STM32446E_EVAL BSP User Manual

Data Fields

LCD_CONTROLLER_TypeDef Struct Reference

STM32446E EVAL LOW LEVEL Private TypesDefinitions

Data Fields

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

Detailed Description

Definition at line **69** of file **stm32446e_eval.c**.

Field Documentation

__IO uint16_t LCD_CONTROLLER_TypeDef::RAM

Definition at line **72** of file **stm32446e_eval.c**.

__IO uint16_t LCD_CONTROLLER_TypeDef::REG

Definition at line **71** of file **stm32446e_eval.c**.

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

- **stm32446e_eval.c**

STM32446E_EVAL BSP User Manual

STM32446E EVAL LOW LEVEL Private Macros

STM32446E EVAL LOW LEVEL

Generated on Fri Jan 15 2016 10:06:21 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL LOW LEVEL Exported Macros

STM32446E EVAL LOW LEVEL

Generated on Fri Jan 15 2016 10:06:21 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL AUDIO Private Types

STM32446E EVAL AUDIO

Generated on Fri Jan 15 2016 10:06:21 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL AUDIO Private Defines

STM32446E EVAL AUDIO

Generated on Fri Jan 15 2016 10:06:21 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL AUDIO Private Macros

STM32446E EVAL AUDIO

Generated on Fri Jan 15 2016 10:06:21 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL AUDIO Exported Types

STM32446E EVAL AUDIO

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL CAMERA Private TypesDefinitions

STM32446E-EVAL CAMERA

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL CAMERA Private Defines

STM32446E-EVAL CAMERA

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL CAMERA Private Macros

STM32446E-EVAL CAMERA

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL CAMERA Private FunctionPrototypes

STM32446E-EVAL CAMERA

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL CAMERA Exported Constants

STM32446E-EVAL CAMERA

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL EEPROM Private Types

STM32446E EVAL EEPROM

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL EEPROM Private Defines

STM32446E EVAL EEPROM

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL EEPROM Private Macros

STM32446E EVAL EEPROM

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL EEPROM Private Function Prototypes

STM32446E EVAL EEPROM

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL EEPROM Exported Types

STM32446E EVAL EEPROM

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL EEPROM Exported Macros

STM32446E EVAL EEPROM

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL IO Private Types Definitions

STM32446E EVAL IO

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL IO Private Defines

STM32446E EVAL IO

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL IO Private Macros

STM32446E EVAL IO

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL IO Private Function Prototypes

STM32446E EVAL IO

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL IO Exported Macro

STM32446E EVAL IO

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL LCD Private TypesDefinitions

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL LCD Private FunctionPrototypes

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

LCD_DrawPropTypeDef Struct Reference

STM32446E EVAL LCD Exported Types

```
#include <stm32446e_eval_lcd.h>
```

Data Fields

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

Detailed Description

Definition at line **67** of file **stm32446e_eval_lcd.h**.

Field Documentation

uint32_t LCD_DrawPropTypeDef::BackColor

Definition at line 70 of file stm32446e_eval_lcd.h.

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

sFONT* LCD_DrawPropTypeDef::pFont

Definition at line 71 of file stm32446e_eval_lcd.h.

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

uint32_t LCD_DrawPropTypeDef::TextColor

Definition at line 69 of file stm32446e_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 DrawChar().

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

- stm32446e_eval_lcd.h

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP
User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

Point Struct Reference

STM32446E EVAL LCD Exported Constants

```
#include <stm32446e_eval_lcd.h>
```

Data Fields

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

Detailed Description

Definition at line **87** of file **stm32446e_eval_lcd.h**.

Field Documentation

int16_t Point::X

Definition at line **89** of file **stm32446e_eval_lcd.h**.

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

int16_t Point::Y

Definition at line **90** of file **stm32446e_eval_lcd.h**.

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

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

- **stm32446e_eval_lcd.h**

STM32446E_EVAL BSP User Manual

QSPI_Info Struct Reference

STM32446E EVAL QSPI Exported Types

```
#include <stm32446e_eval_qspi.h>
```

Data Fields

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

Detailed Description

Definition at line **123** of file **stm32446e_eval_qspi.h**.

Field Documentation

uint32_t QSPI_Info::EraseSectorSize

Size of sectors for the erase operation

Definition at line **125** of file **stm32446e_eval_qspi.h**.

Referenced by **BSP_QSPI_GetInfo()**.

uint32_t QSPI_Info::EraseSectorsNumber

Number of sectors for the erase operation

Definition at line **126** of file **stm32446e_eval_qspi.h**.

Referenced by **BSP_QSPI_GetInfo()**.

uint32_t QSPI_Info::FlashSize

Size of the flash

Definition at line **124** of file **stm32446e_eval_qspi.h**.

Referenced by **BSP_QSPI_GetInfo()**.

uint32_t QSPI_Info::ProgPageSize

Size of pages for the program operation

Definition at line **127** of file **stm32446e_eval_qspi.h**.

Referenced by **BSP_QSPI_GetInfo()**.

uint32_t QSPI_Info::ProgPagesNumber

Number of pages for the program operation

Definition at line **128** of file **stm32446e_eval_qspi.h**.

Referenced by **BSP_QSPI_GetInfo()**.

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

- **stm32446e_eval_qspi.h**

STM32446E_EVAL BSP User Manual

STM32446E EVAL SD Private TypesDefinitions

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL SD Private Defines

STM32446E EVAL SD

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL SD Private Macros

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL SD Private FunctionPrototypes

STM32446E EVAL SD

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL SD Exported Macro

STM32446E EVAL SD

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL SDRAM Private Types Definitions

STM32446E EVAL SDRAM

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL SDRAM Private Defines

STM32446E EVAL SDRAM

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL SDRAM Private Macros

STM32446E EVAL SDRAM

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL SDRAM Private Function Prototypes

STM32446E EVAL SDRAM

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL SDRAM Exported Macro

STM32446E EVAL SDRAM

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL TS Private Types Definitions

STM32446E EVAL TS

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL TS Private Defines

STM32446E EVAL TS

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL TS Private Macros

STM32446E EVAL TS

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL TS Private Function Prototypes

STM32446E EVAL TS

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

TS_StateTypeDef Struct Reference

STM32446E EVAL TS Exported Types

`#include <stm32446e_eval_ts.h>`

Data Fields

uint16_t	**TouchDetected**
uint16_t	**x**
uint16_t	**y**
uint16_t	**z**

Detailed Description

Definition at line **67** of file **stm32446e_eval_ts.h**.

Field Documentation

uint16_t TS_StateTypeDef::TouchDetected

Definition at line 69 of file stm32446e_eval_ts.h.

Referenced by BSP_TS_GetState().

uint16_t TS_StateTypeDef::x

Definition at line 70 of file stm32446e_eval_ts.h.

Referenced by BSP_TS_GetState().

uint16_t TS_StateTypeDef::y

Definition at line 71 of file stm32446e_eval_ts.h.

Referenced by BSP_TS_GetState().

uint16_t TS_StateTypeDef::z

Definition at line 72 of file stm32446e_eval_ts.h.

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

- stm32446e_eval_ts.h

STM32446E_EVAL BSP User Manual

STM32446E EVAL TS Exported Macros

STM32446E EVAL TS

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_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**
- EraseSectorSize : **QSPI_Info**
- EraseSectorsNumber : **QSPI_Info**
- FlashSize : **QSPI_Info**
- pFont : **LCD_DrawPropTypeDef**
- ProgPageSize : **QSPI_Info**
- ProgPagesNumber : **QSPI_Info**
- RAM : **LCD_CONTROLLER_TypeDef**
- REG : **LCD_CONTROLLER_TypeDef**
- TextColor : **LCD_DrawPropTypeDef**
- TouchDetected : **TS_StateTypeDef**
- X : **Point**
- x : **TS_StateTypeDef**
- Y : **Point**
- y : **TS_StateTypeDef**
- z : **TS_StateTypeDef**

STM32446E_EVAL BSP User Manual

- BackColor : **LCD_DrawPropTypeDef**
- EraseSectorSize : **QSPI_Info**
- EraseSectorsNumber : **QSPI_Info**
- FlashSize : **QSPI_Info**
- pFont : **LCD_DrawPropTypeDef**
- ProgPageSize : **QSPI_Info**
- ProgPagesNumber : **QSPI_Info**
- RAM : **LCD_CONTROLLER_TypeDef**
- REG : **LCD_CONTROLLER_TypeDef**
- TextColor : **LCD_DrawPropTypeDef**
- TouchDetected : **TS_StateTypeDef**
- X : **Point**
- x : **TS_StateTypeDef**
- Y : **Point**
- y : **TS_StateTypeDef**
- z : **TS_StateTypeDef**

STM32446E_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_CLK_DISABLE : **stm32446e_eval_sdram.h**
- __DMAx_CLK_ENABLE : **stm32446e_eval_sdram.h**
- __DMAx_TxRx_CLK_ENABLE : **stm32446e_eval_sd.h**
- __STM32446E_EVAL_BSP_VERSION : **stm32446e_eval.c**
- __STM32446E_EVAL_BSP_VERSION_MAIN : **stm32446e_eval.c**
- __STM32446E_EVAL_BSP_VERSION_RC : **stm32446e_eval.c**
- __STM32446E_EVAL_BSP_VERSION_SUB1 : **stm32446e_eval.c**
- __STM32446E_EVAL_BSP_VERSION_SUB2 : **stm32446e_eval.c**

STM32446E_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 : **stm32446e_eval_lcd.c**
- AddressI2C : **stm32446e_eval_ts.c**
- audio_drv : **stm32446e_eval_audio.c**
- AUDIO_ERROR : **stm32446e_eval_audio.h**
- AUDIO_I2C_ADDRESS : **stm32446e_eval.h**
- AUDIO_I2Sx : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_CLK_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_CLK_ENABLE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_DMAx_CHANNEL : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_DMAx_CLK_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_DMAx_CLK_ENABLE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_DMAx_IRQ : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_DMAx_IRQHandler : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_DMAx_MEM_DATA_SIZE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_DMAx_PERIPH_DATA_SIZE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_DMAx_STREAM : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_SCK_AF : **stm32446e_eval_audio.h**

- AUDIO_I2Sx_SCK_GPIO_CLK_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_SCK_GPIO_CLK_ENABLE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_SCK_GPIO_PORT : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_SCK_PIN : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_SD_AF : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_SD_GPIO_CLK_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_SD_GPIO_CLK_ENABLE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_SD_GPIO_PORT : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_SD_PIN : **stm32446e_eval_audio.h**
- AUDIO_IN_IRQ_PREPRIO : **stm32446e_eval_audio.h**
- AUDIO_IO_DeInit() : **stm32446e_eval.c**
- AUDIO_IO_Delay() : **stm32446e_eval.c**
- AUDIO_IO_Init() : **stm32446e_eval.c**
- AUDIO_IO_Read() : **stm32446e_eval.c**
- AUDIO_IO_Write() : **stm32446e_eval.c**
- AUDIO_OK : **stm32446e_eval_audio.h**
- AUDIO_OUT_IRQ_PREPRIO : **stm32446e_eval_audio.h**
- AUDIO_SAIx : **stm32446e_eval_audio.h**
- AUDIO_SAIx_CLK_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_SAIx_CLK_ENABLE : **stm32446e_eval_audio.h**
- AUDIO_SAIx_DMAx_CHANNEL : **stm32446e_eval_audio.h**
- AUDIO_SAIx_DMAx_CLK_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_SAIx_DMAx_CLK_ENABLE : **stm32446e_eval_audio.h**
- AUDIO_SAIx_DMAx_IRQ : **stm32446e_eval_audio.h**
- AUDIO_SAIx_DMAx_IRQHandler : **stm32446e_eval_audio.h**
- AUDIO_SAIx_DMAx_MEM_DATA_SIZE : **stm32446e_eval_audio.h**
- AUDIO_SAIx_DMAx_PERIPH_DATA_SIZE : **stm32446e_eval_audio.h**
- AUDIO_SAIx_DMAx_STREAM : **stm32446e_eval_audio.h**
- AUDIO_SAIx_FS_PIN : **stm32446e_eval_audio.h**
- AUDIO_SAIx_MCK_PIN : **stm32446e_eval_audio.h**
- AUDIO_SAIx_MCLK_SCK_DISABLE : **stm32446e_eval_audio.h**

- AUDIO_SAIx_MCLK_SCK_ENABLE : **stm32446e_eval_audio.h**
- AUDIO_SAIx_MCLK_SCK_GPIO_PORT : **stm32446e_eval_audio.h**
- AUDIO_SAIx_MCLK_SD_FS_AF : **stm32446e_eval_audio.h**
- AUDIO_SAIx_SCK_AF : **stm32446e_eval_audio.h**
- AUDIO_SAIx_SCK_PIN : **stm32446e_eval_audio.h**
- AUDIO_SAIx_SD_FS_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_SAIx_SD_FS_ENABLE : **stm32446e_eval_audio.h**
- AUDIO_SAIx_SD_FS_GPIO_PORT : **stm32446e_eval_audio.h**
- AUDIO_SAIx_SD_PIN : **stm32446e_eval_audio.h**
- AUDIO_TIMEOUT : **stm32446e_eval_audio.h**
- AUDIO_TIMx : **stm32446e_eval_audio.h**
- AUDIO_TIMx_AF : **stm32446e_eval_audio.h**
- AUDIO_TIMx_CLK_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_TIMx_CLK_ENABLE : **stm32446e_eval_audio.h**
- AUDIO_TIMx_GPIO_CLK_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_TIMx_GPIO_CLK_ENABLE : **stm32446e_eval_audio.h**
- AUDIO_TIMx_GPIO_PORT : **stm32446e_eval_audio.h**
- AUDIO_TIMx_IN_CHANNEL : **stm32446e_eval_audio.h**
- AUDIO_TIMx_IN_GPIO_PIN : **stm32446e_eval_audio.h**
- AUDIO_TIMx_OUT_CHANNEL : **stm32446e_eval_audio.h**
- AUDIO_TIMx_OUT_GPIO_PIN : **stm32446e_eval_audio.h**
- AUDIODATA_SIZE : **stm32446e_eval_audio.h**
- AudioInVolume : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**

STM32446E_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 -

- BSP_AUDIO_IN_DeInit() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_IN_Error_Callback() : **stm32446e_eval_audio.h** , **stm32446e_eval_audio.c**
- BSP_AUDIO_IN_HalfTransfer_CallBack() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_IN_Init() : **stm32446e_eval_audio.h** , **stm32446e_eval_audio.c**
- BSP_AUDIO_IN_MspDeInit() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_IN_MspInit() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_IN_Pause() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_IN_PDMToPCM() : **stm32446e_eval_audio.h** , **stm32446e_eval_audio.c**
- BSP_AUDIO_IN_Record() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_IN_Resume() : **stm32446e_eval_audio.c** ,

stm32446e_eval_audio.h

- BSP_AUDIO_IN_SetVolume() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_IN_Stop() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_IN_TransferComplete_CallBack() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_ChangeBuffer() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_ClockConfig() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_DeInit() : **stm32446e_eval_audio.h** , **stm32446e_eval_audio.c**
- BSP_AUDIO_OUT_Error_CallBack() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_HalfTransfer_CallBack() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_Init() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_MspDeInit() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_MspInit() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_Pause() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_Play() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_Resume() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_SetAudioFrameSlot() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_SetFrequency() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_SetMute() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_SetOutputMode() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**

- BSP_AUDIO_OUT_SetVolume() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_Stop() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_TransferComplete_CallBack() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_CAMERA_BlackWhiteConfig() : **stm32446e_eval_camera.h** , **stm32446e_eval_camera.c**
- BSP_CAMERA_ColorEffectConfig() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_ContinuousStart() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_ContrastBrightnessConfig() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_DeInit() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_DMA_IRQHandler() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_ErrorCallback() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_FrameEventCallback() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_HwReset() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_Init() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_IRQHandler() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_LineEventCallback() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_MspDeInit() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_MspInit() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_PwrDown() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_Resume() : **stm32446e_eval_camera.c** ,

stm32446e_eval_camera.h

- BSP_CAMERA_SnapshotStart() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_Stop() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_Suspend() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_VsyncEventCallback() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_COM_DeInit() : **stm32446e_eval.c** , **stm32446e_eval.h**
- BSP_COM_Init() : **stm32446e_eval.c** , **stm32446e_eval.h**
- BSP_EEPROM_DeInit() : **stm32446e_eval_eeprom.c** , **stm32446e_eval_eeprom.h**
- BSP_EEPROM_Init() : **stm32446e_eval_eeprom.c** , **stm32446e_eval_eeprom.h**
- BSP_EEPROM_ReadBuffer() : **stm32446e_eval_eeprom.c** , **stm32446e_eval_eeprom.h**
- BSP_EEPROM_TIMEOUT_UserCallback() : **stm32446e_eval_eeprom.c** , **stm32446e_eval_eeprom.h**
- BSP_EEPROM_WaitEepromStandbyState() : **stm32446e_eval_eeprom.c** , **stm32446e_eval_eeprom.h**
- BSP_EEPROM_WriteBuffer() : **stm32446e_eval_eeprom.c** , **stm32446e_eval_eeprom.h**
- BSP_EEPROM_WritePage() : **stm32446e_eval_eeprom.c** , **stm32446e_eval_eeprom.h**
- BSP_GetVersion() : **stm32446e_eval.c** , **stm32446e_eval.h**
- BSP_IO_ConfigIrqOutPin() : **stm32446e_eval_io.c** , **stm32446e_eval_io.h**
- BSP_IO_ConfigPin() : **stm32446e_eval_io.c** , **stm32446e_eval_io.h**
- BSP_IO_DeInit() : **stm32446e_eval_io.h** , **stm32446e_eval_io.c**
- BSP_IO_Init() : **stm32446e_eval_io.c** , **stm32446e_eval_io.h**
- BSP_IO_ITClear() : **stm32446e_eval_io.c** , **stm32446e_eval_io.h**
- BSP_IO_ITGetStatus() : **stm32446e_eval_io.c** , **stm32446e_eval_io.h**
- BSP_IO_PIN_RESET : **stm32446e_eval_io.h**
- BSP_IO_PIN_SET : **stm32446e_eval_io.h**

- BSP_IO_PinStateTypeDef : **stm32446e_eval_io.h**
- BSP_IO_ReadPin() : **stm32446e_eval_io.c** , **stm32446e_eval_io.h**
- BSP_IO_TogglePin() : **stm32446e_eval_io.c** , **stm32446e_eval_io.h**
- BSP_IO_WritePin() : **stm32446e_eval_io.c** , **stm32446e_eval_io.h**
- BSP_LCD_Clear() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_ClearStringLine() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DeInit() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DisplayChar() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DisplayOff() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DisplayOn() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DisplayStringAt() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DisplayStringAtLine() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DrawBitmap() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DrawCircle() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DrawEllipse() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DrawHLine() : **stm32446e_eval_lcd.h** , **stm32446e_eval_lcd.c**
- BSP_LCD_DrawLine() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DrawPixel() : **stm32446e_eval_lcd.h** , **stm32446e_eval_lcd.c**
- BSP_LCD_DrawPolygon() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**

- BSP_LCD_DrawRect() : **stm32446e_eval_lcd.h** , **stm32446e_eval_lcd.c**
- BSP_LCD_DrawRGBImage() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DrawVLine() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_FillCircle() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_FillEllipse() : **stm32446e_eval_lcd.h** , **stm32446e_eval_lcd.c**
- BSP_LCD_FillPolygon() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_FillRect() : **stm32446e_eval_lcd.h** , **stm32446e_eval_lcd.c**
- BSP_LCD_GetBackColor() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_GetFont() : **stm32446e_eval_lcd.h** , **stm32446e_eval_lcd.c**
- BSP_LCD_GetTextColor() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_GetXSize() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_GetYSize() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_Init() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_ReadPixel() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_SetBackColor() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_SetFont() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_SetTextColor() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LED_DeInit() : **stm32446e_eval.c** , **stm32446e_eval.h**
- BSP_LED_Init() : **stm32446e_eval.c** , **stm32446e_eval.h**
- BSP_LED_Off() : **stm32446e_eval.c** , **stm32446e_eval.h**

