM_
MEMORY_WIDTH;
00168   sramHandle.Init.BurstAccessMode    = SRAM_
BURSTACCESS;
00169   sramHandle.Init.WaitSignalPolarity = FSMC_
WAIT_SIGNAL_POLARITY_LOW;
00170   sramHandle.Init.WrapMode           = FSMC_
WRAP_MODE_DISABLE;
00171   sramHandle.Init.WaitSignalActive   = FSMC_
WAIT_TIMING_BEFORE_WS;
00172   sramHandle.Init.WriteOperation     = FSMC_
WRITE_OPERATION_ENABLE;
00173   sramHandle.Init.WaitSignal         = FSMC_
WAIT_SIGNAL_DISABLE;
```

```
00174   sramHandle.Init.ExtendedMode       = FSMC_
EXTENDED_MODE_DISABLE;
00175   sramHandle.Init.AsynchronousWait   = FSMC_
ASYNCHRONOUS_WAIT_DISABLE;
00176   sramHandle.Init.WriteBurst         = SRAM_
WRITEBURST;
00177
00178   /* SRAM controller initialization */
00179   SRAM_MspInit();
00180
00181   if(HAL_SRAM_Init(&sramHandle, &Timing, &Ti
ming) != HAL_OK)
00182   {
00183     return SRAM_ERROR;
00184   }
00185   else
00186   {
00187     return SRAM_OK;
00188   }
00189 }
00190
00191 /**
00192   * @brief  Reads an amount of data from the
 SRAM device in polling mode.
00193   * @param  uwStartAddress: Read start addre
ss
00194   * @param  pData: Pointer to data to be read

00195   * @param  uwDataSize: Size of read data fr
om the memory
00196   * @retval SRAM status
00197   */
00198 uint8_t BSP_SRAM_ReadData(uint32_t uwStartAd
dress, uint16_t *pData, uint32_t uwDataSize)
00199 {
00200   if(HAL_SRAM_Read_16b(&sramHandle, (uint32_
t *)uwStartAddress, pData, uwDataSize) != HAL_OK)
```

```
00201   {
00202     return SRAM_ERROR;
00203   }
00204   else
00205   {
00206     return SRAM_OK;
00207   }
00208 }
00209
00210 /**
00211   * @brief  Reads an amount of data from the
 SRAM device in DMA mode.
00212   * @param  uwStartAddress: Read start addre
ss
00213   * @param  pData: Pointer to data to be read

00214   * @param  uwDataSize: Size of read data fr
om the memory
00215   * @retval SRAM status
00216   */
00217 uint8_t BSP_SRAM_ReadData_DMA(uint32_t uwSta
rtAddress, uint16_t *pData, uint32_t uwDataSize)
00218 {
00219   if(HAL_SRAM_Read_DMA(&sramHandle, (uint32_
t *)uwStartAddress, (uint32_t *)pData, uwDataSize)
 != HAL_OK)
00220   {
00221     return SRAM_ERROR;
00222   }
00223   else
00224   {
00225     return SRAM_OK;
00226   }
00227 }
00228
00229 /**
00230   * @brief  Writes an amount of data from th
```

```
e SRAM device in polling mode.
00231   * @param  uwStartAddress: Write start addr
ess
00232   * @param  pData: Pointer to data to be wri
tten
00233   * @param  uwDataSize: Size of written data
 from the memory
00234   * @retval SRAM status
00235   */
00236 uint8_t BSP_SRAM_WriteData(uint32_t uwStartA
ddress, uint16_t *pData, uint32_t uwDataSize)
00237 {
00238   if(HAL_SRAM_Write_16b(&sramHandle, (uint32
_t *)uwStartAddress, pData, uwDataSize) != HAL_OK)
00239   {
00240     return SRAM_ERROR;
00241   }
00242   else
00243   {
00244     return SRAM_OK;
00245   }
00246 }
00247
00248 /**
00249   * @brief  Writes an amount of data from th
e SRAM device in DMA mode.
00250   * @param  uwStartAddress: Write start addr
ess
00251   * @param  pData: Pointer to data to be wri
tten
00252   * @param  uwDataSize: Size of written data
 from the memory
00253   * @retval SRAM status
00254   */
00255 uint8_t BSP_SRAM_WriteData_DMA(uint32_t uwSt
artAddress, uint16_t *pData, uint32_t uwDataSize)
00256 {
```