- BSP_LED_On() : **stm32446e_eval.h** , **stm32446e_eval.c**
- BSP_LED_Toggle() : **stm32446e_eval.c** , **stm32446e_eval.h**
- BSP_PB_DeInit() : **stm32446e_eval.c** , **stm32446e_eval.h**
- BSP_PB_GetState() : **stm32446e_eval.h** , **stm32446e_eval.c**
- BSP_PB_Init() : **stm32446e_eval.c** , **stm32446e_eval.h**
- BSP_QSPI_DeInit() : **stm32446e_eval_qspi.c**
- BSP_QSPI_Erase_Block() : **stm32446e_eval_qspi.c**
- BSP_QSPI_Erase_Chip() : **stm32446e_eval_qspi.c**
- BSP_QSPI_GetInfo() : **stm32446e_eval_qspi.c**
- BSP_QSPI_GetStatus() : **stm32446e_eval_qspi.c**
- BSP_QSPI_Init() : **stm32446e_eval_qspi.c**
- BSP_QSPI_MemoryMappedMode() : **stm32446e_eval_qspi.c**
- BSP_QSPI_MspDeInit() : **stm32446e_eval_qspi.c** , **stm32446e_eval_qspi.h**
- BSP_QSPI_MspInit() : **stm32446e_eval_qspi.c** , **stm32446e_eval_qspi.h**
- BSP_QSPI_Read() : **stm32446e_eval_qspi.c**
- BSP_QSPI_Write() : **stm32446e_eval_qspi.c**
- BSP_SD_DeInit() : **stm32446e_eval_sd.h** , **stm32446e_eval_sd.c**
- BSP_SD_DetectCallback() : **stm32446e_eval_sd.h**
- BSP_SD_DetectIT() : **stm32446e_eval_sd.h**
- BSP_SD_DMA_Rx_IRQHandler() : **stm32446e_eval_sd.h** , **stm32446e_eval_sd.c**
- BSP_SD_DMA_Tx_IRQHandler() : **stm32446e_eval_sd.c** , **stm32446e_eval_sd.h**
- BSP_SD_Erase() : **stm32446e_eval_sd.c** , **stm32446e_eval_sd.h**
- BSP_SD_GetCardInfo() : **stm32446e_eval_sd.c** , **stm32446e_eval_sd.h**
- BSP_SD_GetStatus() : **stm32446e_eval_sd.c** , **stm32446e_eval_sd.h**
- BSP_SD_Init() : **stm32446e_eval_sd.c** , **stm32446e_eval_sd.h**
- BSP_SD_IRQHandler() : **stm32446e_eval_sd.c** , **stm32446e_eval_sd.h**
- BSP_SD_IsDetected() : **stm32446e_eval_sd.h** , **stm32446e_eval_sd.c**

- BSP_SD_ITConfig() : **stm32446e_eval_sd.c** , **stm32446e_eval_sd.h**
- BSP_SD_MspDeInit() : **stm32446e_eval_sd.h** , **stm32446e_eval_sd.c**
- BSP_SD_MspInit() : **stm32446e_eval_sd.h** , **stm32446e_eval_sd.c**
- BSP_SD_ReadBlocks() : **stm32446e_eval_sd.h** , **stm32446e_eval_sd.c**
- BSP_SD_ReadBlocks_DMA() : **stm32446e_eval_sd.h** , **stm32446e_eval_sd.c**
- BSP_SD_WriteBlocks() : **stm32446e_eval_sd.h** , **stm32446e_eval_sd.c**
- BSP_SD_WriteBlocks_DMA() : **stm32446e_eval_sd.c** , **stm32446e_eval_sd.h**
- BSP_SDRAM_DeInit() : **stm32446e_eval_sdram.h** , **stm32446e_eval_sdram.c**
- BSP_SDRAM_DMA_IRQHandler() : **stm32446e_eval_sdram.h** , **stm32446e_eval_sdram.c**
- BSP_SDRAM_Init() : **stm32446e_eval_sdram.h** , **stm32446e_eval_sdram.c**
- BSP_SDRAM_Initialization_sequence() : **stm32446e_eval_sdram.h** , **stm32446e_eval_sdram.c**
- BSP_SDRAM_MspDeInit() : **stm32446e_eval_sdram.h** , **stm32446e_eval_sdram.c**
- BSP_SDRAM_MspInit() : **stm32446e_eval_sdram.h** , **stm32446e_eval_sdram.c**
- BSP_SDRAM_ReadData() : **stm32446e_eval_sdram.c** , **stm32446e_eval_sdram.h**
- BSP_SDRAM_ReadData_DMA() : **stm32446e_eval_sdram.c** , **stm32446e_eval_sdram.h**
- BSP_SDRAM_Sendcmd() : **stm32446e_eval_sdram.h** , **stm32446e_eval_sdram.c**
- BSP_SDRAM_WriteData() : **stm32446e_eval_sdram.h** , **stm32446e_eval_sdram.c**
- BSP_SDRAM_WriteData_DMA() : **stm32446e_eval_sdram.c** , **stm32446e_eval_sdram.h**
- BSP_TS_DeInit() : **stm32446e_eval_ts.h** , **stm32446e_eval_ts.c**

- BSP_TS_FIFOClear() : **stm32446e_eval_ts.h** , **stm32446e_eval_ts.c**
- BSP_TS_GetState() : **stm32446e_eval_ts.c** , **stm32446e_eval_ts.h**
- BSP_TS_Init() : **stm32446e_eval_ts.h** , **stm32446e_eval_ts.c**
- BSP_TS_ITClear() : **stm32446e_eval_ts.c** , **stm32446e_eval_ts.h**
- BSP_TS_ITConfig() : **stm32446e_eval_ts.c** , **stm32446e_eval_ts.h**
- BSP_TS_ITDisable() : **stm32446e_eval_ts.h** , **stm32446e_eval_ts.c**
- BSP_TS_ITEnable() : **stm32446e_eval_ts.c** , **stm32446e_eval_ts.h**
- BSP_TS_ITGetStatus() : **stm32446e_eval_ts.h** , **stm32446e_eval_ts.c**
- BUTTON_IRQn : **stm32446e_eval.c**
- BUTTON_KEY : **stm32446e_eval.h**
- BUTTON_MODE_EXTI : **stm32446e_eval.h**
- BUTTON_MODE_GPIO : **stm32446e_eval.h**
- BUTTON_PIN : **stm32446e_eval.c**
- BUTTON_PORT : **stm32446e_eval.c**
- BUTTON_TAMPER : **stm32446e_eval.h**
- Button_TypeDef : **stm32446e_eval.h**
- BUTTON_WAKEUP : **stm32446e_eval.h**
- ButtonMode_TypeDef : **stm32446e_eval.h**
- BUTTONn : **stm32446e_eval.h**
- BUTTONx_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- BUTTONx_GPIO_CLK_ENABLE : **stm32446e_eval.h**

STM32446E_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 -

- CAMERA_Delay() : **stm32446e_eval.c**
- camera_drv : **stm32446e_eval_camera.c**
- CAMERA_ERROR : **stm32446e_eval_camera.h**
- CAMERA_I2C_ADDRESS : **stm32446e_eval.h**
- CAMERA_IO_Init() : **stm32446e_eval.c**
- CAMERA_IO_Read() : **stm32446e_eval.c**
- CAMERA_IO_Write() : **stm32446e_eval.c**
- CAMERA_OK : **stm32446e_eval_camera.h**
- Camera_StatusTypeDef : **stm32446e_eval_camera.h**
- CAMERA_TIMEOUT : **stm32446e_eval_camera.h**
- cameraHwAddress : **stm32446e_eval_camera.c**
- CENTER_MODE : **stm32446e_eval_lcd.h**
- Channel_Demux : **stm32446e_eval_audio.c**
- CHANNEL_DEMUX_MASK : **stm32446e_eval_audio.h**
- CODEC_AUDIOFRAME_SLOT_0123 : **stm32446e_eval_audio.h**
- CODEC_AUDIOFRAME_SLOT_02 : **stm32446e_eval_audio.h**
- CODEC_AUDIOFRAME_SLOT_13 : **stm32446e_eval_audio.h**
- CODEC_RESET_DELAY : **stm32446e_eval_audio.h**
- COM1 : **stm32446e_eval.h**

- COM2 : **stm32446e_eval.h**
- COM_RX_AF : **stm32446e_eval.c**
- COM_RX_PIN : **stm32446e_eval.c**
- COM_RX_PORT : **stm32446e_eval.c**
- COM_TX_AF : **stm32446e_eval.c**
- COM_TX_PIN : **stm32446e_eval.c**
- COM_TX_PORT : **stm32446e_eval.c**
- COM_TypeDef : **stm32446e_eval.h**
- COM_USART : **stm32446e_eval.c**
- Command : **stm32446e_eval_sdram.c**
- COMn : **stm32446e_eval.h**
- currentResolution : **stm32446e_eval_camera.c**

STM32446E_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 -

- DEFAULT_AUDIO_IN_BIT_RESOLUTION : **stm32446e_eval_audio.h**
- DEFAULT_AUDIO_IN_CHANNEL_NBR : **stm32446e_eval_audio.h**
- DEFAULT_AUDIO_IN_FREQ : **stm32446e_eval_audio.h**
- DEFAULT_AUDIO_IN_VOLUME : **stm32446e_eval_audio.h**
- DMA_MAX : **stm32446e_eval_audio.h**
- DMA_MAX_SZE : **stm32446e_eval_audio.h**
- DrawChar() : **stm32446e_eval_lcd.c**
- DrawProp : **stm32446e_eval_lcd.c**

STM32446E_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 -

- EEPROM_FAIL : **stm32446e_eval_eeprom.h**
- EEPROM_I2C_ADDRESS_A01 : **stm32446e_eval.h**
- EEPROM_I2C_ADDRESS_A02 : **stm32446e_eval.h**
- EEPROM_IO_Init() : **stm32446e_eval.c** , **stm32446e_eval_eeprom.h**
- EEPROM_IO_IsDeviceReady() : **stm32446e_eval.c** , **stm32446e_eval_eeprom.h**
- EEPROM_IO_ReadData() : **stm32446e_eval.c** , **stm32446e_eval_eeprom.h**
- EEPROM_IO_WriteData() : **stm32446e_eval.c** , **stm32446e_eval_eeprom.h**
- EEPROM_MAX_SIZE : **stm32446e_eval_eeprom.h**
- EEPROM_MAX_TRIALS : **stm32446e_eval_eeprom.h**
- EEPROM_OK : **stm32446e_eval_eeprom.h**
- EEPROM_PAGESIZE : **stm32446e_eval_eeprom.h**
- EEPROM_TIMEOUT : **stm32446e_eval_eeprom.h**
- EEPROMAddress : **stm32446e_eval_eeprom.c**
- EEPROMDataRead : **stm32446e_eval_eeprom.c**
- EEPROMDataWrite : **stm32446e_eval_eeprom.c**

- EVAL_COM1 : **stm32446e_eval.h**
- EVAL_COM1_CLK_DISABLE : **stm32446e_eval.h**
- EVAL_COM1_CLK_ENABLE : **stm32446e_eval.h**
- EVAL_COM1_IRQn : **stm32446e_eval.h**
- EVAL_COM1_RX_AF : **stm32446e_eval.h**
- EVAL_COM1_RX_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- EVAL_COM1_RX_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- EVAL_COM1_RX_GPIO_PORT : **stm32446e_eval.h**
- EVAL_COM1_RX_PIN : **stm32446e_eval.h**
- EVAL_COM1_TX_AF : **stm32446e_eval.h**
- EVAL_COM1_TX_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- EVAL_COM1_TX_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- EVAL_COM1_TX_GPIO_PORT : **stm32446e_eval.h**
- EVAL_COM1_TX_PIN : **stm32446e_eval.h**
- EVAL_COMx_CLK_DISABLE : **stm32446e_eval.h**
- EVAL_COMx_CLK_ENABLE : **stm32446e_eval.h**
- EVAL_COMx_RX_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- EVAL_COMx_RX_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- EVAL_COMx_TX_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- EVAL_COMx_TX_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- EVAL_DMAx_CLK_ENABLE : **stm32446e_eval.h**
- EVAL_I2Cx : **stm32446e_eval.h**
- EVAL_I2Cx_CLK_DISABLE : **stm32446e_eval.h**
- EVAL_I2Cx_CLK_ENABLE : **stm32446e_eval.h**
- EVAL_I2Cx_ER_IRQn : **stm32446e_eval.h**
- EVAL_I2Cx_EV_IRQn : **stm32446e_eval.h**
- EVAL_I2Cx_FORCE_RESET : **stm32446e_eval.h**
- EVAL_I2Cx_RELEASE_RESET : **stm32446e_eval.h**
- EVAL_I2Cx_SCL_PIN : **stm32446e_eval.h**
- EVAL_I2Cx_SCL_SDA_AF : **stm32446e_eval.h**
- EVAL_I2Cx_SCL_SDA_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- EVAL_I2Cx_SCL_SDA_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- EVAL_I2Cx_SCL_SDA_GPIO_PORT : **stm32446e_eval.h**
- EVAL_I2Cx_SDA_PIN : **stm32446e_eval.h**

User Manual by  1.7.6.1

STM32446E_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 -

- FillTriangle() : **stm32446e_eval_lcd.c**
- Filter : **stm32446e_eval_audio.c**
- FMC_BANK1 : **stm32446e_eval.c**
- FMC_BANK1_BASE : **stm32446e_eval.c**
- FMC_BANK1_Init() : **stm32446e_eval.c**
- FMC_BANK1_MspInit() : **stm32446e_eval.c**
- FMC_BANK1_ReadData() : **stm32446e_eval.c**
- FMC_BANK1_WriteData() : **stm32446e_eval.c**
- FMC_BANK1_WriteReg() : **stm32446e_eval.c**
- FMC_BANK3_BASE : **stm32446e_eval.c**

STM32446E_EVAL BSP User Manual

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

- g -

- GetSize() : **stm32446e_eval_camera.c**
- GPIO_PIN : **stm32446e_eval.c**

STM32446E_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_DCMI_ErrorCallback() : **stm32446e_eval_camera.c**
- HAL_DCMI_FrameEventCallback() : **stm32446e_eval_camera.c**
- HAL_DCMI_LineEventCallback() : **stm32446e_eval_camera.c**
- HAL_DCMI_VsyncEventCallback() : **stm32446e_eval_camera.c**
- HAL_I2S_ErrorCallback() : **stm32446e_eval_audio.c**
- HAL_I2S_RxCpltCallback() : **stm32446e_eval_audio.c**
- HAL_I2S_RxHalfCpltCallback() : **stm32446e_eval_audio.c**
- HAL_SAI_ErrorCallback() : **stm32446e_eval_audio.c**
- HAL_SAI_TxCpltCallback() : **stm32446e_eval_audio.c**
- HAL_SAI_TxHalfCpltCallback() : **stm32446e_eval_audio.c**
- haudio_in_i2s : **stm32446e_eval_audio.c**
- haudio_out_sai : **stm32446e_eval_audio.c**
- haudio_tim : **stm32446e_eval_audio.c**
- hDcmiEval : **stm32446e_eval_camera.c**
- hEvalI2c : **stm32446e_eval.c**

STM32446E_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() : **stm32446e_eval.c**
- I2Cx_Init() : **stm32446e_eval.c**
- I2Cx_IsDeviceReady() : **stm32446e_eval.c**
- I2Cx_MspInit() : **stm32446e_eval.c**
- I2Cx_ReadMultiple() : **stm32446e_eval.c**
- I2Cx_WriteMultiple() : **stm32446e_eval.c**
- I2Sx_DeInit() : **stm32446e_eval_audio.c**
- I2Sx_Init() : **stm32446e_eval_audio.c**
- INTERNAL_BUFF_SIZE : **stm32446e_eval_audio.h**
- IO_ERROR : **stm32446e_eval_io.h**
- IO_I2C_ADDRESS : **stm32446e_eval.h**
- IO_I2C_ADDRESS_2 : **stm32446e_eval.h**
- IO_OK : **stm32446e_eval_io.h**
- IO_PIN_0 : **stm32446e_eval_io.h**
- IO_PIN_1 : **stm32446e_eval_io.h**
- IO_PIN_10 : **stm32446e_eval_io.h**
- IO_PIN_11 : **stm32446e_eval_io.h**
- IO_PIN_12 : **stm32446e_eval_io.h**
- IO_PIN_13 : **stm32446e_eval_io.h**

- IO_PIN_14 : **stm32446e_eval_io.h**
- IO_PIN_15 : **stm32446e_eval_io.h**
- IO_PIN_16 : **stm32446e_eval_io.h**
- IO_PIN_17 : **stm32446e_eval_io.h**
- IO_PIN_18 : **stm32446e_eval_io.h**
- IO_PIN_19 : **stm32446e_eval_io.h**
- IO_PIN_2 : **stm32446e_eval_io.h**
- IO_PIN_20 : **stm32446e_eval_io.h**
- IO_PIN_21 : **stm32446e_eval_io.h**
- IO_PIN_22 : **stm32446e_eval_io.h**
- IO_PIN_23 : **stm32446e_eval_io.h**
- IO_PIN_3 : **stm32446e_eval_io.h**
- IO_PIN_4 : **stm32446e_eval_io.h**
- IO_PIN_5 : **stm32446e_eval_io.h**
- IO_PIN_6 : **stm32446e_eval_io.h**
- IO_PIN_7 : **stm32446e_eval_io.h**
- IO_PIN_8 : **stm32446e_eval_io.h**
- IO_PIN_9 : **stm32446e_eval_io.h**
- IO_PIN_ALL : **stm32446e_eval_io.h**
- IO_StatusTypeDef : **stm32446e_eval_io.h**
- IO_TIMEOUT : **stm32446e_eval_io.h**
- IoDrv : **stm32446e_eval_io.c**

STM32446E_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 : **stm32446e_eval.h**
- KEY_BUTTON_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- KEY_BUTTON_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- KEY_BUTTON_GPIO_PORT : **stm32446e_eval.h**
- KEY_BUTTON_PIN : **stm32446e_eval.h**

STM32446E_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 : **stm32446e_eval_lcd.h**
- LCD_COLOR_BLUE : **stm32446e_eval_lcd.h**
- LCD_COLOR_BROWN : **stm32446e_eval_lcd.h**
- LCD_COLOR_CYAN : **stm32446e_eval_lcd.h**
- LCD_COLOR_DARKBLUE : **stm32446e_eval_lcd.h**
- LCD_COLOR_DARKCYAN : **stm32446e_eval_lcd.h**
- LCD_COLOR_DARKGRAY : **stm32446e_eval_lcd.h**
- LCD_COLOR_DARKGREEN : **stm32446e_eval_lcd.h**
- LCD_COLOR_DARKMAGENTA : **stm32446e_eval_lcd.h**
- LCD_COLOR_DARKRED : **stm32446e_eval_lcd.h**
- LCD_COLOR_DARKYELLOW : **stm32446e_eval_lcd.h**
- LCD_COLOR_GRAY : **stm32446e_eval_lcd.h**
- LCD_COLOR_GREEN : **stm32446e_eval_lcd.h**
- LCD_COLOR_LIGHTBLUE : **stm32446e_eval_lcd.h**
- LCD_COLOR_LIGHTGRAY : **stm32446e_eval_lcd.h**
- LCD_COLOR_LIGHTGREEN : **stm32446e_eval_lcd.h**
- LCD_COLOR_LIGHTMAGENTA : **stm32446e_eval_lcd.h**
- LCD_COLOR_LIGHTRED : **stm32446e_eval_lcd.h**
- LCD_COLOR_LIGHTYELLOW : **stm32446e_eval_lcd.h**

- LCD_COLOR_MAGENTA : **stm32446e_eval_lcd.h**
- LCD_COLOR_ORANGE : **stm32446e_eval_lcd.h**
- LCD_COLOR_RED : **stm32446e_eval_lcd.h**
- LCD_COLOR_WHITE : **stm32446e_eval_lcd.h**
- LCD_COLOR_YELLOW : **stm32446e_eval_lcd.h**
- LCD_DEFAULT_FONT : **stm32446e_eval_lcd.h**
- LCD_ERROR : **stm32446e_eval_lcd.h**
- LCD_IO_Init() : **stm32446e_eval.c**
- LCD_IO_ReadData() : **stm32446e_eval.c**
- LCD_IO_WriteData() : **stm32446e_eval.c**
- LCD_IO_WriteReg() : **stm32446e_eval.c**
- LCD_OK : **stm32446e_eval_lcd.h**
- LCD_TIMEOUT : **stm32446e_eval_lcd.h**
- LcdDrv : **stm32446e_eval_lcd.c**
- LED1 : **stm32446e_eval.h**
- LED1_PIN : **stm32446e_eval.h**
- LED3 : **stm32446e_eval.h**
- LED3_PIN : **stm32446e_eval.h**
- LED_GREEN : **stm32446e_eval.h**
- LED_RED : **stm32446e_eval.h**
- Led_TypeDef : **stm32446e_eval.h**
- LEDn : **stm32446e_eval.h**
- LEDx_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- LEDx_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- LEDx_GPIO_PORT : **stm32446e_eval.h**
- LEFT_MODE : **stm32446e_eval_lcd.h**
- Line_ModeTypdef : **stm32446e_eval_lcd.h**

STM32446E_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 -

- MFX_IRQOUT_EXTI_IRQn : **stm32446e_eval.h**
- MFX_IRQOUT_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- MFX_IRQOUT_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- MFX_IRQOUT_GPIO_PORT : **stm32446e_eval.h**
- MFX_IRQOUT_PIN : **stm32446e_eval.h**
- mfxstm32l152Identifier : **stm32446e_eval_io.c**
- MSD_ERROR : **stm32446e_eval_sd.h**
- MSD_ERROR_SD_NOT_PRESENT : **stm32446e_eval_sd.h**
- MSD_OK : **stm32446e_eval_sd.h**

STM32446E_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 -

- OUTPUT_DEVICE_HEADPHONE1 : **stm32446e_eval_audio.h**
- OUTPUT_DEVICE_HEADPHONE2 : **stm32446e_eval_audio.h**

STM32446E_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 -

- PCM_OUT_SIZE : **stm32446e_eval_audio.h**
- PDMDecoder_Init() : **stm32446e_eval_audio.c**
- POLY_X : **stm32446e_eval_lcd.c**
- POLY_Y : **stm32446e_eval_lcd.c**
- pPoint : **stm32446e_eval_lcd.h**

STM32446E_EVAL BSP User Manual

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

- q -

- QSPI_AutoPollingMemReady() : **stm32446e_eval_qspi.c**
- QSPI_BUSY : **stm32446e_eval_qspi.h**
- QSPI_CLK_DISABLE : **stm32446e_eval_qspi.h**
- QSPI_CLK_ENABLE : **stm32446e_eval_qspi.h**
- QSPI_CLK_GPIO_CLK_DISABLE : **stm32446e_eval_qspi.h**
- QSPI_CLK_GPIO_CLK_ENABLE : **stm32446e_eval_qspi.h**
- QSPI_CLK_GPIO_PORT : **stm32446e_eval_qspi.h**
- QSPI_CLK_PIN : **stm32446e_eval_qspi.h**
- QSPI_CS_GPIO_CLK_DISABLE : **stm32446e_eval_qspi.h**
- QSPI_CS_GPIO_CLK_ENABLE : **stm32446e_eval_qspi.h**
- QSPI_CS_GPIO_PORT : **stm32446e_eval_qspi.h**
- QSPI_CS_PIN : **stm32446e_eval_qspi.h**
- QSPI_D0_GPIO_PORT : **stm32446e_eval_qspi.h**
- QSPI_D0_PIN : **stm32446e_eval_qspi.h**
- QSPI_D1_GPIO_PORT : **stm32446e_eval_qspi.h**
- QSPI_D1_PIN : **stm32446e_eval_qspi.h**
- QSPI_D2_GPIO_PORT : **stm32446e_eval_qspi.h**
- QSPI_D2_PIN : **stm32446e_eval_qspi.h**
- QSPI_D3_GPIO_PORT : **stm32446e_eval_qspi.h**

- QSPI_D3_PIN : **stm32446e_eval_qspi.h**
- QSPI_DummyCyclesCfg() : **stm32446e_eval_qspi.c**
- QSPI_Dx_GPIO_CLK_DISABLE : **stm32446e_eval_qspi.h**
- QSPI_Dx_GPIO_CLK_ENABLE : **stm32446e_eval_qspi.h**
- QSPI_EnterFourBytesAddress() : **stm32446e_eval_qspi.c**
- QSPI_ERROR : **stm32446e_eval_qspi.h**
- QSPI_FORCE_RESET : **stm32446e_eval_qspi.h**
- QSPI_NOT_SUPPORTED : **stm32446e_eval_qspi.h**
- QSPI_OK : **stm32446e_eval_qspi.h**
- QSPI_RELEASE_RESET : **stm32446e_eval_qspi.h**
- QSPI_ResetMemory() : **stm32446e_eval_qspi.c**
- QSPI_SUSPENDED : **stm32446e_eval_qspi.h**
- QSPI_WriteEnable() : **stm32446e_eval_qspi.c**
- QSPIHandle : **stm32446e_eval_qspi.c**

STM32446E_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 -

- REFRESH_COUNT : **stm32446e_eval_sdram.h**
- RESOLUTION_R160x120 : **stm32446e_eval_camera.h**
- RESOLUTION_R320x240 : **stm32446e_eval_camera.h**
- RESOLUTION_R480x272 : **stm32446e_eval_camera.h**
- RESOLUTION_R640x480 : **stm32446e_eval_camera.h**
- RIGHT_MODE : **stm32446e_eval_lcd.h**

STM32446E_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 -

- SAIx_DeInit() : **stm32446e_eval_audio.c**
- SAIx_Init() : **stm32446e_eval_audio.c**
- SD_CardInfo : **stm32446e_eval_sd.h**
- SD_DATATIMEOUT : **stm32446e_eval_sd.h**
- SD_DetectIRQHandler : **stm32446e_eval_sd.h**
- SD_DMAx_Rx_CHANNEL : **stm32446e_eval_sd.h**
- SD_DMAx_Rx_IRQHandler : **stm32446e_eval_sd.h**
- SD_DMAx_Rx_IRQn : **stm32446e_eval_sd.h**
- SD_DMAx_Rx_STREAM : **stm32446e_eval_sd.h**
- SD_DMAx_Tx_CHANNEL : **stm32446e_eval_sd.h**
- SD_DMAx_Tx_IRQHandler : **stm32446e_eval_sd.h**
- SD_DMAx_Tx_IRQn : **stm32446e_eval_sd.h**
- SD_DMAx_Tx_STREAM : **stm32446e_eval_sd.h**
- SD_NOT_PRESENT : **stm32446e_eval_sd.h**
- SD_PRESENT : **stm32446e_eval_sd.h**
- SDCLOCK_PERIOD : **stm32446e_eval_sdram.h**
- SDRAM_DEVICE_ADDR : **stm32446e_eval_sdram.h**
- SDRAM_DEVICE_SIZE : **stm32446e_eval_sdram.h**
- SDRAM_DMAx_CHANNEL : **stm32446e_eval_sdram.h**

- SDRAM_DMAx_IRQHandler : **stm32446e_eval_sdram.h**
- SDRAM_DMAx_IRQn : **stm32446e_eval_sdram.h**
- SDRAM_DMAx_STREAM : **stm32446e_eval_sdram.h**
- SDRAM_ERROR : **stm32446e_eval_sdram.h**
- SDRAM_MEMORY_WIDTH : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_BURST_LENGTH_1 : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_BURST_LENGTH_2 : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_BURST_LENGTH_4 : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_BURST_LENGTH_8 : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_BURST_TYPE_INTERLEAVED : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_CAS_LATENCY_2 : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_CAS_LATENCY_3 : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_OPERATING_MODE_STANDARD : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_WRITEBURST_MODE_PROGRAMMED : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_WRITEBURST_MODE_SINGLE : **stm32446e_eval_sdram.h**
- SDRAM_OK : **stm32446e_eval_sdram.h**
- SDRAM_TIMEOUT : **stm32446e_eval_sdram.h**
- sdramHandle : **stm32446e_eval_sdram.c**
- SetDisplayWindow() : **stm32446e_eval_lcd.c**

STM32446E_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 : **stm32446e_eval.h**
- TAMPER_BUTTON_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- TAMPER_BUTTON_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- TAMPER_BUTTON_GPIO_PORT : **stm32446e_eval.h**
- TAMPER_BUTTON_PIN : **stm32446e_eval.h**
- Timing : **stm32446e_eval_sdram.c**
- TIMx_DeInit() : **stm32446e_eval_audio.c**
- TIMx_IC_MspDeInit() : **stm32446e_eval_audio.c**
- TIMx_IC_MspInit() : **stm32446e_eval_audio.c**
- TIMx_Init() : **stm32446e_eval_audio.c**
- ts_driver : **stm32446e_eval_ts.c**
- TS_ERROR : **stm32446e_eval_ts.h**
- TS_I2C_ADDRESS : **stm32446e_eval.h**
- TS_I2C_ADDRESS_2 : **stm32446e_eval.h**
- TS_OK : **stm32446e_eval_ts.h**
- TS_StatusTypeDef : **stm32446e_eval_ts.h**
- TS_SWAP_NONE : **stm32446e_eval_ts.h**
- TS_SWAP_X : **stm32446e_eval_ts.h**
- TS_SWAP_XY : **stm32446e_eval_ts.h**

- TS_SWAP_Y : **stm32446e_eval_ts.h**
- TS_TIMEOUT : **stm32446e_eval_ts.h**
- tsBundaryX : **stm32446e_eval_ts.c**
- tsBundaryY : **stm32446e_eval_ts.c**
- tsOrientation : **stm32446e_eval_ts.c**

STM32446E_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 -

- uSdCardInfo : **stm32446e_eval_sd.c**
- uSdHandle : **stm32446e_eval_sd.c**
- UseExtiModeDetection : **stm32446e_eval_sd.c**

STM32446E_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 : **stm32446e_eval.h**
- WAKEUP_BUTTON_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- WAKEUP_BUTTON_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- WAKEUP_BUTTON_GPIO_PORT : **stm32446e_eval.h**
- WAKEUP_BUTTON_PIN : **stm32446e_eval.h**

STM32446E_EVAL BSP User Manual

- a -

- AUDIO_IO_DeInit() : **stm32446e_eval.c**
- AUDIO_IO_Delay() : **stm32446e_eval.c**
- AUDIO_IO_Init() : **stm32446e_eval.c**
- AUDIO_IO_Read() : **stm32446e_eval.c**
- AUDIO_IO_Write() : **stm32446e_eval.c**

STM32446E_EVAL BSP User Manual

- b -

- BSP_AUDIO_IN_DeInit() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_IN_Error_Callback() : **stm32446e_eval_audio.h** , **stm32446e_eval_audio.c**
- BSP_AUDIO_IN_HalfTransfer_CallBack() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_IN_Init() : **stm32446e_eval_audio.h** , **stm32446e_eval_audio.c**
- BSP_AUDIO_IN_MspDeInit() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_IN_MspInit() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_IN_Pause() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_IN_PDMToPCM() : **stm32446e_eval_audio.h** , **stm32446e_eval_audio.c**
- BSP_AUDIO_IN_Record() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_IN_Resume() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**

- BSP_AUDIO_IN_SetVolume() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_IN_Stop() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_IN_TransferComplete_CallBack() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_ChangeBuffer() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_ClockConfig() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_DeInit() : **stm32446e_eval_audio.h** , **stm32446e_eval_audio.c**
- BSP_AUDIO_OUT_Error_CallBack() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_HalfTransfer_CallBack() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_Init() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_MspDeInit() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_MspInit() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_Pause() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_Play() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_Resume() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_SetAudioFrameSlot() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_SetFrequency() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_SetMute() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_SetOutputMode() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_SetVolume() : **stm32446e_eval_audio.c** ,

stm32446e_eval_audio.h

- BSP_AUDIO_OUT_Stop() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_AUDIO_OUT_TransferComplete_CallBack() : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**
- BSP_CAMERA_BlackWhiteConfig() : **stm32446e_eval_camera.h** , **stm32446e_eval_camera.c**
- BSP_CAMERA_ColorEffectConfig() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_ContinuousStart() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_ContrastBrightnessConfig() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_DeInit() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_DMA_IRQHandler() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_ErrorCallback() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_FrameEventCallback() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_HwReset() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_Init() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_IRQHandler() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_LineEventCallback() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_MspDeInit() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_MspInit() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_PwrDown() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_Resume() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**

- BSP_CAMERA_SnapshotStart() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_Stop() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_Suspend() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_CAMERA_VsyncEventCallback() : **stm32446e_eval_camera.c** , **stm32446e_eval_camera.h**
- BSP_COM_DeInit() : **stm32446e_eval.c** , **stm32446e_eval.h**
- BSP_COM_Init() : **stm32446e_eval.c** , **stm32446e_eval.h**
- BSP_EEPROM_DeInit() : **stm32446e_eval_eeprom.c** , **stm32446e_eval_eeprom.h**
- BSP_EEPROM_Init() : **stm32446e_eval_eeprom.c** , **stm32446e_eval_eeprom.h**
- BSP_EEPROM_ReadBuffer() : **stm32446e_eval_eeprom.c** , **stm32446e_eval_eeprom.h**
- BSP_EEPROM_TIMEOUT_UserCallback() : **stm32446e_eval_eeprom.c** , **stm32446e_eval_eeprom.h**
- BSP_EEPROM_WaitEepromStandbyState() : **stm32446e_eval_eeprom.c** , **stm32446e_eval_eeprom.h**
- BSP_EEPROM_WriteBuffer() : **stm32446e_eval_eeprom.c** , **stm32446e_eval_eeprom.h**
- BSP_EEPROM_WritePage() : **stm32446e_eval_eeprom.c** , **stm32446e_eval_eeprom.h**
- BSP_GetVersion() : **stm32446e_eval.c** , **stm32446e_eval.h**
- BSP_IO_ConfigIrqOutPin() : **stm32446e_eval_io.c** , **stm32446e_eval_io.h**
- BSP_IO_ConfigPin() : **stm32446e_eval_io.c** , **stm32446e_eval_io.h**
- BSP_IO_DeInit() : **stm32446e_eval_io.h** , **stm32446e_eval_io.c**
- BSP_IO_Init() : **stm32446e_eval_io.c** , **stm32446e_eval_io.h**
- BSP_IO_ITClear() : **stm32446e_eval_io.c** , **stm32446e_eval_io.h**
- BSP_IO_ITGetStatus() : **stm32446e_eval_io.c** , **stm32446e_eval_io.h**
- BSP_IO_ReadPin() : **stm32446e_eval_io.c** , **stm32446e_eval_io.h**
- BSP_IO_TogglePin() : **stm32446e_eval_io.c** ,

stm32446e_eval_io.h

- BSP_IO_WritePin() : **stm32446e_eval_io.c** , **stm32446e_eval_io.h**
- BSP_LCD_Clear() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_ClearStringLine() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DeInit() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DisplayChar() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DisplayOff() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DisplayOn() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DisplayStringAt() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DisplayStringAtLine() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DrawBitmap() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DrawCircle() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DrawEllipse() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DrawHLine() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DrawLine() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DrawPixel() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DrawPolygon() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DrawRect() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_DrawRGBImage() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**

- BSP_LCD_DrawVLine() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_FillCircle() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_FillEllipse() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_FillPolygon() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_FillRect() : **stm32446e_eval_lcd.h** , **stm32446e_eval_lcd.c**
- BSP_LCD_GetBackColor() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_GetFont() : **stm32446e_eval_lcd.h** , **stm32446e_eval_lcd.c**
- BSP_LCD_GetTextColor() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_GetXSize() : **stm32446e_eval_lcd.h** , **stm32446e_eval_lcd.c**
- BSP_LCD_GetYSize() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_Init() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_ReadPixel() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_SetBackColor() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_SetFont() : **stm32446e_eval_lcd.c** , **stm32446e_eval_lcd.h**
- BSP_LCD_SetTextColor() : **stm32446e_eval_lcd.h** , **stm32446e_eval_lcd.c**
- BSP_LED_DeInit() : **stm32446e_eval.c** , **stm32446e_eval.h**
- BSP_LED_Init() : **stm32446e_eval.h** , **stm32446e_eval.c**
- BSP_LED_Off() : **stm32446e_eval.h** , **stm32446e_eval.c**
- BSP_LED_On() : **stm32446e_eval.h** , **stm32446e_eval.c**
- BSP_LED_Toggle() : **stm32446e_eval.h** , **stm32446e_eval.c**
- BSP_PB_DeInit() : **stm32446e_eval.c** , **stm32446e_eval.h**
- BSP_PB_GetState() : **stm32446e_eval.c** , **stm32446e_eval.h**

- BSP_PB_Init() : **stm32446e_eval.h** , **stm32446e_eval.c**
- BSP_QSPI_DeInit() : **stm32446e_eval_qspi.c**
- BSP_QSPI_Erase_Block() : **stm32446e_eval_qspi.c**
- BSP_QSPI_Erase_Chip() : **stm32446e_eval_qspi.c**
- BSP_QSPI_GetInfo() : **stm32446e_eval_qspi.c**
- BSP_QSPI_GetStatus() : **stm32446e_eval_qspi.c**
- BSP_QSPI_Init() : **stm32446e_eval_qspi.c**
- BSP_QSPI_MemoryMappedMode() : **stm32446e_eval_qspi.c**
- BSP_QSPI_MspDeInit() : **stm32446e_eval_qspi.h** , **stm32446e_eval_qspi.c**
- BSP_QSPI_MspInit() : **stm32446e_eval_qspi.h** , **stm32446e_eval_qspi.c**
- BSP_QSPI_Read() : **stm32446e_eval_qspi.c**
- BSP_QSPI_Write() : **stm32446e_eval_qspi.c**
- BSP_SD_DeInit() : **stm32446e_eval_sd.h** , **stm32446e_eval_sd.c**
- BSP_SD_DetectCallback() : **stm32446e_eval_sd.h**
- BSP_SD_DetectIT() : **stm32446e_eval_sd.h**
- BSP_SD_DMA_Rx_IRQHandler() : **stm32446e_eval_sd.c** , **stm32446e_eval_sd.h**
- BSP_SD_DMA_Tx_IRQHandler() : **stm32446e_eval_sd.h** , **stm32446e_eval_sd.c**
- BSP_SD_Erase() : **stm32446e_eval_sd.h** , **stm32446e_eval_sd.c**
- BSP_SD_GetCardInfo() : **stm32446e_eval_sd.h** , **stm32446e_eval_sd.c**
- BSP_SD_GetStatus() : **stm32446e_eval_sd.c** , **stm32446e_eval_sd.h**
- BSP_SD_Init() : **stm32446e_eval_sd.c** , **stm32446e_eval_sd.h**
- BSP_SD_IRQHandler() : **stm32446e_eval_sd.h** , **stm32446e_eval_sd.c**
- BSP_SD_IsDetected() : **stm32446e_eval_sd.h** , **stm32446e_eval_sd.c**
- BSP_SD_ITConfig() : **stm32446e_eval_sd.c** , **stm32446e_eval_sd.h**
- BSP_SD_MspDeInit() : **stm32446e_eval_sd.c** , **stm32446e_eval_sd.h**

- BSP_SD_MspInit() : **stm32446e_eval_sd.c** , **stm32446e_eval_sd.h**
- BSP_SD_ReadBlocks() : **stm32446e_eval_sd.c** , **stm32446e_eval_sd.h**
- BSP_SD_ReadBlocks_DMA() : **stm32446e_eval_sd.c** , **stm32446e_eval_sd.h**
- BSP_SD_WriteBlocks() : **stm32446e_eval_sd.h** , **stm32446e_eval_sd.c**
- BSP_SD_WriteBlocks_DMA() : **stm32446e_eval_sd.h** , **stm32446e_eval_sd.c**
- BSP_SDRAM_DeInit() : **stm32446e_eval_sdram.h** , **stm32446e_eval_sdram.c**
- BSP_SDRAM_DMA_IRQHandler() : **stm32446e_eval_sdram.c** , **stm32446e_eval_sdram.h**
- BSP_SDRAM_Init() : **stm32446e_eval_sdram.c** , **stm32446e_eval_sdram.h**
- BSP_SDRAM_Initialization_sequence() : **stm32446e_eval_sdram.c** , **stm32446e_eval_sdram.h**
- BSP_SDRAM_MspDeInit() : **stm32446e_eval_sdram.c** , **stm32446e_eval_sdram.h**
- BSP_SDRAM_MspInit() : **stm32446e_eval_sdram.h** , **stm32446e_eval_sdram.c**
- BSP_SDRAM_ReadData() : **stm32446e_eval_sdram.h** , **stm32446e_eval_sdram.c**
- BSP_SDRAM_ReadData_DMA() : **stm32446e_eval_sdram.c** , **stm32446e_eval_sdram.h**
- BSP_SDRAM_Sendcmd() : **stm32446e_eval_sdram.h** , **stm32446e_eval_sdram.c**
- BSP_SDRAM_WriteData() : **stm32446e_eval_sdram.c** , **stm32446e_eval_sdram.h**
- BSP_SDRAM_WriteData_DMA() : **stm32446e_eval_sdram.c** , **stm32446e_eval_sdram.h**
- BSP_TS_DeInit() : **stm32446e_eval_ts.c** , **stm32446e_eval_ts.h**
- BSP_TS_FIFOClear() : **stm32446e_eval_ts.h** , **stm32446e_eval_ts.c**
- BSP_TS_GetState() : **stm32446e_eval_ts.h** , **stm32446e_eval_ts.c**