```
00257   if(HAL_SRAM_Write_DMA(&sramHandle, (uint32
_t *)uwStartAddress, (uint32_t *)pData, uwDataSize
) != HAL_OK)
00258   {
00259     return SRAM_ERROR;
00260   }
00261   else
00262   {
00263     return SRAM_OK;
00264   }
00265 }
00266
00267 /**
00268   * @brief  Handles SRAM DMA transfer interr
upt request.
00269   * @retval None
00270   */
00271 void BSP_SRAM_DMA_IRQHandler(void)
00272 {
00273   HAL_DMA_IRQHandler(sramHandle.hdma);
00274 }
00275
00276 /**
00277   * @brief  Initializes SRAM MSP.
00278   * @retval None
00279   */
00280 static void SRAM_MspInit(void)
00281 {
00282   static DMA_HandleTypeDef hdma1;
00283   GPIO_InitTypeDef gpioinitstruct = {0};
00284   /* Enable FSMC clock */
00285   __HAL_RCC_FSMC_CLK_ENABLE();
00286
00287   /* Enable DMA1 and DMA2 clocks */
00288   __HAL_RCC_DMA1_CLK_ENABLE();
00289
00290   /* Enable GPIOs clock */
```

```
00291   __HAL_RCC_GPIOD_CLK_ENABLE();
00292   __HAL_RCC_GPIOE_CLK_ENABLE();
00293   __HAL_RCC_GPIOF_CLK_ENABLE();
00294   __HAL_RCC_GPIOG_CLK_ENABLE();
00295
00296   /* Common GPIO configuration */
00297   gpioinitstruct.Mode      = GPIO_MODE_AF_PP
;
00298   gpioinitstruct.Pull      = GPIO_NOPULL;
00299   gpioinitstruct.Speed     = GPIO_SPEED_FREQ
_HIGH;
00300
00301 /*-- GPIO Configuration --------------------
-----------------------------------*/
00302   /*!< SRAM Data lines configuration */
00303   gpioinitstruct.Pin = GPIO_PIN_0 | GPIO_PIN
_1 | GPIO_PIN_8 | GPIO_PIN_9 |
00304                                        GPIO_PIN_10
| GPIO_PIN_14 | GPIO_PIN_15;
00305   HAL_GPIO_Init(GPIOD, &gpioinitstruct);
00306
00307   gpioinitstruct.Pin = GPIO_PIN_7 | GPIO_PIN
_8 | GPIO_PIN_9 | GPIO_PIN_10 |
00308                                        GPIO_PIN_11
| GPIO_PIN_12 | GPIO_PIN_13 | GPIO_PIN_14 |
00309                                        GPIO_PIN_15;
00310   HAL_GPIO_Init(GPIOE, &gpioinitstruct);
00311
00312   /*!< SRAM Address lines configuration */
00313   gpioinitstruct.Pin = GPIO_PIN_0 | GPIO_PIN
_1 | GPIO_PIN_2 | GPIO_PIN_3 |
00314                                        GPIO_PIN_4 |
 GPIO_PIN_5 | GPIO_PIN_12 | GPIO_PIN_13 |
00315                                        GPIO_PIN_14
| GPIO_PIN_15;
00316   HAL_GPIO_Init(GPIOF, &gpioinitstruct);
00317
```

```
00318   gpioinitstruct.Pin = GPIO_PIN_0 | GPIO_PIN
_1 | GPIO_PIN_2 | GPIO_PIN_3 |
00319                                GPIO_PIN_4 |
 GPIO_PIN_5;
00320   HAL_GPIO_Init(GPIOG, &gpioinitstruct);
00321
00322   gpioinitstruct.Pin = GPIO_PIN_11 | GPIO_PI
N_12 | GPIO_PIN_13;
00323   HAL_GPIO_Init(GPIOD, &gpioinitstruct);
00324
00325   /*!< NOE and NWE configuration */
00326   gpioinitstruct.Pin = GPIO_PIN_4 |GPIO_PIN_
5;
00327   HAL_GPIO_Init(GPIOD, &gpioinitstruct);
00328
00329   /*!< NE3 configuration */
00330   gpioinitstruct.Pin = GPIO_PIN_10 |GPIO_PIN
_12;
00331   HAL_GPIO_Init(GPIOG, &gpioinitstruct);
00332
00333   /*!< NBL0, NBL1 configuration */
00334   gpioinitstruct.Pin = GPIO_PIN_0 | GPIO_PIN
_1;
00335   HAL_GPIO_Init(GPIOE, &gpioinitstruct);
00336
00337   /* Configure common DMA parameters */
00338   hdma1.Init.Direction           = DMA_MEMOR
Y_TO_MEMORY;
00339   hdma1.Init.PeriphInc           = DMA_PINC_
ENABLE;
00340   hdma1.Init.MemInc              = DMA_MINC_
ENABLE;
00341   hdma1.Init.PeriphDataAlignment = DMA_PDATA
ALIGN_HALFWORD;
00342   hdma1.Init.MemDataAlignment    = DMA_MDATA
ALIGN_HALFWORD;
00343   hdma1.Init.Mode                = DMA_NORMA
```