- BSP_TS_Init() : **stm32446e_eval_ts.h** , **stm32446e_eval_ts.c**
- BSP_TS_ITClear() : **stm32446e_eval_ts.h** , **stm32446e_eval_ts.c**
- BSP_TS_ITConfig() : **stm32446e_eval_ts.h** , **stm32446e_eval_ts.c**
- BSP_TS_ITDisable() : **stm32446e_eval_ts.c** , **stm32446e_eval_ts.h**
- BSP_TS_ITEnable() : **stm32446e_eval_ts.c** , **stm32446e_eval_ts.h**
- BSP_TS_ITGetStatus() : **stm32446e_eval_ts.c** , **stm32446e_eval_ts.h**

STM32446E_EVAL BSP User Manual

- c -

- CAMERA_Delay() : **stm32446e_eval.c**
- CAMERA_IO_Init() : **stm32446e_eval.c**
- CAMERA_IO_Read() : **stm32446e_eval.c**
- CAMERA_IO_Write() : **stm32446e_eval.c**

STM32446E_EVAL BSP User Manual

- d -

- DrawChar() : **stm32446e_eval_lcd.c**

STM32446E_EVAL BSP User Manual

- e -

- EEPROM_IO_Init() : **stm32446e_eval.c** , **stm32446e_eval_eeprom.h**
- EEPROM_IO_IsDeviceReady() : **stm32446e_eval_eeprom.h** , **stm32446e_eval.c**
- EEPROM_IO_ReadData() : **stm32446e_eval.c** , **stm32446e_eval_eeprom.h**
- EEPROM_IO_WriteData() : **stm32446e_eval_eeprom.h** , **stm32446e_eval.c**

STM32446E_EVAL BSP User Manual

- f -

- FillTriangle() : **stm32446e_eval_lcd.c**
- FMC_BANK1_Init() : **stm32446e_eval.c**
- FMC_BANK1_MspInit() : **stm32446e_eval.c**
- FMC_BANK1_ReadData() : **stm32446e_eval.c**
- FMC_BANK1_WriteData() : **stm32446e_eval.c**
- FMC_BANK1_WriteReg() : **stm32446e_eval.c**

STM32446E_EVAL BSP User Manual

- g -

- GetSize() : **stm32446e_eval_camera.c**

STM32446E_EVAL BSP User Manual

- h -

- HAL_DCMI_ErrorCallback() : **stm32446e_eval_camera.c**
- HAL_DCMI_FrameEventCallback() : **stm32446e_eval_camera.c**
- HAL_DCMI_LineEventCallback() : **stm32446e_eval_camera.c**
- HAL_DCMI_VsyncEventCallback() : **stm32446e_eval_camera.c**
- HAL_I2S_ErrorCallback() : **stm32446e_eval_audio.c**
- HAL_I2S_RxCpltCallback() : **stm32446e_eval_audio.c**
- HAL_I2S_RxHalfCpltCallback() : **stm32446e_eval_audio.c**
- HAL_SAI_ErrorCallback() : **stm32446e_eval_audio.c**
- HAL_SAI_TxCpltCallback() : **stm32446e_eval_audio.c**
- HAL_SAI_TxHalfCpltCallback() : **stm32446e_eval_audio.c**

STM32446E_EVAL BSP User Manual

- i -

- I2Cx_Error() : **stm32446e_eval.c**
- I2Cx_Init() : **stm32446e_eval.c**
- I2Cx_IsDeviceReady() : **stm32446e_eval.c**
- I2Cx_MspInit() : **stm32446e_eval.c**
- I2Cx_ReadMultiple() : **stm32446e_eval.c**
- I2Cx_WriteMultiple() : **stm32446e_eval.c**
- I2Sx_DeInit() : **stm32446e_eval_audio.c**
- I2Sx_Init() : **stm32446e_eval_audio.c**

STM32446E_EVAL BSP User Manual

- l -

- LCD_IO_Init() : **stm32446e_eval.c**
- LCD_IO_ReadData() : **stm32446e_eval.c**
- LCD_IO_WriteData() : **stm32446e_eval.c**
- LCD_IO_WriteReg() : **stm32446e_eval.c**

STM32446E_EVAL BSP User Manual

- p -

- PDMDecoder_Init() : **stm32446e_eval_audio.c**

STM32446E_EVAL BSP User Manual

- q -

- QSPI_AutoPollingMemReady() : **stm32446e_eval_qspi.c**
- QSPI_DummyCyclesCfg() : **stm32446e_eval_qspi.c**
- QSPI_EnterFourBytesAddress() : **stm32446e_eval_qspi.c**
- QSPI_ResetMemory() : **stm32446e_eval_qspi.c**
- QSPI_WriteEnable() : **stm32446e_eval_qspi.c**

STM32446E_EVAL BSP User Manual

- s -

- SAIx_DeInit() : **stm32446e_eval_audio.c**
- SAIx_Init() : **stm32446e_eval_audio.c**
- SetDisplayWindow() : **stm32446e_eval_lcd.c**

STM32446E_EVAL BSP User Manual

- t -

- TIMx_DeInit() : **stm32446e_eval_audio.c**
- TIMx_IC_MspDeInit() : **stm32446e_eval_audio.c**
- TIMx_IC_MspInit() : **stm32446e_eval_audio.c**
- TIMx_Init() : **stm32446e_eval_audio.c**

STM32446E_EVAL BSP User Manual

- a -

- AddressI2C : **stm32446e_eval_ts.c**
- audio_drv : **stm32446e_eval_audio.c**
- AudioInVolume : **stm32446e_eval_audio.c** , **stm32446e_eval_audio.h**

- b -

- BUTTON_IRQn : **stm32446e_eval.c**
- BUTTON_PIN : **stm32446e_eval.c**
- BUTTON_PORT : **stm32446e_eval.c**

- c -

- camera_drv : **stm32446e_eval_camera.c**
- cameraHwAddress : **stm32446e_eval_camera.c**
- Channel_Demux : **stm32446e_eval_audio.c**
- COM_RX_AF : **stm32446e_eval.c**
- COM_RX_PIN : **stm32446e_eval.c**
- COM_RX_PORT : **stm32446e_eval.c**
- COM_TX_AF : **stm32446e_eval.c**

- COM_TX_PIN : **stm32446e_eval.c**
- COM_TX_PORT : **stm32446e_eval.c**
- COM_USART : **stm32446e_eval.c**
- Command : **stm32446e_eval_sdram.c**
- currentResolution : **stm32446e_eval_camera.c**

- d -

- DrawProp : **stm32446e_eval_lcd.c**

- e -

- EEPROMAddress : **stm32446e_eval_eeprom.c**
- EEPROMDataRead : **stm32446e_eval_eeprom.c**
- EEPROMDataWrite : **stm32446e_eval_eeprom.c**

- f -

- Filter : **stm32446e_eval_audio.c**

- g -

- GPIO_PIN : **stm32446e_eval.c**

- h -

- haudio_in_i2s : **stm32446e_eval_audio.c**
- haudio_out_sai : **stm32446e_eval_audio.c**
- haudio_tim : **stm32446e_eval_audio.c**
- hDcmiEval : **stm32446e_eval_camera.c**
- hEvalI2c : **stm32446e_eval.c**

- i -

- IoDrv : **stm32446e_eval_io.c**

- l -

- LcdDrv : **stm32446e_eval_lcd.c**

- m -

- mfxstm32l152Identifier : **stm32446e_eval_io.c**

- q -

- QSPIHandle : **stm32446e_eval_qspi.c**

- s -

- sdramHandle : **stm32446e_eval_sdram.c**

- t -

- Timing : **stm32446e_eval_sdram.c**
- ts_driver : **stm32446e_eval_ts.c**
- tsBundaryX : **stm32446e_eval_ts.c**
- tsBundaryY : **stm32446e_eval_ts.c**
- tsOrientation : **stm32446e_eval_ts.c**

- u -

- uSdCardInfo : **stm32446e_eval_sd.c**
- uSdHandle : **stm32446e_eval_sd.c**
- UseExtiModeDetection : **stm32446e_eval_sd.c**

STM32446E_EVAL BSP User Manual

- pPoint : **stm32446e_eval_lcd.h**

Generated on Fri Jan 15 2016 10:06:23 for STM32446E_EVAL BSP User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

- BSP_IO_PinStateTypeDef : **stm32446e_eval_io.h**
- Button_TypeDef : **stm32446e_eval.h**
- ButtonMode_TypeDef : **stm32446e_eval.h**
- Camera_StatusTypeDef : **stm32446e_eval_camera.h**
- COM_TypeDef : **stm32446e_eval.h**
- IO_StatusTypeDef : **stm32446e_eval_io.h**
- Led_TypeDef : **stm32446e_eval.h**
- Line_ModeTypdef : **stm32446e_eval_lcd.h**
- TS_StatusTypeDef : **stm32446e_eval_ts.h**

STM32446E_EVAL BSP User Manual

- BSP_IO_PIN_RESET : **stm32446e_eval_io.h**
- BSP_IO_PIN_SET : **stm32446e_eval_io.h**
- BUTTON_KEY : **stm32446e_eval.h**
- BUTTON_MODE_EXTI : **stm32446e_eval.h**
- BUTTON_MODE_GPIO : **stm32446e_eval.h**
- BUTTON_TAMPER : **stm32446e_eval.h**
- BUTTON_WAKEUP : **stm32446e_eval.h**
- CAMERA_ERROR : **stm32446e_eval_camera.h**
- CAMERA_OK : **stm32446e_eval_camera.h**
- CAMERA_TIMEOUT : **stm32446e_eval_camera.h**
- CENTER_MODE : **stm32446e_eval_lcd.h**
- COM1 : **stm32446e_eval.h**
- COM2 : **stm32446e_eval.h**
- IO_ERROR : **stm32446e_eval_io.h**
- IO_OK : **stm32446e_eval_io.h**
- IO_TIMEOUT : **stm32446e_eval_io.h**
- LED1 : **stm32446e_eval.h**
- LED3 : **stm32446e_eval.h**
- LED_GREEN : **stm32446e_eval.h**
- LED_RED : **stm32446e_eval.h**
- LEFT_MODE : **stm32446e_eval_lcd.h**
- RIGHT_MODE : **stm32446e_eval_lcd.h**
- TS_ERROR : **stm32446e_eval_ts.h**

- TS_OK : **stm32446e_eval_ts.h**
- TS_TIMEOUT : **stm32446e_eval_ts.h**

STM32446E_EVAL BSP User Manual

- _ -

- __DMAx_CLK_DISABLE : **stm32446e_eval_sdram.h**
- __DMAx_CLK_ENABLE : **stm32446e_eval_sdram.h**
- __DMAx_TxRx_CLK_ENABLE : **stm32446e_eval_sd.h**
- __STM32446E_EVAL_BSP_VERSION : **stm32446e_eval.c**
- __STM32446E_EVAL_BSP_VERSION_MAIN : **stm32446e_eval.c**
- __STM32446E_EVAL_BSP_VERSION_RC : **stm32446e_eval.c**
- __STM32446E_EVAL_BSP_VERSION_SUB1 : **stm32446e_eval.c**
- __STM32446E_EVAL_BSP_VERSION_SUB2 : **stm32446e_eval.c**

STM32446E_EVAL BSP User Manual

- a -

- ABS : **stm32446e_eval_lcd.c**
- AUDIO_ERROR : **stm32446e_eval_audio.h**
- AUDIO_I2C_ADDRESS : **stm32446e_eval.h**
- AUDIO_I2Sx : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_CLK_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_CLK_ENABLE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_DMAx_CHANNEL : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_DMAx_CLK_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_DMAx_CLK_ENABLE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_DMAx_IRQ : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_DMAx_IRQHandler : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_DMAx_MEM_DATA_SIZE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_DMAx_PERIPH_DATA_SIZE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_DMAx_STREAM : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_SCK_AF : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_SCK_GPIO_CLK_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_SCK_GPIO_CLK_ENABLE :

stm32446e_eval_audio.h

- AUDIO_I2Sx_SCK_GPIO_PORT : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_SCK_PIN : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_SD_AF : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_SD_GPIO_CLK_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_SD_GPIO_CLK_ENABLE : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_SD_GPIO_PORT : **stm32446e_eval_audio.h**
- AUDIO_I2Sx_SD_PIN : **stm32446e_eval_audio.h**
- AUDIO_IN_IRQ_PREPRIO : **stm32446e_eval_audio.h**
- AUDIO_OK : **stm32446e_eval_audio.h**
- AUDIO_OUT_IRQ_PREPRIO : **stm32446e_eval_audio.h**
- AUDIO_SAIx : **stm32446e_eval_audio.h**
- AUDIO_SAIx_CLK_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_SAIx_CLK_ENABLE : **stm32446e_eval_audio.h**
- AUDIO_SAIx_DMAx_CHANNEL : **stm32446e_eval_audio.h**
- AUDIO_SAIx_DMAx_CLK_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_SAIx_DMAx_CLK_ENABLE : **stm32446e_eval_audio.h**
- AUDIO_SAIx_DMAx_IRQ : **stm32446e_eval_audio.h**
- AUDIO_SAIx_DMAx_IRQHandler : **stm32446e_eval_audio.h**
- AUDIO_SAIx_DMAx_MEM_DATA_SIZE : **stm32446e_eval_audio.h**
- AUDIO_SAIx_DMAx_PERIPH_DATA_SIZE : **stm32446e_eval_audio.h**
- AUDIO_SAIx_DMAx_STREAM : **stm32446e_eval_audio.h**
- AUDIO_SAIx_FS_PIN : **stm32446e_eval_audio.h**
- AUDIO_SAIx_MCK_PIN : **stm32446e_eval_audio.h**
- AUDIO_SAIx_MCLK_SCK_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_SAIx_MCLK_SCK_ENABLE : **stm32446e_eval_audio.h**
- AUDIO_SAIx_MCLK_SCK_GPIO_PORT : **stm32446e_eval_audio.h**
- AUDIO_SAIx_MCLK_SD_FS_AF : **stm32446e_eval_audio.h**
- AUDIO_SAIx_SCK_AF : **stm32446e_eval_audio.h**
- AUDIO_SAIx_SCK_PIN : **stm32446e_eval_audio.h**
- AUDIO_SAIx_SD_FS_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_SAIx_SD_FS_ENABLE : **stm32446e_eval_audio.h**

- AUDIO_SAIx_SD_FS_GPIO_PORT : **stm32446e_eval_audio.h**
- AUDIO_SAIx_SD_PIN : **stm32446e_eval_audio.h**
- AUDIO_TIMEOUT : **stm32446e_eval_audio.h**
- AUDIO_TIMx : **stm32446e_eval_audio.h**
- AUDIO_TIMx_AF : **stm32446e_eval_audio.h**
- AUDIO_TIMx_CLK_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_TIMx_CLK_ENABLE : **stm32446e_eval_audio.h**
- AUDIO_TIMx_GPIO_CLK_DISABLE : **stm32446e_eval_audio.h**
- AUDIO_TIMx_GPIO_CLK_ENABLE : **stm32446e_eval_audio.h**
- AUDIO_TIMx_GPIO_PORT : **stm32446e_eval_audio.h**
- AUDIO_TIMx_IN_CHANNEL : **stm32446e_eval_audio.h**
- AUDIO_TIMx_IN_GPIO_PIN : **stm32446e_eval_audio.h**
- AUDIO_TIMx_OUT_CHANNEL : **stm32446e_eval_audio.h**
- AUDIO_TIMx_OUT_GPIO_PIN : **stm32446e_eval_audio.h**
- AUDIODATA_SIZE : **stm32446e_eval_audio.h**

STM32446E_EVAL BSP User Manual

- b -

- BUTTONn : **stm32446e_eval.h**
- BUTTONx_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- BUTTONx_GPIO_CLK_ENABLE : **stm32446e_eval.h**

STM32446E_EVAL BSP User Manual

- c -

- CAMERA_I2C_ADDRESS : **stm32446e_eval.h**
- CHANNEL_DEMUX_MASK : **stm32446e_eval_audio.h**
- CODEC_AUDIOFRAME_SLOT_0123 : **stm32446e_eval_audio.h**
- CODEC_AUDIOFRAME_SLOT_02 : **stm32446e_eval_audio.h**
- CODEC_AUDIOFRAME_SLOT_13 : **stm32446e_eval_audio.h**
- CODEC_RESET_DELAY : **stm32446e_eval_audio.h**
- COMn : **stm32446e_eval.h**

STM32446E_EVAL BSP User Manual

- d -

- DEFAULT_AUDIO_IN_BIT_RESOLUTION : **stm32446e_eval_audio.h**
- DEFAULT_AUDIO_IN_CHANNEL_NBR : **stm32446e_eval_audio.h**
- DEFAULT_AUDIO_IN_FREQ : **stm32446e_eval_audio.h**
- DEFAULT_AUDIO_IN_VOLUME : **stm32446e_eval_audio.h**
- DMA_MAX : **stm32446e_eval_audio.h**
- DMA_MAX_SZE : **stm32446e_eval_audio.h**

STM32446E_EVAL BSP User Manual

- e -

- EEPROM_FAIL : **stm32446e_eval_eeprom.h**
- EEPROM_I2C_ADDRESS_A01 : **stm32446e_eval.h**
- EEPROM_I2C_ADDRESS_A02 : **stm32446e_eval.h**
- EEPROM_MAX_SIZE : **stm32446e_eval_eeprom.h**
- EEPROM_MAX_TRIALS : **stm32446e_eval_eeprom.h**
- EEPROM_OK : **stm32446e_eval_eeprom.h**
- EEPROM_PAGESIZE : **stm32446e_eval_eeprom.h**
- EEPROM_TIMEOUT : **stm32446e_eval_eeprom.h**
- EVAL_COM1 : **stm32446e_eval.h**
- EVAL_COM1_CLK_DISABLE : **stm32446e_eval.h**
- EVAL_COM1_CLK_ENABLE : **stm32446e_eval.h**
- EVAL_COM1_IRQn : **stm32446e_eval.h**
- EVAL_COM1_RX_AF : **stm32446e_eval.h**
- EVAL_COM1_RX_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- EVAL_COM1_RX_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- EVAL_COM1_RX_GPIO_PORT : **stm32446e_eval.h**
- EVAL_COM1_RX_PIN : **stm32446e_eval.h**
- EVAL_COM1_TX_AF : **stm32446e_eval.h**
- EVAL_COM1_TX_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- EVAL_COM1_TX_GPIO_CLK_ENABLE : **stm32446e_eval.h**

- EVAL_COM1_TX_GPIO_PORT : **stm32446e_eval.h**
- EVAL_COM1_TX_PIN : **stm32446e_eval.h**
- EVAL_COMx_CLK_DISABLE : **stm32446e_eval.h**
- EVAL_COMx_CLK_ENABLE : **stm32446e_eval.h**
- EVAL_COMx_RX_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- EVAL_COMx_RX_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- EVAL_COMx_TX_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- EVAL_COMx_TX_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- EVAL_DMAx_CLK_ENABLE : **stm32446e_eval.h**
- EVAL_I2Cx : **stm32446e_eval.h**
- EVAL_I2Cx_CLK_DISABLE : **stm32446e_eval.h**
- EVAL_I2Cx_CLK_ENABLE : **stm32446e_eval.h**
- EVAL_I2Cx_ER_IRQn : **stm32446e_eval.h**
- EVAL_I2Cx_EV_IRQn : **stm32446e_eval.h**
- EVAL_I2Cx_FORCE_RESET : **stm32446e_eval.h**
- EVAL_I2Cx_RELEASE_RESET : **stm32446e_eval.h**
- EVAL_I2Cx_SCL_PIN : **stm32446e_eval.h**
- EVAL_I2Cx_SCL_SDA_AF : **stm32446e_eval.h**
- EVAL_I2Cx_SCL_SDA_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- EVAL_I2Cx_SCL_SDA_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- EVAL_I2Cx_SCL_SDA_GPIO_PORT : **stm32446e_eval.h**
- EVAL_I2Cx_SDA_PIN : **stm32446e_eval.h**

STM32446E_EVAL BSP User Manual

- f -

- FMC_BANK1 : **stm32446e_eval.c**
- FMC_BANK1_BASE : **stm32446e_eval.c**
- FMC_BANK3_BASE : **stm32446e_eval.c**

STM32446E_EVAL BSP User Manual

- i -

- INTERNAL_BUFF_SIZE : **stm32446e_eval_audio.h**
- IO_I2C_ADDRESS : **stm32446e_eval.h**
- IO_I2C_ADDRESS_2 : **stm32446e_eval.h**
- IO_PIN_0 : **stm32446e_eval_io.h**
- IO_PIN_1 : **stm32446e_eval_io.h**
- IO_PIN_10 : **stm32446e_eval_io.h**
- IO_PIN_11 : **stm32446e_eval_io.h**
- IO_PIN_12 : **stm32446e_eval_io.h**
- IO_PIN_13 : **stm32446e_eval_io.h**
- IO_PIN_14 : **stm32446e_eval_io.h**
- IO_PIN_15 : **stm32446e_eval_io.h**
- IO_PIN_16 : **stm32446e_eval_io.h**
- IO_PIN_17 : **stm32446e_eval_io.h**
- IO_PIN_18 : **stm32446e_eval_io.h**
- IO_PIN_19 : **stm32446e_eval_io.h**
- IO_PIN_2 : **stm32446e_eval_io.h**
- IO_PIN_20 : **stm32446e_eval_io.h**
- IO_PIN_21 : **stm32446e_eval_io.h**
- IO_PIN_22 : **stm32446e_eval_io.h**
- IO_PIN_23 : **stm32446e_eval_io.h**