```
L;
00344   hdma1.Init.Priority           = DMA_PRIOR
ITY_HIGH;
00345
00346   hdma1.Instance = DMA1_Channel1;
00347
00348   /* Deinitialize the Stream for new transfe
r */
00349   HAL_DMA_DeInit(&hdma1);
00350
00351   /* Configure the DMA Stream */
00352   HAL_DMA_Init(&hdma1);
00353
00354   /* Associate the DMA handle to the FSMC SR
AM one */
00355   sramHandle.hdma = &hdma1;
00356
00357   /* NVIC configuration for DMA transfer com
plete interrupt */
00358   HAL_NVIC_SetPriority(DMA1_Channel1_IRQn, 0
, 0);
00359   HAL_NVIC_EnableIRQ(DMA1_Channel1_IRQn);
00360
00361 }
00362 /**
00363   * @}
00364   */
00365
00366 /**
00367   * @}
00368   */
00369
00370 /**
00371   * @}
00372   */
00373
00374 /**
```

```
00375   * @}
00376   */
00377
00378 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210E_EVAL BSP User Manual

stm3210e_eval_tsensor.h

Go to the documentation of this file.

```
00001 /**
00002   ********************************************************************************
00003   * @file    stm3210e_eval_tsensor.h
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file contains all the functions prototypes for the
00008   *          stm3210e_tsensor.c firmware driver.
00009   ********************************************************************************
00010   * @attention
00011   *
00012   * <h2><center>&copy; COPYRIGHT(c) 2015 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 __STM3210E_EVAL_TSENSOR_H
00041 #define __STM3210E_EVAL_TSENSOR_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------------------------------------------*/
00048 #include "stm3210e_eval.h"
00049 #include "../Components/stlm75/stlm75.h"
00050
00051 /** @addtogroup BSP
00052   * @{
00053   */
00054
00055 /** @addtogroup STM3210E_EVAL
00056   * @{
00057   */
00058
00059 /** @addtogroup STM3210E_EVAL_TSENSOR
00060   * @{
00061   */
00062
00063 /** @defgroup STM3210E_EVAL_TSENSOR_Exported_Types Exported_Types
00064   * @{
00065   */
00066
00067 /**
```

```
00068   * @brief  TSENSOR Status
00069   */
00070 typedef enum
00071 {
00072   TSENSOR_OK = 0,
00073   TSENSOR_ERROR
00074 }TSENSOR_Status_TypDef;
00075
00076 /**
00077   * @}
00078   */
00079
00080 /** @defgroup STM3210E_EVAL_TSENSOR_Exported
_Constants Exported_Constants
00081   * @{
00082   */
00083 /* Temperature Sensor hardware I2C address */

00084 #define TSENSOR_I2C_ADDRESS_A01 0x90
00085 #define TSENSOR_I2C_ADDRESS_A02 0x92
00086
00087 /* Maximum number of trials use for STLM75_I
sReady function */
00088 #define TSENSOR_MAX_TRIALS      50
00089 /**
00090   * @}
00091   */
00092
00093
00094 /** @addtogroup STM3210E_EVAL_EVAL_I2C_TSENS
OR_Exported_Functions
00095   * @{
00096   */
00097 uint32_t BSP_TSENSOR_Init(void);
00098 uint8_t  BSP_TSENSOR_ReadStatus(void);
00099 uint16_t BSP_TSENSOR_ReadTemp(void);
00100
```

```
00101 #ifdef __cplusplus
00102 }
00103 #endif
00104
00105 #endif /* __STM3210E_EVAL_TSENSOR_H */
00106 /**
00107   * @}
00108   */
00109
00110 /**
00111   * @}
00112   */
00113
00114 /**
00115   * @}
00116   */
00117
00118 /**
00119   * @}
00120   */
00121
00122 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210E_EVAL BSP User Manual

stm3210e_eval_tsensor.c

Go to the documentation of this file.