- IO_PIN_3 : **stm32446e_eval_io.h**
- IO_PIN_4 : **stm32446e_eval_io.h**
- IO_PIN_5 : **stm32446e_eval_io.h**
- IO_PIN_6 : **stm32446e_eval_io.h**
- IO_PIN_7 : **stm32446e_eval_io.h**
- IO_PIN_8 : **stm32446e_eval_io.h**
- IO_PIN_9 : **stm32446e_eval_io.h**
- IO_PIN_ALL : **stm32446e_eval_io.h**

STM32446E_EVAL BSP User Manual

- k -

- KEY_BUTTON_EXTI_IRQn : **stm32446e_eval.h**
- KEY_BUTTON_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- KEY_BUTTON_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- KEY_BUTTON_GPIO_PORT : **stm32446e_eval.h**
- KEY_BUTTON_PIN : **stm32446e_eval.h**

STM32446E_EVAL BSP User Manual

- l -

- LCD_COLOR_BLACK : **stm32446e_eval_lcd.h**
- LCD_COLOR_BLUE : **stm32446e_eval_lcd.h**
- LCD_COLOR_BROWN : **stm32446e_eval_lcd.h**
- LCD_COLOR_CYAN : **stm32446e_eval_lcd.h**
- LCD_COLOR_DARKBLUE : **stm32446e_eval_lcd.h**
- LCD_COLOR_DARKCYAN : **stm32446e_eval_lcd.h**
- LCD_COLOR_DARKGRAY : **stm32446e_eval_lcd.h**
- LCD_COLOR_DARKGREEN : **stm32446e_eval_lcd.h**
- LCD_COLOR_DARKMAGENTA : **stm32446e_eval_lcd.h**
- LCD_COLOR_DARKRED : **stm32446e_eval_lcd.h**
- LCD_COLOR_DARKYELLOW : **stm32446e_eval_lcd.h**
- LCD_COLOR_GRAY : **stm32446e_eval_lcd.h**
- LCD_COLOR_GREEN : **stm32446e_eval_lcd.h**
- LCD_COLOR_LIGHTBLUE : **stm32446e_eval_lcd.h**
- LCD_COLOR_LIGHTGRAY : **stm32446e_eval_lcd.h**
- LCD_COLOR_LIGHTGREEN : **stm32446e_eval_lcd.h**
- LCD_COLOR_LIGHTMAGENTA : **stm32446e_eval_lcd.h**
- LCD_COLOR_LIGHTRED : **stm32446e_eval_lcd.h**
- LCD_COLOR_LIGHTYELLOW : **stm32446e_eval_lcd.h**
- LCD_COLOR_MAGENTA : **stm32446e_eval_lcd.h**

- LCD_COLOR_ORANGE : **stm32446e_eval_lcd.h**
- LCD_COLOR_RED : **stm32446e_eval_lcd.h**
- LCD_COLOR_WHITE : **stm32446e_eval_lcd.h**
- LCD_COLOR_YELLOW : **stm32446e_eval_lcd.h**
- LCD_DEFAULT_FONT : **stm32446e_eval_lcd.h**
- LCD_ERROR : **stm32446e_eval_lcd.h**
- LCD_OK : **stm32446e_eval_lcd.h**
- LCD_TIMEOUT : **stm32446e_eval_lcd.h**
- LED1_PIN : **stm32446e_eval.h**
- LED3_PIN : **stm32446e_eval.h**
- LEDn : **stm32446e_eval.h**
- LEDx_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- LEDx_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- LEDx_GPIO_PORT : **stm32446e_eval.h**

STM32446E_EVAL BSP User Manual

- m -

- MFX_IRQOUT_EXTI_IRQn : **stm32446e_eval.h**
- MFX_IRQOUT_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- MFX_IRQOUT_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- MFX_IRQOUT_GPIO_PORT : **stm32446e_eval.h**
- MFX_IRQOUT_PIN : **stm32446e_eval.h**
- MSD_ERROR : **stm32446e_eval_sd.h**
- MSD_ERROR_SD_NOT_PRESENT : **stm32446e_eval_sd.h**
- MSD_OK : **stm32446e_eval_sd.h**

STM32446E_EVAL BSP User Manual

- o -

- OUTPUT_DEVICE_HEADPHONE1 : **stm32446e_eval_audio.h**
- OUTPUT_DEVICE_HEADPHONE2 : **stm32446e_eval_audio.h**

STM32446E_EVAL BSP User Manual

- p -

- PCM_OUT_SIZE : **stm32446e_eval_audio.h**
- POLY_X : **stm32446e_eval_lcd.c**
- POLY_Y : **stm32446e_eval_lcd.c**

STM32446E_EVAL BSP User Manual

- q -

- QSPI_BUSY : **stm32446e_eval_qspi.h**
- QSPI_CLK_DISABLE : **stm32446e_eval_qspi.h**
- QSPI_CLK_ENABLE : **stm32446e_eval_qspi.h**
- QSPI_CLK_GPIO_CLK_DISABLE : **stm32446e_eval_qspi.h**
- QSPI_CLK_GPIO_CLK_ENABLE : **stm32446e_eval_qspi.h**
- QSPI_CLK_GPIO_PORT : **stm32446e_eval_qspi.h**
- QSPI_CLK_PIN : **stm32446e_eval_qspi.h**
- QSPI_CS_GPIO_CLK_DISABLE : **stm32446e_eval_qspi.h**
- QSPI_CS_GPIO_CLK_ENABLE : **stm32446e_eval_qspi.h**
- QSPI_CS_GPIO_PORT : **stm32446e_eval_qspi.h**
- QSPI_CS_PIN : **stm32446e_eval_qspi.h**
- QSPI_D0_GPIO_PORT : **stm32446e_eval_qspi.h**
- QSPI_D0_PIN : **stm32446e_eval_qspi.h**
- QSPI_D1_GPIO_PORT : **stm32446e_eval_qspi.h**
- QSPI_D1_PIN : **stm32446e_eval_qspi.h**
- QSPI_D2_GPIO_PORT : **stm32446e_eval_qspi.h**
- QSPI_D2_PIN : **stm32446e_eval_qspi.h**
- QSPI_D3_GPIO_PORT : **stm32446e_eval_qspi.h**
- QSPI_D3_PIN : **stm32446e_eval_qspi.h**
- QSPI_Dx_GPIO_CLK_DISABLE : **stm32446e_eval_qspi.h**

- QSPI_Dx_GPIO_CLK_ENABLE : **stm32446e_eval_qspi.h**
- QSPI_ERROR : **stm32446e_eval_qspi.h**
- QSPI_FORCE_RESET : **stm32446e_eval_qspi.h**
- QSPI_NOT_SUPPORTED : **stm32446e_eval_qspi.h**
- QSPI_OK : **stm32446e_eval_qspi.h**
- QSPI_RELEASE_RESET : **stm32446e_eval_qspi.h**
- QSPI_SUSPENDED : **stm32446e_eval_qspi.h**

STM32446E_EVAL BSP User Manual

- r -

- REFRESH_COUNT : **stm32446e_eval_sdram.h**
- RESOLUTION_R160x120 : **stm32446e_eval_camera.h**
- RESOLUTION_R320x240 : **stm32446e_eval_camera.h**
- RESOLUTION_R480x272 : **stm32446e_eval_camera.h**
- RESOLUTION_R640x480 : **stm32446e_eval_camera.h**

STM32446E_EVAL BSP User Manual

- s -

- SD_CardInfo : **stm32446e_eval_sd.h**
- SD_DATATIMEOUT : **stm32446e_eval_sd.h**
- SD_DetectIRQHandler : **stm32446e_eval_sd.h**
- SD_DMAx_Rx_CHANNEL : **stm32446e_eval_sd.h**
- SD_DMAx_Rx_IRQHandler : **stm32446e_eval_sd.h**
- SD_DMAx_Rx_IRQn : **stm32446e_eval_sd.h**
- SD_DMAx_Rx_STREAM : **stm32446e_eval_sd.h**
- SD_DMAx_Tx_CHANNEL : **stm32446e_eval_sd.h**
- SD_DMAx_Tx_IRQHandler : **stm32446e_eval_sd.h**
- SD_DMAx_Tx_IRQn : **stm32446e_eval_sd.h**
- SD_DMAx_Tx_STREAM : **stm32446e_eval_sd.h**
- SD_NOT_PRESENT : **stm32446e_eval_sd.h**
- SD_PRESENT : **stm32446e_eval_sd.h**
- SDCLOCK_PERIOD : **stm32446e_eval_sdram.h**
- SDRAM_DEVICE_ADDR : **stm32446e_eval_sdram.h**
- SDRAM_DEVICE_SIZE : **stm32446e_eval_sdram.h**
- SDRAM_DMAx_CHANNEL : **stm32446e_eval_sdram.h**
- SDRAM_DMAx_IRQHandler : **stm32446e_eval_sdram.h**
- SDRAM_DMAx_IRQn : **stm32446e_eval_sdram.h**
- SDRAM_DMAx_STREAM : **stm32446e_eval_sdram.h**

- SDRAM_ERROR : **stm32446e_eval_sdram.h**
- SDRAM_MEMORY_WIDTH : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_BURST_LENGTH_1 : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_BURST_LENGTH_2 : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_BURST_LENGTH_4 : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_BURST_LENGTH_8 : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_BURST_TYPE_INTERLEAVED : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_CAS_LATENCY_2 : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_CAS_LATENCY_3 : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_OPERATING_MODE_STANDARD : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_WRITEBURST_MODE_PROGRAMMED : **stm32446e_eval_sdram.h**
- SDRAM_MODEREG_WRITEBURST_MODE_SINGLE : **stm32446e_eval_sdram.h**
- SDRAM_OK : **stm32446e_eval_sdram.h**
- SDRAM_TIMEOUT : **stm32446e_eval_sdram.h**

STM32446E_EVAL BSP User Manual

- t -

- TAMPER_BUTTON_EXTI_IRQn : **stm32446e_eval.h**
- TAMPER_BUTTON_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- TAMPER_BUTTON_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- TAMPER_BUTTON_GPIO_PORT : **stm32446e_eval.h**
- TAMPER_BUTTON_PIN : **stm32446e_eval.h**
- TS_I2C_ADDRESS : **stm32446e_eval.h**
- TS_I2C_ADDRESS_2 : **stm32446e_eval.h**
- TS_SWAP_NONE : **stm32446e_eval_ts.h**
- TS_SWAP_X : **stm32446e_eval_ts.h**
- TS_SWAP_XY : **stm32446e_eval_ts.h**
- TS_SWAP_Y : **stm32446e_eval_ts.h**

STM32446E_EVAL BSP User Manual

- w -

- WAKEUP_BUTTON_EXTI_IRQn : **stm32446e_eval.h**
- WAKEUP_BUTTON_GPIO_CLK_DISABLE : **stm32446e_eval.h**
- WAKEUP_BUTTON_GPIO_CLK_ENABLE : **stm32446e_eval.h**
- WAKEUP_BUTTON_GPIO_PORT : **stm32446e_eval.h**
- WAKEUP_BUTTON_PIN : **stm32446e_eval.h**

STM32446E_EVAL BSP User Manual

stm32446e_eval.c File Reference

This file provides a set of firmware functions to manage LEDs, push-buttons and COM ports available on STM32446E-EVAL evaluation board(MB1045) RevB from STMicroelectronics. More...

```
#include "stm32446e_eval.h"
```

Go to the source code of this file.

Data Structures

struct	**LCD_CONTROLLER_TypeDef**

Defines

#define	**__STM32446E_EVAL_BSP_VERSION_MAIN** (0x01) STM32446E EVAL BSP Driver version number V1.1.1.
#define	**__STM32446E_EVAL_BSP_VERSION_SUB1** (0x01)
#define	**__STM32446E_EVAL_BSP_VERSION_SUB2** (0x01)
#define	**__STM32446E_EVAL_BSP_VERSION_RC** (0x00)
#define	**__STM32446E_EVAL_BSP_VERSION**
#define	**FMC_BANK1_BASE** ((uint32_t)(0x60000000 | 0x00000000))
#define	**FMC_BANK3_BASE** ((uint32_t)(0x60000000 | 0x08000000))
#define	**FMC_BANK1** ((**LCD_CONTROLLER_TypeDef** *) **FMC_BANK1_BASE**)

Functions

static void	**I2Cx_MspInit** (void) Initializes I2C MSP.
static void	**I2Cx_Init** (void) Initializes I2C HAL.
static HAL_StatusTypeDef	**I2Cx_ReadMultiple** (uint8_t Addr, uint16_t Reg, uint16_t MemAddSize, uint8_t *Buffer, uint16_t Length) Reads multiple data.
static HAL_StatusTypeDef	**I2Cx_WriteMultiple** (uint8_t Addr, uint16_t Reg, uint16_t MemAddSize, uint8_t *Buffer, uint16_t Length) Writes a value in a register of the device through BUS in using DMA mode.
static HAL_StatusTypeDef	**I2Cx_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.
static void	**I2Cx_Error** (uint8_t Addr) Manages error callback by re-initializing I2C.
static void	**FMC_BANK1_WriteData** (uint16_t Data) Writes register value.
static void	**FMC_BANK1_WriteReg** (uint8_t Reg) Writes register address.
static uint16_t	**FMC_BANK1_ReadData** (void) Reads register value.
static void	**FMC_BANK1_Init** (void) Initializes LCD IO.
static void	**FMC_BANK1_MspInit** (void) Initializes FMC_BANK1 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_WriteReg** (uint8_t Reg) Writes register on LCD register.
uint16_t	**LCD_IO_ReadData** (void) Reads data from LCD data register.
void	**AUDIO_IO_Init** (void) Initializes Audio low level.
void	**AUDIO_IO_DeInit** (void) DeInitializes Audio low level.
void	**AUDIO_IO_Write** (uint8_t Addr, uint16_t Reg, uint16_t Value) Writes a single data.
uint16_t	**AUDIO_IO_Read** (uint8_t Addr, uint16_t Reg) Reads a single data.
void	**AUDIO_IO_Delay** (uint32_t Delay) AUDIO Codec delay.
void	**CAMERA_IO_Init** (void) Initializes Camera low level.
void	**CAMERA_Delay** (uint32_t Delay) Camera delay.
void	**CAMERA_IO_Write** (uint8_t Addr, uint16_t Reg, uint16_t Value) Camera writes single data.
uint16_t	**CAMERA_IO_Read** (uint8_t Addr, uint16_t Reg) Camera reads single data.
void	**EEPROM_IO_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
	EEPROM_IO_WriteData (uint16_t DevAddress, uint16_t MemAddress,

HAL_StatusTypeDef	uint8_t *pBuffer, uint32_t BufferSize) Write data to I2C EEPROM driver in using DMA channel.
HAL_StatusTypeDef	**EEPROM_IO_ReadData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Read data from I2C EEPROM driver in using DMA channel.
HAL_StatusTypeDef	**EEPROM_IO_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.
uint32_t	**BSP_GetVersion** (void) This method returns the STM32446E EVAL BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LEDs.
void	**BSP_LED_DeInit** (**Led_TypeDef** Led) DeInit LEDs.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** ButtonMode) Configures button GPIO and EXTI Line.
void	**BSP_PB_DeInit** (**Button_TypeDef** Button) Push Button DeInit.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button)

	Returns the selected button state.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) Configures COM port.
void	**BSP_COM_DeInit** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) DeInit COM port.

Variables

const uint32_t	**GPIO_PIN** [**LEDn**]
GPIO_TypeDef *	**BUTTON_PORT** [**BUTTONn**]
const uint16_t	**BUTTON_PIN** [**BUTTONn**]
const uint16_t	**BUTTON_IRQn** [**BUTTONn**]
USART_TypeDef *	**COM_USART** [**COMn**] = {**EVAL_COM1**}
GPIO_TypeDef *	**COM_TX_PORT** [**COMn**] = {**EVAL_COM1_TX_GPIO_PORT**}
GPIO_TypeDef *	**COM_RX_PORT** [**COMn**] = {**EVAL_COM1_RX_GPIO_PORT**}
const uint16_t	**COM_TX_PIN** [**COMn**] = {**EVAL_COM1_TX_PIN**}
const uint16_t	**COM_RX_PIN** [**COMn**] = {**EVAL_COM1_RX_PIN**}
const uint16_t	**COM_TX_AF** [**COMn**] = {**EVAL_COM1_TX_AF**}
const uint16_t	**COM_RX_AF** [**COMn**] = {**EVAL_COM1_RX_AF**}
static FMPI2C_HandleTypeDef	**hEvalI2c**

Detailed Description

This file provides a set of firmware functions to manage LEDs, push-buttons and COM ports available on STM32446E-EVAL evaluation board(MB1045) RevB from STMicroelectronics.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

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 **stm32446e_eval.c**.

STM32446E_EVAL BSP User Manual

Defines | Enumerations | Functions

stm32446e_eval.h File Reference

This file contains definitions for STM32446E_EVAL's LEDs, push-buttons and COM ports hardware resources. More...

```
#include "stm32f4xx_hal.h"
```

Go to the source code of this file.

Defines

#define **LEDn** ((uint8_t)2)

#define **LEDx_GPIO_PORT** GPIOB

#define **LEDx_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_CLK_E

#define **LEDx_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_CLK_D

#define **LED1_PIN** GPIO_PIN_11

#define **LED3_PIN** GPIO_PIN_4

#define **BUTTONn** ((uint8_t)3)

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

#define **WAKEUP_BUTTON_GPIO_PORT** GPIOC

#define **WAKEUP_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_G

#define **WAKEUP_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_(

#define **WAKEUP_BUTTON_EXTI_IRQn** EXTI15_10_IRQn

#define **TAMPER_BUTTON_PIN** GPIO_PIN_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_G

#define **TAMPER_BUTTON_EXTI_IRQn** EXTI15_10_IRQn

#define **KEY_BUTTON_PIN** GPIO_PIN_13
Key push-button.

#define **KEY_BUTTON_GPIO_PORT** GPIOC

#define **KEY_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOC

#define **KEY_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_GPIO

#define **KEY_BUTTON_EXTI_IRQn** EXTI15_10_IRQn

#define **MFX_IRQOUT_PIN** GPIO_PIN_0
MFX Irq Out Pin.