```
00001 /**
00002   ********************************************************************************
00003   * @file    stm3210e_eval_tsensor.c
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file provides a set of functions needed to manage the I2C STLM75
00008   *          temperature sensor mounted on STM3210E_EVAL board .
00009   *          It implements a high level communication layer for read and write
00010   *          from/to this sensor. The needed STM32L152xD hardware resources (I2C and
00011   *          GPIO) are defined in stm3210e_eval.h file, and the initialization is
00012   *          performed in TSENSOR_IO_Init() function declared in stm3210e_eval.c
00013   *          file.
00014   *          You can easily tailor this driver to any other development board,
```

```
00015   *          by just adapting the defines fo
r hardware resources and
00016   *          TSENSOR_IO_Init() function.
00017   *
00018   *     +-----------------------------------
---------------------------------+
00019   *     |                          Pin assignm
ent                              |
00020   *     +-----------------------------------
-----+--------------+------------+
00021   *     |  STM3210E I2C Pins
     |   STLM75     |   Pin      |
00022   *     +-----------------------------------
-----+--------------+------------+
00023   *     | EVAL_I2C2_SDA_PIN
     |   SDA        |    1       |
00024   *     | EVAL_I2C2_SCL_PIN
     |   SCL        |    2       |
00025   *     | EVAL_I2C2_SMBUS_PIN
     |   OS/INT     |    3       |
00026   *     | .
     |   GND        |    4 (0V)  |
00027   *     | .
     |   A2         |    5       |
00028   *     | .
     |   A1         |    6       |
00029   *     | .
     |   A0         |    7       |
00030   *     | .
     |   VDD        |    8 (3.3V)|
00031   *     +-----------------------------------
-----+--------------+------------+
00032   ***********************************************
**********************************
00033   * @attention
00034   *
00035   * <h2><center>&copy; COPYRIGHT(c) 2015 STM
```

```
icroelectronics</center></h2>
00036   *
00037   * Redistribution and use in source and bin
ary forms, with or without modification,
00038   * are permitted provided that the followin
g conditions are met:
00039   *   1. Redistributions of source code must
 retain the above copyright notice,
00040   *      this list of conditions and the fol
lowing disclaimer.
00041   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00042   *      this list of conditions and the fol
lowing disclaimer in the documentation
00043   *      and/or other materials provided wit
h the distribution.
00044   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00045   *      may be used to endorse or promote p
roducts derived from this software
00046   *      without specific prior written perm
ission.
00047   *
00048   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00049   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00050   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00051   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00052   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00053   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
00054   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
```

```
00055   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00056   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00057   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00058   *
00059   ******************************************
**********************************
00060   */
00061
00062 /* Includes ------------------------------
-----------------------------------*/
00063 #include "stm3210e_eval_tsensor.h"
00064
00065 /** @addtogroup BSP
00066   * @{
00067   */
00068
00069 /** @addtogroup STM3210E_EVAL
00070   * @{
00071   */
00072
00073 /** @addtogroup STM3210E_EVAL_TSENSOR STM321
0E_EVAL TSENSOR
00074   * @brief      This file includes the STLM7
5 Temperature Sensor driver of
00075   *             STM3210E_EVAL board.
00076   * @{
00077   */
00078
00079
00080
00081 /** @defgroup STM3210E_EVAL_EVAL_TSENSOR_Pri
vate_Variables Private Variables
00082   * @{
00083   */
```

```
00084 static TSENSOR_DrvTypeDef  *tsensor_drv;
00085 __IO uint16_t  TSENSORAddress = 0;
00086 /**
00087   * @}
00088   */
00089
00090
00091 /** @defgroup STM3210E_EVAL_EVAL_I2C_TSENSOR
_Exported_Functions Exported Functions
00092   * @{
00093   */
00094
00095 /**
00096   * @brief  Initializes peripherals used by
the I2C Temperature Sensor driver.
00097   * @retval TSENSOR status
00098   */
00099 uint32_t BSP_TSENSOR_Init(void)
00100 {
00101   uint8_t ret = TSENSOR_ERROR;
00102   TSENSOR_InitTypeDef tsensor_initstruct = {
0};
00103
00104   /* Temperature Sensor Initialization */
00105   if(Stlm75Drv.IsReady(TSENSOR_I2C_ADDRESS_A
01, TSENSOR_MAX_TRIALS) == HAL_OK)
00106   {
00107     /* Initialize the temperature sensor dri
ver structure */
00108     TSENSORAddress = TSENSOR_I2C_ADDRESS_A01
;
00109     tsensor_drv = &Stlm75Drv;
00110
00111     ret = TSENSOR_OK;
00112   }
00113   else
00114   {
```