#define **MFX_IRQOUT_GPIO_PORT** GPIOA

#define **MFX_IRQOUT_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOA

#define **MFX_IRQOUT_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOA

#define **MFX_IRQOUT_EXTI_IRQn** EXTI0_IRQn

#define	**BUTTONx_GPIO_CLK_ENABLE**(__INDEX__)
#define	**BUTTONx_GPIO_CLK_DISABLE**(__INDEX__)
#define	**COMn** ((uint8_t)1)
#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
#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_TX_AF** GPIO_AF7_USART1
#define	**EVAL_COM1_RX_PIN** GPIO_PIN_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_RX_AF** GPIO_AF7_USART1
#define	**EVAL_COM1_IRQn** USART1_IRQn
#define	**EVAL_COMx_CLK_ENABLE**(__INDEX__) do { if((__INDEX_ {**EVAL_COM1_CLK_ENABLE**();} } while(0)
#define	**EVAL_COMx_CLK_DISABLE**(__INDEX__) (((__INDEX__) = **EVAL_COM1_CLK_DISABLE**() : 0)
#define	**EVAL_COMx_TX_GPIO_CLK_ENABLE**(__INDEX__) do { if({**EVAL_COM1_TX_GPIO_CLK_ENABLE**();} } while(0)
#define	**EVAL_COMx_TX_GPIO_CLK_DISABLE**(__INDEX__) (((__I **EVAL_COM1_TX_GPIO_CLK_DISABLE**() : 0)
#define	**EVAL_COMx_RX_GPIO_CLK_ENABLE**(__INDEX__) do { if {**EVAL_COM1_RX_GPIO_CLK_ENABLE**();} } while(0)
#define	**EVAL_COMx_RX_GPIO_CLK_DISABLE**(__INDEX__) (((__I **EVAL_COM1_RX_GPIO_CLK_DISABLE**() : 0)
#define	**IO_I2C_ADDRESS** ((uint16_t)0x84) /*mfx MFX_I2C_ADDR 0 Joystick Pins definition.
#define	**IO_I2C_ADDRESS_2** ((uint16_t)0x86) /*mfx MFX_I2C_ADDR
#define	**TS_I2C_ADDRESS** ((uint16_t)0x84) /*mfx MFX_I2C_ADDR (
#define	**TS_I2C_ADDRESS_2** ((uint16_t)0x86) /*mfx MFX_I2C_ADDI

```
#define CAMERA_I2C_ADDRESS ((uint16_t)0x5A)
#define AUDIO_I2C_ADDRESS ((uint16_t)0x34)
#define EEPROM_I2C_ADDRESS_A01 ((uint16_t)0xA0)
#define EEPROM_I2C_ADDRESS_A02 ((uint16_t)0xA6)
#define EVAL_I2Cx FMPI2C1
#define EVAL_I2Cx_CLK_ENABLE() __HAL_RCC_FMPI2C1_CLK_
#define EVAL_I2Cx_CLK_DISABLE() __HAL_RCC_FMPI2C1_CLK_
#define EVAL_DMAx_CLK_ENABLE() __HAL_RCC_DMA1_CLK_E
#define EVAL_I2Cx_SCL_SDA_GPIO_CLK_ENABLE() __HAL_RCC
#define EVAL_I2Cx_SCL_SDA_GPIO_CLK_DISABLE() __HAL_RC
#define EVAL_I2Cx_FORCE_RESET() __HAL_RCC_FMPI2C1_FOR
#define EVAL_I2Cx_RELEASE_RESET() __HAL_RCC_FMPI2C1_R
#define EVAL_I2Cx_SCL_PIN GPIO_PIN_12
#define EVAL_I2Cx_SCL_SDA_GPIO_PORT GPIOD
#define EVAL_I2Cx_SCL_SDA_AF GPIO_AF4_FMPI2C1
#define EVAL_I2Cx_SDA_PIN GPIO_PIN_13
#define EVAL_I2Cx_EV_IRQn FMPI2C1_EV_IRQn
#define EVAL_I2Cx_ER_IRQn FMPI2C1_ER_IRQn
```

Enumerations

enum	**Led_TypeDef** { **LED1** = 0, **LED_GREEN** = LED1, **LED3** = 1, **LED_RED** = LED3 }
enum	**Button_TypeDef** { **BUTTON_WAKEUP** = 0, **BUTTON_TAMPER** = 1, **BUTTON_KEY** = 2 }
enum	**ButtonMode_TypeDef** { **BUTTON_MODE_GPIO** = 0, **BUTTON_MODE_EXTI** = 1 }
enum	**COM_TypeDef** { **COM1** = 0, **COM2** = 1 }

Functions

uint32_t	**BSP_GetVersion** (void) This method returns the STM32446E EVAL BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LEDs.
void	**BSP_LED_DeInit** (**Led_TypeDef** Led) DeInit LEDs.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** ButtonMode) Configures button GPIO and EXTI Line.
void	**BSP_PB_DeInit** (**Button_TypeDef** Button) Push Button DeInit.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *husart) Configures COM port.
void	**BSP_COM_DeInit** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) DeInit COM port.

Detailed Description

This file contains definitions for STM32446E_EVAL's LEDs, push-buttons and COM ports hardware resources.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

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 **stm32446e_eval.h**.

STM32446E_EVAL BSP User Manual

stm32446e_eval_audio.c File Reference

This file provides the Audio driver for the STM32446E-EVAL evaluation board. More...

```
#include "stm32446e_eval_audio.h"
```

Go to the source code of this file.

Functions

static void	**SAIx_Init** (uint32_t AudioFreq) Initializes the Audio Codec audio interface (SAI).
static void	**SAIx_DeInit** (void) Deinitializes the Audio Codec audio interface (SAI).
static void	**I2Sx_Init** (uint32_t AudioFreq) Initializes the Audio Codec audio interface (I2S)
static void	**I2Sx_DeInit** (void) Deinitializes the Audio Codec audio interface (I2S).
static void	**TIMx_IC_MspInit** (TIM_HandleTypeDef *htim) Initializes the TIM INput Capture MSP.
static void	**TIMx_IC_MspDeInit** (TIM_HandleTypeDef *htim) Initializes the TIM INput Capture MSP.
static void	**TIMx_Init** (void) Configure TIM as a clock divider by 2.
static void	**TIMx_DeInit** (void) Configure TIM as a clock divider by 2.
static void	**PDMDecoder_Init** (uint32_t AudioFreq, uint32_t ChnlNbr) Initializes the PDM library.
uint8_t	**BSP_AUDIO_OUT_Init** (uint16_t OutputDevice, uint8_t Volume, uint32_t AudioFreq) Configures 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 SAI 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) Updates the audio frequency.
void	**BSP_AUDIO_OUT_SetAudioFrameSlot** (uint32_t AudioFrameSlot) Updates the Audio frame slot configuration.
void	**BSP_AUDIO_OUT_DeInit** (void) Deinit the audio peripherals.
void	**HAL_SAI_TxCpltCallback** (SAI_HandleTypeDef *hsai) Tx Transfer completed callbacks.
void	**HAL_SAI_TxHalfCpltCallback** (SAI_HandleTypeDef *hsai) Tx Half Transfer completed callbacks.
void	**HAL_SAI_ErrorCallback** (SAI_HandleTypeDef *hsai) SAI 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.
__weak void	**BSP_AUDIO_OUT_MspInit** (SAI_HandleTypeDef *hsai,

	void *Params) Initializes BSP_AUDIO_OUT MSP.
__weak void	**BSP_AUDIO_OUT_MspDeInit** (SAI_HandleTypeDef *hsai, void *Params) Deinitializes SAI MSP.
__weak void	**BSP_AUDIO_OUT_ClockConfig** (SAI_HandleTypeDef *hsai, uint32_t AudioFreq, void *Params) Clock Config.
uint8_t	**BSP_AUDIO_IN_Init** (uint32_t AudioFreq, uint32_t BitRes, uint32_t ChnlNbr) Initializes wave recording.
uint8_t	**BSP_AUDIO_IN_Record** (uint16_t *pbuf, uint32_t size) Starts audio recording.
uint8_t	**BSP_AUDIO_IN_Stop** (void) Stops audio recording.
uint8_t	**BSP_AUDIO_IN_Pause** (void) Pauses the audio file stream.
uint8_t	**BSP_AUDIO_IN_Resume** (void) Resumes the audio file stream.
uint8_t	**BSP_AUDIO_IN_SetVolume** (uint8_t Volume) Controls the audio in volume level.
void	**BSP_AUDIO_IN_DeInit** (void) Deinit the audio IN peripherals.
uint8_t	**BSP_AUDIO_IN_PDMToPCM** (uint16_t *PDMBuf, uint16_t *PCMBuf) Converts audio format from PDM to PCM.
void	**HAL_I2S_RxCpltCallback** (I2S_HandleTypeDef *hi2s) Rx Transfer completed callbacks.
void	**HAL_I2S_RxHalfCpltCallback** (I2S_HandleTypeDef *hi2s) Rx Half Transfer completed callbacks.
void	**HAL_I2S_ErrorCallback** (I2S_HandleTypeDef *hi2s) I2S error callbacks.

__weak void	**BSP_AUDIO_IN_TransferComplete_CallBack** (void) User callback when record buffer is filled.
__weak void	**BSP_AUDIO_IN_HalfTransfer_CallBack** (void) Manages the DMA Half Transfer complete event.
__weak void	**BSP_AUDIO_IN_Error_Callback** (void) Audio IN Error callback function.
__weak void	**BSP_AUDIO_IN_MspInit** (I2S_HandleTypeDef *hi2s, void *Params) Initializes BSP_AUDIO_IN MSP.
__weak void	**BSP_AUDIO_IN_MspDeInit** (I2S_HandleTypeDef *hi2s, void *Params) DeInitializes BSP_AUDIO_IN MSP.

Variables

AUDIO_DrvTypeDef *	**audio_drv**
SAI_HandleTypeDef	**haudio_out_sai**
I2S_HandleTypeDef	**haudio_in_i2s**
TIM_HandleTypeDef	**haudio_tim**
PDMFilter_InitStruct	**Filter** [2]
uint8_t	**Channel_Demux** [128]
uint16_t __IO	**AudioInVolume** = **DEFAULT_AUDIO_IN_VOLUME**

Detailed Description

This file provides the Audio driver for the STM32446E-EVAL evaluation board.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

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 **stm32446e_eval_audio.c**.

STM32446E_EVAL BSP User Manual

stm32446e_eval_audio.h File Reference

This file contains the common defines and functions prototypes for the **stm32446e_eval_audio.c** driver. More...

```
#include "../Components/wm8994/wm8994.h" #include
"stm32446e_eval.h"
#include
"../../../Middlewares/ST/STM32_Audio/Addons/PDM/pdm_filter.h"
```

Go to the source code of this file.

Defines

#define	**CODEC_AUDIOFRAME_SLOT_0123** SAI_SLOTACTIVE_0 | SAI_SLOTACTIVE_3
#define	**CODEC_AUDIOFRAME_SLOT_02** SAI_SLOTACTIVE_0 | SA
#define	**CODEC_AUDIOFRAME_SLOT_13** SAI_SLOTACTIVE_1 | SA
#define	**AUDIO_SAIx** SAI2_Block_B
#define	**AUDIO_SAIx_CLK_ENABLE**() __HAL_RCC_SAI2_CLK_EN
#define	**AUDIO_SAIx_CLK_DISABLE**() __HAL_RCC_SAI2_CLK_DI
#define	**AUDIO_SAIx_SCK_AF** GPIO_AF8_SAI2
#define	**AUDIO_SAIx_MCLK_SD_FS_AF** GPIO_AF10_SAI2
#define	**AUDIO_SAIx_MCLK_SCK_ENABLE**() __HAL_RCC_GPIOA
#define	**AUDIO_SAIx_MCLK_SCK_DISABLE**() __HAL_RCC_GPIOA
#define	**AUDIO_SAIx_MCK_PIN** GPIO_PIN_1 /* PA1 */
#define	**AUDIO_SAIx_SCK_PIN** GPIO_PIN_2 /* PA2 */
#define	**AUDIO_SAIx_MCLK_SCK_GPIO_PORT** GPIOA
#define	**AUDIO_SAIx_SD_FS_ENABLE**() __HAL_RCC_GPIOG_CLK
#define	**AUDIO_SAIx_SD_FS_DISABLE**() __HAL_RCC_GPIOG_CL
#define	**AUDIO_SAIx_FS_PIN** GPIO_PIN_9 /* PG9 */
#define	**AUDIO_SAIx_SD_PIN** GPIO_PIN_10 /* PG10 */
#define	**AUDIO_SAIx_SD_FS_GPIO_PORT** GPIOG
#define	**AUDIO_SAIx_DMAx_CLK_ENABLE**() __HAL_RCC_DMA2_
#define	**AUDIO_SAIx_DMAx_CLK_DISABLE**() __HAL_RCC_DMA2_
#define	**AUDIO_SAIx_DMAx_STREAM** DMA2_Stream6
#define	**AUDIO_SAIx_DMAx_CHANNEL** DMA_CHANNEL_3
#define	**AUDIO_SAIx_DMAx_IRQ** DMA2_Stream6_IRQn
#define	**AUDIO_SAIx_DMAx_PERIPH_DATA_SIZE** DMA_PDATAAL
#define	**AUDIO_SAIx_DMAx_MEM_DATA_SIZE** DMA_MDATAALIGN
#define	**DMA_MAX_SZE** 0xFFFF
#define	**AUDIO_SAIx_DMAx_IRQHandler** DMA2_Stream6_IRQHan
#define	**AUDIO_OUT_IRQ_PREPRIO** ((uint32_t)5) /* Select the preem
#define	**AUDIO_I2Sx** SPI3
#define	**AUDIO_I2Sx_CLK_ENABLE**() __HAL_RCC_SPI3_CLK_EN

```
#define AUDIO_I2Sx_CLK_DISABLE()   __HAL_RCC_SPI3_CLK_DIS
#define AUDIO_I2Sx_SCK_PIN   GPIO_PIN_3
#define AUDIO_I2Sx_SCK_GPIO_PORT   GPIOB
#define AUDIO_I2Sx_SCK_GPIO_CLK_ENABLE()   __HAL_RCC_GP
#define AUDIO_I2Sx_SCK_GPIO_CLK_DISABLE()   __HAL_RCC_G
#define AUDIO_I2Sx_SCK_AF   GPIO_AF6_SPI3
#define AUDIO_I2Sx_SD_PIN   GPIO_PIN_1
#define AUDIO_I2Sx_SD_GPIO_PORT   GPIOC
#define AUDIO_I2Sx_SD_GPIO_CLK_ENABLE()   __HAL_RCC_GPI
#define AUDIO_I2Sx_SD_GPIO_CLK_DISABLE()   __HAL_RCC_GP
#define AUDIO_I2Sx_SD_AF   GPIO_AF5_SPI3
#define AUDIO_I2Sx_DMAx_CLK_ENABLE()   __HAL_RCC_DMA1_
#define AUDIO_I2Sx_DMAx_CLK_DISABLE()   __HAL_RCC_DMA1_
#define AUDIO_I2Sx_DMAx_STREAM   DMA1_Stream2
#define AUDIO_I2Sx_DMAx_CHANNEL   DMA_CHANNEL_0
#define AUDIO_I2Sx_DMAx_IRQ   DMA1_Stream2_IRQn
#define AUDIO_I2Sx_DMAx_PERIPH_DATA_SIZE   DMA_PDATAALI
#define AUDIO_I2Sx_DMAx_MEM_DATA_SIZE   DMA_MDATAALIGN
#define AUDIO_I2Sx_DMAx_IRQHandler   DMA1_Stream2_IRQHand
#define AUDIO_IN_IRQ_PREPRIO   ((uint32_t)6) /* Select the preemp
#define AUDIO_TIMx_CLK_ENABLE()   __HAL_RCC_TIM4_CLK_EN
#define AUDIO_TIMx_CLK_DISABLE()   __HAL_RCC_TIM4_CLK_DI
#define AUDIO_TIMx   TIM4
#define AUDIO_TIMx_IN_CHANNEL   TIM_CHANNEL_1
#define AUDIO_TIMx_OUT_CHANNEL   TIM_CHANNEL_2 /* Select c
#define AUDIO_TIMx_GPIO_CLK_ENABLE()   __HAL_RCC_GPIOB_
#define AUDIO_TIMx_GPIO_CLK_DISABLE()   __HAL_RCC_GPIOB_
#define AUDIO_TIMx_GPIO_PORT   GPIOB
#define AUDIO_TIMx_IN_GPIO_PIN   GPIO_PIN_6
#define AUDIO_TIMx_OUT_GPIO_PIN   GPIO_PIN_7
#define AUDIO_TIMx_AF   GPIO_AF2_TIM4
#define AUDIODATA_SIZE   2 /* 16-bits audio data size */
#define AUDIO_OK   ((uint8_t)0)
#define AUDIO_ERROR   ((uint8_t)1)
```

#define	**AUDIO_TIMEOUT** ((uint8_t)2)
#define	**DEFAULT_AUDIO_IN_FREQ** I2S_AUDIOFREQ_16K
#define	**DEFAULT_AUDIO_IN_BIT_RESOLUTION** ((uint8_t)16)
#define	**DEFAULT_AUDIO_IN_CHANNEL_NBR** ((uint8_t)2) /* Mono
#define	**DEFAULT_AUDIO_IN_VOLUME** ((uint16_t)64)
#define	**INTERNAL_BUFF_SIZE** (128***DEFAULT_AUDIO_IN_FREQ**/
#define	**PCM_OUT_SIZE** (**DEFAULT_AUDIO_IN_FREQ**/1000*2)
#define	**CHANNEL_DEMUX_MASK** ((uint8_t)0x55)
#define	**CODEC_RESET_DELAY** ((uint8_t)5)
#define	**OUTPUT_DEVICE_HEADPHONE1** OUTPUT_DEVICE_HEA
#define	**OUTPUT_DEVICE_HEADPHONE2** OUTPUT_DEVICE_SPE output of the wm8994 */
#define	**DMA_MAX**(x) (((x) <= **DMA_MAX_SZE**)? (x):**DMA_MAX_SZ**

Functions

uint8_t	**BSP_AUDIO_OUT_Init** (uint16_t OutputDevice, uint8_t Volume, uint32_t AudioFreq) Configures 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 SAI 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) Updates the audio frequency.
void	**BSP_AUDIO_OUT_SetAudioFrameSlot** (uint32_t AudioFrameSlot) Updates the Audio frame slot configuration.
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_DeInit** (void) Deinit the audio peripherals.
void	**BSP_AUDIO_OUT_TransferComplete_CallBack** (void) Manages the DMA full Transfer complete event.

void	**BSP_AUDIO_OUT_HalfTransfer_CallBack** (void) Manages the DMA Half Transfer complete event.
void	**BSP_AUDIO_OUT_Error_CallBack** (void) Manages the DMA FIFO error event.
void	**BSP_AUDIO_OUT_ClockConfig** (SAI_HandleTypeDef *hsai, uint32_t AudioFreq, void *Params) Clock Config.
void	**BSP_AUDIO_OUT_MspInit** (SAI_HandleTypeDef *hsai, void *Params) Initializes BSP_AUDIO_OUT MSP.
void	**BSP_AUDIO_OUT_MspDeInit** (SAI_HandleTypeDef *hsai, void *Params) Deinitializes SAI MSP.
uint8_t	**BSP_AUDIO_IN_Init** (uint32_t AudioFreq, uint32_t BitRes, uint32_t ChnlNbr) Initializes wave recording.
uint8_t	**BSP_AUDIO_IN_Record** (uint16_t *pData, uint32_t Size) Starts audio recording.
uint8_t	**BSP_AUDIO_IN_Stop** (void) Stops audio recording.
uint8_t	**BSP_AUDIO_IN_Pause** (void) Pauses the audio file stream.
uint8_t	**BSP_AUDIO_IN_Resume** (void) Resumes the audio file stream.
uint8_t	**BSP_AUDIO_IN_SetVolume** (uint8_t Volume) Controls the audio in volume level.
void	**BSP_AUDIO_IN_DeInit** (void) Deinit the audio IN peripherals.
uint8_t	**BSP_AUDIO_IN_PDMToPCM** (uint16_t *PDMBuf, uint16_t *PCMBuf) Converts audio format from PDM to PCM.
void	**BSP_AUDIO_IN_TransferComplete_CallBack** (void) User callback when record buffer is filled.
void	**BSP_AUDIO_IN_HalfTransfer_CallBack** (void)

	Manages the DMA Half Transfer complete event.
void	**BSP_AUDIO_IN_Error_Callback** (void) Audio IN Error callback function.
void	**BSP_AUDIO_IN_MspInit** (I2S_HandleTypeDef *hi2s, void *Params) Initializes BSP_AUDIO_IN MSP.
void	**BSP_AUDIO_IN_MspDeInit** (I2S_HandleTypeDef *hi2s, void *Params) DeInitializes BSP_AUDIO_IN MSP.

Variables

__IO uint16_t	**AudioInVolume**

Detailed Description

This file contains the common defines and functions prototypes for the **stm32446e_eval_audio.c** driver.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

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 **stm32446e_eval_audio.h**.

STM32446E_EVAL BSP User Manual

stm32446e_eval_camera.c File Reference

This file includes the driver for Camera modules mounted on STM32446E-EVAL evaluation board. More...

```
#include "stm32446e_eval_camera.h"
```

Go to the source code of this file.

Functions

static uint32_t	**GetSize** (uint32_t Resolution) Get the capture size.
uint8_t	**BSP_CAMERA_Init** (uint32_t Resolution) Initializes the camera.
uint8_t	**BSP_CAMERA_DeInit** (void) DeInitializes the camera.
void	**BSP_CAMERA_ContinuousStart** (uint8_t *buff) Starts the camera capture in continuous mode.
void	**BSP_CAMERA_SnapshotStart** (uint8_t *buff) Starts the camera capture in snapshot mode.
void	**BSP_CAMERA_Suspend** (void) Suspend the CAMERA capture.
void	**BSP_CAMERA_Resume** (void) Resume the CAMERA capture.
uint8_t	**BSP_CAMERA_Stop** (void) Stop the CAMERA capture.
void	**BSP_CAMERA_HwReset** (void) CAMERA hardware reset.
void	**BSP_CAMERA_PwrDown** (void) CAMERA power down.
void	**BSP_CAMERA_ContrastBrightnessConfig** (uint32_t contrast_level, uint32_t brightness_level) Configures the camera contrast and brightness.
void	**BSP_CAMERA_BlackWhiteConfig** (uint32_t Mode) Configures the camera white balance.
void	**BSP_CAMERA_ColorEffectConfig** (uint32_t Effect) Configures the camera color effect.
void	**BSP_CAMERA_IRQHandler** (void) Handles DCMI interrupt request.
void	**BSP_CAMERA_DMA_IRQHandler** (void)

	Handles DMA interrupt request.
__weak void	**BSP_CAMERA_MspInit** (DCMI_HandleTypeDef *hdcmi, void *Params) Initializes the DCMI MSP.
__weak void	**BSP_CAMERA_MspDeInit** (DCMI_HandleTypeDef *hdcmi, void *Params) DeInitializes the DCMI MSP.
void	**HAL_DCMI_LineEventCallback** (DCMI_HandleTypeDef *hdcmi) Line event callback.
__weak void	**BSP_CAMERA_LineEventCallback** (void) Line Event callback.
void	**HAL_DCMI_VsyncEventCallback** (DCMI_HandleTypeDef *hdcmi) VSYNC event callback.
__weak void	**BSP_CAMERA_VsyncEventCallback** (void) VSYNC Event callback.
void	**HAL_DCMI_FrameEventCallback** (DCMI_HandleTypeDef *hdcmi) Frame event callback.
__weak void	**BSP_CAMERA_FrameEventCallback** (void) Frame Event callback.
void	**HAL_DCMI_ErrorCallback** (DCMI_HandleTypeDef *hdcmi) Error callback.
__weak void	**BSP_CAMERA_ErrorCallback** (void) Error callback.

Variables

CAMERA_DrvTypeDef *	**camera_drv**
static DCMI_HandleTypeDef	**hDcmiEval**
uint32_t	**currentResolution**
static uint32_t	**cameraHwAddress**

Detailed Description

This file includes the driver for Camera modules mounted on STM32446E-EVAL evaluation board.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

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 **stm32446e_eval_camera.c**.

STM32446E_EVAL BSP User Manual

stm32446e_eval_camera.h File Reference

This file contains the common defines and functions prototypes for the **stm32446e_eval_camera.c** driver. More...

```
#include "../Components/s5k5cag/s5k5cag.h" #include
"stm32446e_eval_io.h"
```

Go to the source code of this file.

Defines

#define	**RESOLUTION_R160x120**	CAMERA_R160x120 /* QQVGA Resolution */
#define	**RESOLUTION_R320x240**	CAMERA_R320x240 /* QVGA Resolution */
#define	**RESOLUTION_R480x272**	CAMERA_R480x272 /* 480x272 Resolution */
#define	**RESOLUTION_R640x480**	CAMERA_R640x480 /* VGA Resolution */

Enumerations

enum	**Camera_StatusTypeDef** { **CAMERA_OK** = 0x00, **CAMERA_ERROR** = 0x01, **CAMERA_TIMEOUT** = 0x02 } Camera State structures definition. More...

Functions

uint8_t	**BSP_CAMERA_Init** (uint32_t Resolution) Initializes the camera.
uint8_t	**BSP_CAMERA_DeInit** (void) DeInitializes the camera.
void	**BSP_CAMERA_ContinuousStart** (uint8_t *buff) Starts the camera capture in continuous mode.
void	**BSP_CAMERA_SnapshotStart** (uint8_t *buff) Starts the camera capture in snapshot mode.
void	**BSP_CAMERA_Suspend** (void) Suspend the CAMERA capture.
void	**BSP_CAMERA_Resume** (void) Resume the CAMERA capture.
uint8_t	**BSP_CAMERA_Stop** (void) Stop the CAMERA capture.
void	**BSP_CAMERA_HwReset** (void) CAMERA hardware reset.
void	**BSP_CAMERA_PwrDown** (void) CAMERA power down.
void	**BSP_CAMERA_LineEventCallback** (void) Line Event callback.
void	**BSP_CAMERA_VsyncEventCallback** (void) VSYNC Event callback.
void	**BSP_CAMERA_FrameEventCallback** (void) Frame Event callback.
void	**BSP_CAMERA_ErrorCallback** (void) Error callback.
void	**BSP_CAMERA_ContrastBrightnessConfig** (uint32_t contrast_level, uint32_t brightness_level) Configures the camera contrast and brightness.
void	**BSP_CAMERA_BlackWhiteConfig** (uint32_t Mode)

	Configures the camera white balance.
void	**BSP_CAMERA_ColorEffectConfig** (uint32_t Effect) Configures the camera color effect.
void	**BSP_CAMERA_IRQHandler** (void) Handles DCMI interrupt request.
void	**BSP_CAMERA_DMA_IRQHandler** (void) Handles DMA interrupt request.
void	**BSP_CAMERA_MspInit** (DCMI_HandleTypeDef *hdcmi, void *Params) Initializes the DCMI MSP.
void	**BSP_CAMERA_MspDeInit** (DCMI_HandleTypeDef *hdcmi, void *Params) DeInitializes the DCMI MSP.

Detailed Description

This file contains the common defines and functions prototypes for the **stm32446e_eval_camera.c** driver.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

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 **stm32446e_eval_camera.h**.

STM32446E_EVAL BSP User Manual

stm32446e_eval_eeprom.c File Reference

This file provides a set of functions needed to manage an I2C M24LR64 EEPROM memory. To be able to use this driver, the switch EE_M24LR64 must be defined in your toolchain compiler preprocessor. More...

```
#include "stm32446e_eval_eeprom.h"
```

Go to the source code of this file.

Functions

uint32_t	**BSP_EEPROM_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
uint8_t	**BSP_EEPROM_DeInit** (void) DeInitializes the EEPROM.
uint32_t	**BSP_EEPROM_ReadBuffer** (uint8_t *pBuffer, uint16_t ReadAddr, uint16_t *NumByteToRead) Reads a block of data from the EEPROM.
uint32_t	**BSP_EEPROM_WritePage** (uint8_t *pBuffer, uint16_t WriteAddr, uint8_t *NumByteToWrite) Writes more than one byte to the EEPROM with a single WRITE cycle.
uint32_t	**BSP_EEPROM_WriteBuffer** (uint8_t *pBuffer, uint16_t WriteAddr, uint16_t NumByteToWrite) Writes buffer of data to the I2C EEPROM.
uint32_t	**BSP_EEPROM_WaitEepromStandbyState** (void) Wait for EEPROM Standby state.
__weak void	**BSP_EEPROM_TIMEOUT_UserCallback** (void) Basic management of the timeout situation.

Variables

__IO uint16_t	**EEPROMAddress** = 0
__IO uint16_t	**EEPROMDataRead**
__IO uint8_t	**EEPROMDataWrite**

Detailed Description

This file provides a set of functions needed to manage an I2C M24LR64 EEPROM memory. To be able to use this driver, the switch EE_M24LR64 must be defined in your toolchain compiler preprocessor.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016
==========
Notes:

- This driver is intended for STM32F4xx families devices only.
- The I2C EEPROM memory (M24LR64) is available on separate daughter board ANT7-M24LR-A, which is not provided with the STM32446E_EVAL board. To use this driver you have to connect the ANT7-M24LR-A to CN3 connector of STM32446E_EVAL board.

  ==========

It implements a high level communication layer for read and write from/to this memory. The needed STM32F4xx hardware resources (I2C and GPIO) are defined in **stm32446e_eval.h** file, and the initialization is performed in **EEPROM_IO_Init()** function declared in **stm32446e_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **EEPROM_IO_Init()** function.

> **Note:**
> In this driver, basic read and write functions (**BSP_EEPROM_ReadBuffer()** and **BSP_EEPROM_WritePage()**) use DMA mode to perform the data transfer to/from EEPROM memory.

> Regarding **BSP_EEPROM_WritePage()**, it is a optimized function to perform small write (less than 1 page) BUT The number of bytes (combined to write start address) must not cross the EEPROM page boundary. This function can only write into the boundaries of an EEPROM page. This function doesn't check on boundaries condition (in this driver the function **BSP_EEPROM_WriteBuffer()** which calls **BSP_EEPROM_WritePage()** is responsible of checking on Page boundaries).

+--+ | Pin assignment for M24LR64 EEPROM | +---------------------------------------+-----------+-------------+ | STM32F4xx I2C Pins | EEPROM | Pin | +---------------------------------------+-----------+-------------+ | . | E0(GND) | 1 (0V) | | . | AC0 | 2 | | . | AC1 | 3 | | . | VSS | 4 (0V) | | SDA | SDA | 5 | | SCL | SCL | 6 | | . | E1(GND) | 7 (0V) | | . | 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 **stm32446e_eval_eeprom.c**.

STM32446E_EVAL BSP User Manual

stm32446e_eval_eeprom.h File Reference

This file contains all the functions prototypes for the **stm32446e_eval_eeprom.c** firmware driver. More...

```
#include "stm32446e_eval.h"
```

Go to the source code of this file.

Defines

```
#define EEPROM_PAGESIZE   ((uint8_t)4)
#define EEPROM_MAX_SIZE   ((uint16_t)0x2000) /* 64Kbit */
#define EEPROM_MAX_TRIALS   ((uint32_t)3000)
#define EEPROM_OK   ((uint32_t)0)
#define EEPROM_FAIL   ((uint32_t)1)
#define EEPROM_TIMEOUT   ((uint32_t)2)
```

Functions

uint32_t	**BSP_EEPROM_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
uint8_t	**BSP_EEPROM_DeInit** (void) DeInitializes the EEPROM.
uint32_t	**BSP_EEPROM_ReadBuffer** (uint8_t *pBuffer, uint16_t ReadAddr, uint16_t *NumByteToRead) Reads a block of data from the EEPROM.
uint32_t	**BSP_EEPROM_WritePage** (uint8_t *pBuffer, uint16_t WriteAddr, uint8_t *NumByteToWrite) Writes more than one byte to the EEPROM with a single WRITE cycle.
uint32_t	**BSP_EEPROM_WriteBuffer** (uint8_t *pBuffer, uint16_t WriteAddr, uint16_t NumByteToWrite) Writes buffer of data to the I2C EEPROM.
uint32_t	**BSP_EEPROM_WaitEepromStandbyState** (void) Wait for EEPROM Standby state.
void	**BSP_EEPROM_TIMEOUT_UserCallback** (void) Basic management of the timeout situation.
void	**EEPROM_IO_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_IO_WriteData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Write data to I2C EEPROM driver in using DMA channel.
HAL_StatusTypeDef	**EEPROM_IO_ReadData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize)

	Read data from I2C EEPROM driver in using DMA channel.
HAL_StatusTypeDef	**EEPROM_IO_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.

Detailed Description

This file contains all the functions prototypes for the **stm32446e_eval_eeprom.c** firmware driver.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

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 **stm32446e_eval_eeprom.h**.

STM32446E_EVAL BSP User Manual

stm32446e_eval_io.c File Reference

This file provides a set of functions needed to manage the IO pins on STM32446E-EVAL evaluation board. More...

```
#include "stm32446e_eval_io.h"
```

Go to the source code of this file.

Functions

uint8_t	**BSP_IO_Init** (void) Initializes and configures the IO functionalities and configures all necessary hardware resources (MFX, ...).
uint8_t	**BSP_IO_DeInit** (void) DeInit allows Mfx Initialization to be executed again.
uint32_t	**BSP_IO_ITGetStatus** (uint32_t IoPin) Gets the selected pins IT status.
void	**BSP_IO_ITClear** (void) Clears all the IO IT pending bits.
uint8_t	**BSP_IO_ConfigPin** (uint32_t IoPin, IO_ModeTypedef IoMode) Configures the IO pin(s) according to IO mode structure value.
uint8_t	**BSP_IO_ConfigIrqOutPin** (uint8_t IoIrqOutPinPolarity, uint8_t IoIrqOutPinType) Sets the IRQ_OUT pin polarity and type.
void	**BSP_IO_WritePin** (uint32_t IoPin, **BSP_IO_PinStateTypeDef** PinState) Sets the selected pins state.
uint32_t	**BSP_IO_ReadPin** (uint32_t IoPin) Gets the selected pins current state.
void	**BSP_IO_TogglePin** (uint32_t IoPin) Toggles the selected pins state.

Variables

static IO_DrvTypeDef *	**IoDrv** = NULL
static uint8_t	**mfxstm32l152Identifier**

Detailed Description

This file provides a set of functions needed to manage the IO pins on STM32446E-EVAL evaluation board.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

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 **stm32446e_eval_io.c**.

STM32446E_EVAL BSP User Manual

stm32446e_eval_io.h File Reference

This file contains the common defines and functions prototypes for the **stm32446e_eval_io.c** driver. More...

```
#include "stm32446e_eval.h" #include
"../Components/mfxstm32l152/mfxstm32l152.h"
```

Go to the source code of this file.

Defines

#define	**IO_PIN_0** ((uint32_t)0x0001)
#define	**IO_PIN_1** ((uint32_t)0x0002)
#define	**IO_PIN_2** ((uint32_t)0x0004)
#define	**IO_PIN_3** ((uint32_t)0x0008)
#define	**IO_PIN_4** ((uint32_t)0x0010)
#define	**IO_PIN_5** ((uint32_t)0x0020)
#define	**IO_PIN_6** ((uint32_t)0x0040)
#define	**IO_PIN_7** ((uint32_t)0x0080)
#define	**IO_PIN_8** ((uint32_t)0x0100)
#define	**IO_PIN_9** ((uint32_t)0x0200)
#define	**IO_PIN_10** ((uint32_t)0x0400)
#define	**IO_PIN_11** ((uint32_t)0x0800)
#define	**IO_PIN_12** ((uint32_t)0x1000)
#define	**IO_PIN_13** ((uint32_t)0x2000)
#define	**IO_PIN_14** ((uint32_t)0x4000)
#define	**IO_PIN_15** ((uint32_t)0x8000)
#define	**IO_PIN_16** ((uint32_t)0x010000)
#define	**IO_PIN_17** ((uint32_t)0x020000)
#define	**IO_PIN_18** ((uint32_t)0x040000)
#define	**IO_PIN_19** ((uint32_t)0x080000)
#define	**IO_PIN_20** ((uint32_t)0x100000)
#define	**IO_PIN_21** ((uint32_t)0x200000)
#define	**IO_PIN_22** ((uint32_t)0x400000)
#define	**IO_PIN_23** ((uint32_t)0x800000)
#define	**IO_PIN_ALL** ((uint32_t)0xFFFFFF)

Enumerations

enum	**BSP_IO_PinStateTypeDef** { **BSP_IO_PIN_RESET** = 0, **BSP_IO_PIN_SET** = 1 }
enum	**IO_StatusTypeDef** { **IO_OK** = 0, **IO_ERROR** = 1, **IO_TIMEOUT** = 2 }

Functions

uint8_t	**BSP_IO_Init** (void) Initializes and configures the IO functionalities and configures all necessary hardware resources (MFX, ...).
uint8_t	**BSP_IO_DeInit** (void) DeInit allows Mfx Initialization to be executed again.
uint8_t	**BSP_IO_ConfigIrqOutPin** (uint8_t IoIrqOutPinPolarity, uint8_t IoIrqOutPinType) Sets the IRQ_OUT pin polarity and type.
uint32_t	**BSP_IO_ITGetStatus** (uint32_t IoPin) Gets the selected pins IT status.
void	**BSP_IO_ITClear** (void) Clears all the IO IT pending bits.
uint8_t	**BSP_IO_ConfigPin** (uint32_t IoPin, IO_ModeTypedef IoMode) Configures the IO pin(s) according to IO mode structure value.
void	**BSP_IO_WritePin** (uint32_t IoPin, **BSP_IO_PinStateTypeDef** PinState) Sets the selected pins state.
uint32_t	**BSP_IO_ReadPin** (uint32_t IoPin) Gets the selected pins current state.
void	**BSP_IO_TogglePin** (uint32_t IoPin) Toggles the selected pins state.

Detailed Description

This file contains the common defines and functions prototypes for the **stm32446e_eval_io.c** driver.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

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 **stm32446e_eval_io.h**.

STM32446E_EVAL BSP User Manual

stm32446e_eval_lcd.c File Reference

This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM32446E-EVAL evaluation board. More...

```
#include "stm32446e_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)((Points + Z)->X))
#define POLY_Y(Z)   ((int32_t)((Points + Z)->Y))
#define ABS(X)   ((X) > 0 ? (X) : -(X))
```

Functions

static void	**DrawChar** (uint16_t Xpos, uint16_t Ypos, const uint8_t *c) Draws a character on LCD.
static void	**SetDisplayWindow** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Sets display window.
static void	**FillTriangle** (uint16_t x1, uint16_t x2, uint16_t x3, uint16_t y1, uint16_t y2, uint16_t y3) Fills a triangle (between 3 points).
uint8_t	**BSP_LCD_Init** (void) Initializes the LCD.
uint8_t	**BSP_LCD_DeInit** (void) DeInitializes 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 *fonts) 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 *Text, **Line_ModeTypdef** Mode) Displays characters on the LCD.
void	**BSP_LCD_DisplayStringAtLine** (uint16_t Line, uint8_t *ptr) Displays a character on the LCD.
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) Draws a pixel on LCD.
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** Points, 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 (16 bpp).
void	**BSP_LCD_DrawRGBImage** (uint16_t Xpos, uint16_t Ypos, uint16_t Xsize, uint16_t Ysize, uint8_t *pdata) Draws RGB Image (16 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_FillPolygon** (**pPoint** Points, uint16_t PointCount) Draws 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 *	**LcdDrv**

Detailed Description

This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM32446E-EVAL evaluation board.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

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 **stm32446e_eval_lcd.c**.

STM32446E_EVAL BSP User Manual

Data Structures | Defines | Typedefs | Enumerations | Functions

stm32446e_eval_lcd.h File Reference

This file contains the common defines and functions prototypes for the **stm32446e_eval_lcd.c** driver. More...

```
#include "stm32446e_eval.h" #include
"../Components/ili9325/ili9325.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** ((uint8_t)0x00) LCD status structure definition.
#define	**LCD_ERROR** ((uint8_t)0x01)
#define	**LCD_TIMEOUT** ((uint8_t)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_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.
uint8_t	**BSP_LCD_DeInit** (void) DeInitializes 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 *fonts) 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 *ptr) Displays a character on the LCD.
void	**BSP_LCD_DisplayStringAt** (uint16_t Xpos, uint16_t Ypos, uint8_t *Text, **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) Draws a pixel on LCD.
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** Points, 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 (16 bpp).
void	**BSP_LCD_DrawRGBImage** (uint16_t Xpos, uint16_t Ypos, uint16_t Xsize, uint16_t Ysize, uint8_t *pbmp) Draws RGB Image (16 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_FillPolygon** (**pPoint** Points, uint16_t PointCount) Draws 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_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 **stm32446e_eval_lcd.c** driver.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

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 **stm32446e_eval_lcd.h**.

STM32446E_EVAL BSP User Manual

Functions | Variables

stm32446e_eval_qspi.c File Reference

This file includes a standard driver for the N25Q256A QSPI memory mounted on STM32446E-EVAL board. More...

```
#include "stm32446e_eval_qspi.h"
```

Go to the source code of this file.

Functions

static uint8_t	**QSPI_ResetMemory** (QSPI_HandleTypeDef *hqspi) This function reset the QSPI memory.
static uint8_t	**QSPI_EnterFourBytesAddress** (QSPI_HandleTypeDef *hqspi) This function set the QSPI memory in 4-byte address mode.
static uint8_t	**QSPI_DummyCyclesCfg** (QSPI_HandleTypeDef *hqspi) This function configure the dummy cycles on memory side.
static uint8_t	**QSPI_WriteEnable** (QSPI_HandleTypeDef *hqspi) This function send a Write Enable and wait it is effective.
static uint8_t	**QSPI_AutoPollingMemReady** (QSPI_HandleTypeDef *hqspi, uint32_t Timeout) This function read the SR of the memory and wait the EOP.
uint8_t	**BSP_QSPI_Init** (void) Initializes the QSPI interface.
uint8_t	**BSP_QSPI_DeInit** (void) De-Initializes the QSPI interface.
uint8_t	**BSP_QSPI_Read** (uint8_t *pData, uint32_t ReadAddr, uint32_t Size) Reads an amount of data from the QSPI memory.
uint8_t	**BSP_QSPI_Write** (uint8_t *pData, uint32_t WriteAddr, uint32_t Size) Writes an amount of data to the QSPI memory.
uint8_t	**BSP_QSPI_Erase_Block** (uint32_t BlockAddress) Erases the specified block of the QSPI memory.
uint8_t	**BSP_QSPI_Erase_Chip** (void) Erases the entire QSPI memory.

uint8_t	**BSP_QSPI_GetStatus** (void) Reads current status of the QSPI memory.
uint8_t	**BSP_QSPI_GetInfo** (**QSPI_Info** *pInfo) Return the configuration of the QSPI memory.
uint8_t	**BSP_QSPI_MemoryMappedMode** (void) Configure the QSPI in memory-mapped mode.
__weak void	**BSP_QSPI_MspInit** (QSPI_HandleTypeDef *hqspi, void *Params) QSPI MSP Initialization This function configures the hardware resources used in this example:
__weak void	**BSP_QSPI_MspDeInit** (QSPI_HandleTypeDef *hqspi, void *Params) QSPI MSP De-Initialization This function frees the hardware resources used in this example:

Variables

QSPI_HandleTypeDef	**QSPIHandle**

Detailed Description

This file includes a standard driver for the N25Q256A QSPI memory mounted on STM32446E-EVAL board.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

```
  ==========================================
==================================
                     ##### How to use this dri
ver #####
  ==========================================
==================================
  [..]
   (#) This driver is used to drive the N25Q25
6A QSPI external
       memory mounted on STM32446E-EVAL evalua
tion board.