```
00115     if(Stlm75Drv.IsReady(TSENSOR_I2C_ADDRESS
_A02, TSENSOR_MAX_TRIALS) == HAL_OK)
00116     {
00117       /* Initialize the temperature sensor d
river structure */
00118       TSENSORAddress = TSENSOR_I2C_ADDRESS_A
02;
00119       tsensor_drv = &Stlm75Drv;
00120
00121       ret = TSENSOR_OK;
00122     }
00123     else
00124     {
00125       ret = TSENSOR_ERROR;
00126     }
00127   }
00128
00129   if (ret == TSENSOR_OK)
00130   {
00131     /* Configure Temperature Sensor : Conver
sion 9 bits in continuous mode */
00132     /* Alert outside range Limit Temperature
 12 <-> 24c */
00133     tsensor_initstruct.AlertMode
 = STLM75_INTERRUPT_MODE;
00134     tsensor_initstruct.ConversionMode
 = STLM75_CONTINUOUS_MODE;
00135     tsensor_initstruct.TemperatureLimitHigh
 = 24;
00136     tsensor_initstruct.TemperatureLimitLow
 = 12;
00137
00138     /* TSENSOR Init */
00139     tsensor_drv->Init(TSENSORAddress, &tsens
or_initstruct);
00140
00141     ret = TSENSOR_OK;
```

```
00142   }
00143
00144   return ret;
00145 }
00146
00147 /**
00148   * @brief  Returns the Temperature Sensor s
tatus.
00149   * @retval The Temperature Sensor status.
00150   */
00151 uint8_t BSP_TSENSOR_ReadStatus(void)
00152 {
00153   return (tsensor_drv->ReadStatus(TSENSORAdd
ress));
00154 }
00155
00156 /**
00157   * @brief  Read Temperature register of STL
M75.
00158   * @retval STLM75 measured temperature valu
e.
00159   */
00160 uint16_t BSP_TSENSOR_ReadTemp(void)
00161 {
00162   return tsensor_drv->ReadTemp(TSENSORAddress
);
00163
00164 }
00165
00166 /**
00167   * @}
00168   */
00169
00170
00171 /**
00172   * @}
00173   */
```

```
00174
00175
00176 /**
00177   * @}
00178   */
00179
00180 /**
00181   * @}
00182   */
00183
00184 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210E_EVAL BSP User Manual

Modules

BSP

Modules

STM3210E-EVAL

Generated on Thu Dec 10 2015 17:39:43 for STM3210E_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210E_EVAL BSP User Manual

STM3210E-EVAL

BSP

Modules

STM3210E-EVAL Common

STM3210E-EVAL LCD

STM3210E_EVAL NAND

STM3210E_EVAL NOR

STM3210E-EVAL SD

STM3210E_EVAL Serial FLASH

This file includes the M25Pxxx SPI FLASH driver of STM3210E-EVAL board.

STM3210E_EVAL SRAM

STM3210E_EVAL_AUDIO

This file includes the low layer audio driver available on STM3210E-Eval eval board.

STM3210E_EVAL TSENSOR

This file includes the STLM75 Temperature Sensor driver of STM3210E_EVAL board.

STM3210E_EVAL BSP User Manual

STM3210E-EVAL Common

STM3210E-EVAL

Modules

- Private Types Definitions
- Private Defines
- Private Variables
- Exported Functions
- Bus Operations Functions
- Link Operations Functions
- Exported Types
- Exported Constants

STM3210E_EVAL BSP User Manual

Exported Constants

STM3210E-EVAL Common

Modules

STM3210E_EVAL_LED Define for STM3210E_EVAL board.
STM3210E_EVAL_BUTTON
STM3210E_EVAL_COM
STM3210E_EVAL_BUS
STM3210E_EVAL_COMPONENT

STM3210E_EVAL BSP User Manual

STM3210E-EVAL LCD

STM3210E-EVAL

Modules

- Private Defines
- Private Macros
- Private Variables
- Private Functions
- Exported Functions
- Exported Types
- Exported Constants

STM3210E_EVAL BSP User Manual

STM3210E-EVAL SD

STM3210E-EVAL

Modules

- Private Variables
- Private Functions
- Exported_Functions
- Exported_Types
- Exported_Constants
- STM3210E_EVAL_SD_Exported_Functions

STM3210E_EVAL BSP User Manual

STM3210E_EVAL TSENSOR

STM3210E-EVAL

This file includes the STLM75 Temperature Sensor driver of STM3210E_EVAL board. More...

Modules

- Private Variables
- Exported Functions
- Exported_Types
- Exported_Constants

Detailed Description

This file includes the STLM75 Temperature Sensor driver of STM3210E_EVAL board.