   (#) This driver need a specific component d
river (N25Q256A) to be included with.

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

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

            (see the QSPI memory data sheet)
```

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 **stm32446e_eval_qspi.c**.

STM32446E_EVAL BSP User Manual

stm32446e_eval_qspi.h File Reference

This file contains the common defines and functions prototypes for the **stm32446e_eval_qspi.c** driver. More...

```
#include "stm32f4xx_hal.h" #include
"../Components/n25q256a/n25q256a.h"
```

Go to the source code of this file.

Data Structures

struct **QSPI_Info**

Defines

```
#define QSPI_OK ((uint8_t)0x00)
#define QSPI_ERROR ((uint8_t)0x01)
#define QSPI_BUSY ((uint8_t)0x02)
#define QSPI_NOT_SUPPORTED ((uint8_t)0x04)
#define QSPI_SUSPENDED ((uint8_t)0x08)
#define QSPI_CLK_ENABLE() __HAL_RCC_QSPI_CLK_ENABLE()
#define QSPI_CLK_DISABLE() __HAL_RCC_QSPI_CLK_DISABLE(
#define QSPI_CS_GPIO_CLK_ENABLE() __HAL_RCC_GPIOG_CL
#define QSPI_CS_GPIO_CLK_DISABLE() __HAL_RCC_GPIOG_CL
#define QSPI_CLK_GPIO_CLK_ENABLE() __HAL_RCC_GPIOD_C
#define QSPI_CLK_GPIO_CLK_DISABLE() __HAL_RCC_GPIOD_C
#define QSPI_Dx_GPIO_CLK_ENABLE() __HAL_RCC_GPIOF_CLK
#define QSPI_Dx_GPIO_CLK_DISABLE() __HAL_RCC_GPIOF_CL
#define QSPI_FORCE_RESET() __HAL_RCC_QSPI_FORCE_RESE
#define QSPI_RELEASE_RESET() __HAL_RCC_QSPI_RELEASE_
#define QSPI_CS_PIN GPIO_PIN_6
#define QSPI_CS_GPIO_PORT GPIOG
#define QSPI_CLK_PIN GPIO_PIN_3
#define QSPI_CLK_GPIO_PORT GPIOD
#define QSPI_D0_PIN GPIO_PIN_8
#define QSPI_D0_GPIO_PORT GPIOF
#define QSPI_D1_PIN GPIO_PIN_9
#define QSPI_D1_GPIO_PORT GPIOF
#define QSPI_D2_PIN GPIO_PIN_7
#define QSPI_D2_GPIO_PORT GPIOF
#define QSPI_D3_PIN GPIO_PIN_6
#define QSPI_D3_GPIO_PORT GPIOF
```

Functions

uint8_t	**BSP_QSPI_Init** (void) Initializes the QSPI interface.
uint8_t	**BSP_QSPI_DeInit** (void) De-Initializes the QSPI interface.
uint8_t	**BSP_QSPI_Read** (uint8_t *pData, uint32_t ReadAddr, uint32_t Size) Reads an amount of data from the QSPI memory.
uint8_t	**BSP_QSPI_Write** (uint8_t *pData, uint32_t WriteAddr, uint32_t Size) Writes an amount of data to the QSPI memory.
uint8_t	**BSP_QSPI_Erase_Block** (uint32_t BlockAddress) Erases the specified block of the QSPI memory.
uint8_t	**BSP_QSPI_Erase_Chip** (void) Erases the entire QSPI memory.
uint8_t	**BSP_QSPI_GetStatus** (void) Reads current status of the QSPI memory.
uint8_t	**BSP_QSPI_GetInfo** (**QSPI_Info** *pInfo) Return the configuration of the QSPI memory.
uint8_t	**BSP_QSPI_MemoryMappedMode** (void) Configure the QSPI in memory-mapped mode.
void	**BSP_QSPI_MspInit** (QSPI_HandleTypeDef *hqspi, void *Params) QSPI MSP Initialization This function configures the hardware resources used in this example:
void	**BSP_QSPI_MspDeInit** (QSPI_HandleTypeDef *hqspi, void *Params) QSPI MSP De-Initialization This function frees the hardware resources used in this example:

Detailed Description

This file contains the common defines and functions prototypes for the **stm32446e_eval_qspi.c** driver.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

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 **stm32446e_eval_qspi.h**.

STM32446E_EVAL BSP User Manual

stm32446e_eval_sd.c File Reference

This file includes the uSD card driver mounted on STM32446E-EVAL evaluation board. More...

```
#include "stm32446e_eval_sd.h"
```

Go to the source code of this file.

Functions

uint8_t	**BSP_SD_Init** (void) Initializes the SD card device.
uint8_t	**BSP_SD_DeInit** (void) DeInitializes 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.
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.
__weak void	**BSP_SD_MspInit** (SD_HandleTypeDef *hsd, void *Params) Initializes the SD MSP.
__weak void	**BSP_SD_MspDeInit** (SD_HandleTypeDef *hsd, void *Params) DeInitializes the SD MSP.
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

static SD_HandleTypeDef	**uSdHandle**
static **SD_CardInfo**	**uSdCardInfo**
static uint8_t	**UseExtiModeDetection** = 0

Detailed Description

This file includes the uSD card driver mounted on STM32446E-EVAL evaluation board.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

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 **stm32446e_eval_sd.c**.

STM32446E_EVAL BSP User Manual

Defines | Functions

stm32446e_eval_sd.h File Reference

This file contains the common defines and functions prototypes for the **stm32446e_eval_sd.c** driver. More...

```
#include "stm32446e_eval.h" #include "stm32446e_eval_io.h"
```

Go to the source code of this file.

Defines

#define	**SD_CardInfo** HAL_SD_CardInfoTypedef SD Card information structure.
#define	**MSD_OK** ((uint8_t)0x00) SD status structure definition.
#define	**MSD_ERROR** ((uint8_t)0x01)
#define	**MSD_ERROR_SD_NOT_PRESENT** ((uint8_t)0x02)
#define	**SD_PRESENT** ((uint8_t)0x01)
#define	**SD_NOT_PRESENT** ((uint8_t)0x00)
#define	**SD_DATATIMEOUT** ((uint32_t)100000000)
#define	**__DMAx_TxRx_CLK_ENABLE** __HAL_RCC_DMA2_CLK_E
#define	**SD_DMAx_Tx_CHANNEL** DMA_CHANNEL_4
#define	**SD_DMAx_Rx_CHANNEL** DMA_CHANNEL_4
#define	**SD_DMAx_Tx_STREAM** DMA2_Stream6
#define	**SD_DMAx_Rx_STREAM** DMA2_Stream3
#define	**SD_DMAx_Tx_IRQn** DMA2_Stream6_IRQn
#define	**SD_DMAx_Rx_IRQn** DMA2_Stream3_IRQn
#define	**SD_DMAx_Tx_IRQHandler** DMA2_Stream6_IRQHandler
#define	**SD_DMAx_Rx_IRQHandler** DMA2_Stream3_IRQHandler
#define	**SD_DetectIRQHandler**() HAL_GPIO_EXTI_IRQHandler(GPI

Functions

uint8_t	**BSP_SD_Init** (void) Initializes the SD card device.
uint8_t	**BSP_SD_DeInit** (void) DeInitializes the SD card device.
uint8_t	**BSP_SD_ITConfig** (void) Configures Interrupt mode for SD detection pin.
void	**BSP_SD_DetectIT** (void)
void	**BSP_SD_DetectCallback** (void)
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.
void	**BSP_SD_MspInit** (SD_HandleTypeDef *hsd, void *Params) Initializes the SD MSP.
void	**BSP_SD_MspDeInit** (SD_HandleTypeDef *hsd, void *Params)

DeInitializes the SD MSP.

Detailed Description

This file contains the common defines and functions prototypes for the **stm32446e_eval_sd.c** driver.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

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 **stm32446e_eval_sd.h**.

STM32446E_EVAL BSP User Manual

stm32446e_eval_sdram.c File Reference

This file includes the SDRAM driver for the MT48LC4M32B2B5-7 memory device mounted on STM32446E-EVAL evaluation board. More...

```
#include "stm32446e_eval_sdram.h"
```

Go to the source code of this file.

Functions

uint8_t	**BSP_SDRAM_Init** (void) Initializes the SDRAM device.
uint8_t	**BSP_SDRAM_DeInit** (void) DeInitializes the SDRAM device.
void	**BSP_SDRAM_Initialization_sequence** (uint32_t RefreshCount) Programs the SDRAM device.
uint8_t	**BSP_SDRAM_ReadData** (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize) Reads an mount of data from the SDRAM memory in polling mode.
uint8_t	**BSP_SDRAM_ReadData_DMA** (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize) Reads an mount of data from the SDRAM memory in DMA mode.
uint8_t	**BSP_SDRAM_WriteData** (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize) Writes an mount of data to the SDRAM memory in polling mode.
uint8_t	**BSP_SDRAM_WriteData_DMA** (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize) Writes an mount of data to the SDRAM memory in DMA mode.
uint8_t	**BSP_SDRAM_Sendcmd** (FMC_SDRAM_CommandTypeDef *SdramCmd) Sends command to the SDRAM bank.
void	**BSP_SDRAM_DMA_IRQHandler** (void) Handles SDRAM DMA transfer interrupt request.
__weak void	**BSP_SDRAM_MspInit** (SDRAM_HandleTypeDef *hsdram, void *Params) Initializes SDRAM MSP.

__weak void	**BSP_SDRAM_MspDeInit** (SDRAM_HandleTypeDef *hsdram, void *Params) DeInitializes SDRAM MSP.

Variables

static SDRAM_HandleTypeDef	**sdramHandle**
static FMC_SDRAM_TimingTypeDef	**Timing**
static FMC_SDRAM_CommandTypeDef	**Command**

Detailed Description

This file includes the SDRAM driver for the MT48LC4M32B2B5-7 memory device mounted on STM32446E-EVAL evaluation board.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

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 **stm32446e_eval_sdram.c**.

STM32446E_EVAL BSP User Manual

stm32446e_eval_sdram.h File Reference

This file contains the common defines and functions prototypes for the **stm32446e_eval_sdram.c** driver. More...

```
#include "stm32f4xx_hal.h"
```

Go to the source code of this file.

Defines

#define	**SDRAM_OK** ((uint8_t)0x00) SDRAM status structure definition.
#define	**SDRAM_ERROR** ((uint8_t)0x01)
#define	**SDRAM_DEVICE_ADDR** ((uint32_t)0xC0000000)
#define	**SDRAM_DEVICE_SIZE** ((uint32_t)0x800000) /* SDRAM devi
#define	**SDRAM_MEMORY_WIDTH** FMC_SDRAM_MEM_BUS_WID
#define	**SDCLOCK_PERIOD** FMC_SDRAM_CLOCK_PERIOD_2
#define	**REFRESH_COUNT** ((uint32_t)0x0569) /* SDRAM refresh cou clock) */
#define	**SDRAM_TIMEOUT** ((uint32_t)0xFFFF)
#define	**__DMAx_CLK_ENABLE** __HAL_RCC_DMA2_CLK_ENABLE
#define	**__DMAx_CLK_DISABLE** __HAL_RCC_DMA2_CLK_DISABL
#define	**SDRAM_DMAx_CHANNEL** DMA_CHANNEL_0
#define	**SDRAM_DMAx_STREAM** DMA2_Stream0
#define	**SDRAM_DMAx_IRQn** DMA2_Stream0_IRQn
#define	**SDRAM_DMAx_IRQHandler** DMA2_Stream0_IRQHandler
#define	**SDRAM_MODEREG_BURST_LENGTH_1** ((uint16_t)0x0000 FMC SDRAM Mode definition register defines.
#define	**SDRAM_MODEREG_BURST_LENGTH_2** ((uint16_t)0x0001
#define	**SDRAM_MODEREG_BURST_LENGTH_4** ((uint16_t)0x0002
#define	**SDRAM_MODEREG_BURST_LENGTH_8** ((uint16_t)0x0004
#define	**SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL** ((uint16
#define	**SDRAM_MODEREG_BURST_TYPE_INTERLEAVED** ((uint1
#define	**SDRAM_MODEREG_CAS_LATENCY_2** ((uint16_t)0x0020)
#define	**SDRAM_MODEREG_CAS_LATENCY_3** ((uint16_t)0x0030)
#define	**SDRAM_MODEREG_OPERATING_MODE_STANDARD** ((ui
#define	**SDRAM_MODEREG_WRITEBURST_MODE_PROGRAMME**
#define	**SDRAM_MODEREG_WRITEBURST_MODE_SINGLE** ((uint

Functions

uint8_t	**BSP_SDRAM_Init** (void) Initializes the SDRAM device.
uint8_t	**BSP_SDRAM_DeInit** (void) DeInitializes the SDRAM device.
void	**BSP_SDRAM_Initialization_sequence** (uint32_t RefreshCount) Programs the SDRAM device.
uint8_t	**BSP_SDRAM_ReadData** (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize) Reads an mount of data from the SDRAM memory in polling mode.
uint8_t	**BSP_SDRAM_ReadData_DMA** (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize) Reads an mount of data from the SDRAM memory in DMA mode.
uint8_t	**BSP_SDRAM_WriteData** (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize) Writes an mount of data to the SDRAM memory in polling mode.
uint8_t	**BSP_SDRAM_WriteData_DMA** (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize) Writes an mount of data to the SDRAM memory in DMA mode.
uint8_t	**BSP_SDRAM_Sendcmd** (FMC_SDRAM_CommandTypeDef *SdramCmd) Sends command to the SDRAM bank.
void	**BSP_SDRAM_DMA_IRQHandler** (void) Handles SDRAM DMA transfer interrupt request.
void	**BSP_SDRAM_MspInit** (SDRAM_HandleTypeDef *hsdram, void *Params) Initializes SDRAM MSP.

void **BSP_SDRAM_MspDeInit** (SDRAM_HandleTypeDef *hsdram, void *Params)

DeInitializes SDRAM MSP.

Detailed Description

This file contains the common defines and functions prototypes for the **stm32446e_eval_sdram.c** driver.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

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 **stm32446e_eval_sdram.h**.

STM32446E_EVAL BSP User Manual

stm32446e_eval_ts.c File Reference

This file provides a set of functions needed to manage the Touch Screen on STM32446E-EVAL evaluation board. More...

```
#include "stm32446e_eval_ts.h" #include "stm32446e_eval_io.h"
```

Go to the source code of this file.

Functions

uint8_t	**BSP_TS_Init** (uint16_t xSize, uint16_t ySize) Initializes and configures the touch screen functionalities and configures all necessary hardware resources (GPIOs, clocks..).
uint8_t	**BSP_TS_DeInit** (void) DeInitializes the TouchScreen.
uint8_t	**BSP_TS_ITEnable** (void) Enables the touch screen interrupts.
uint8_t	**BSP_TS_ITDisable** (void) Disables the touch screen interrupts.
uint8_t	**BSP_TS_ITConfig** (void) Configures and enables the touch screen interrupts.
void	**BSP_TS_ITClear** (void) Clears all touch screen interrupts.
uint8_t	**BSP_TS_ITGetStatus** (void) Gets the touch screen interrupt status.
void	**BSP_TS_FIFOClear** (void) Clears touch screen FIFO containing 128 x,y values.
uint8_t	**BSP_TS_GetState** (**TS_StateTypeDef** *TS_State) Returns status and positions of the touch screen.

Variables

static TS_DrvTypeDef *	**ts_driver**
static uint16_t	**tsBundaryX**
static uint16_t	**tsBundaryY**
static uint8_t	**tsOrientation**
static uint8_t	**AddressI2C**

Detailed Description

This file provides a set of functions needed to manage the Touch Screen on STM32446E-EVAL evaluation board.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

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 **stm32446e_eval_ts.c**.

STM32446E_EVAL BSP User Manual

Data Structures | Defines | Enumerations | Functions

stm32446e_eval_ts.h File Reference

This file contains the common defines and functions prototypes for the **stm32446e_eval_ts.c** driver. More...

```
#include "stm32446e_eval.h" #include
"../Components/mfxstm32l152/mfxstm32l152.h"
```

Go to the source code of this file.

Data Structures

struct **TS_StateTypeDef**

Defines

#define **TS_SWAP_NONE** ((uint8_t)0x00)
#define **TS_SWAP_X** ((uint8_t)0x01)
#define **TS_SWAP_Y** ((uint8_t)0x02)
#define **TS_SWAP_XY** ((uint8_t)0x04)

Enumerations

enum	**TS_StatusTypeDef** { **TS_OK** = 0x00, **TS_ERROR** = 0x01, **TS_TIMEOUT** = 0x02 }

Functions

uint8_t	**BSP_TS_Init** (uint16_t xSize, uint16_t ySize) Initializes and configures the touch screen functionalities and configures all necessary hardware resources (GPIOs, clocks..).
uint8_t	**BSP_TS_DeInit** (void) DeInitializes the TouchScreen.
uint8_t	**BSP_TS_GetState** (**TS_StateTypeDef** *TS_State) Returns status and positions of the touch screen.
uint8_t	**BSP_TS_ITEnable** (void) Enables the touch screen interrupts.
uint8_t	**BSP_TS_ITDisable** (void) Disables the touch screen interrupts.
uint8_t	**BSP_TS_ITConfig** (void) Configures and enables the touch screen interrupts.
uint8_t	**BSP_TS_ITGetStatus** (void) Gets the touch screen interrupt status.
void	**BSP_TS_ITClear** (void) Clears all touch screen interrupts.
void	**BSP_TS_FIFOClear** (void) Clears touch screen FIFO containing 128 x,y values.

Detailed Description

This file contains the common defines and functions prototypes for the **stm32446e_eval_ts.c** driver.

Author:
MCD Application Team

Version:
V1.1.1

Date:
13-January-2016

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 **stm32446e_eval_ts.h**.

STM32446E_EVAL BSP User Manual

Modules

Here is a list of all modules:

- BSP
  - STM32446E EVAL
    - STM32446E EVAL LOW LEVEL
      - STM32446E EVAL LOW LEVEL Private TypesDefinitions
      - STM32446E EVAL LOW LEVEL Private Defines
      - STM32446E EVAL LOW LEVEL Private Macros
      - STM32446E EVAL LOW LEVEL Private Variables
      - STM32446E EVAL LOW LEVEL Private FunctionPrototypes
      - STM32446E EVAL LOW LEVEL Private Functions
      - STM32446E EVAL LOW LEVEL Exported Types
      - STM32446E EVAL LOW LEVEL Exported Constants
        - STM32446E EVAL LOW LEVEL LED
        - STM32446E EVAL LOW LEVEL BUTTON
        - STM32446E EVAL LOW LEVEL COM
      - STM32446E EVAL LOW LEVEL Exported Macros
      - STM32446E EVAL LOW LEVEL Exported Functions
    - STM32446E EVAL AUDIO
      - STM32446E EVAL AUDIO Private Types
      - STM32446E EVAL AUDIO Private Defines
      - STM32446E EVAL AUDIO Private Macros
      - STM32446E EVAL AUDIO Private Variables

    - STM32446E EVAL AUDIO Private Function Prototypes
    - STM32446E EVAL AUDIO OUT Private Functions
    - STM32446E EVAL AUDIO IN Private Functions
    - STM32446E EVAL AUDIO Exported Types
    - STM32446E EVAL AUDIO Exported Constants
        - STM32446E EVAL Audio Slot TDM mode
    - STM32446E EVAL AUDIO Exported Variables
    - STM32446E EVAL AUDIO Exported Macros
    - STM32446E EVAL AUDIO OUT Exported Functions
    - STM32446E EVAL AUDIO IN Exported Functions
- STM32446E-EVAL CAMERA
    - STM32446E EVAL CAMERA Private TypesDefinitions
    - STM32446E EVAL CAMERA Private Defines
    - STM32446E EVAL CAMERA Private Macros
    - STM32446E EVAL CAMERA Private Variables
    - STM32446E EVAL CAMERA Private FunctionPrototypes
    - STM32446E EVAL CAMERA Private Functions
    - STM32446E EVAL CAMERA Exported Types
    - STM32446E EVAL CAMERA Exported Constants
    - STM32446E EVAL CAMERA Exported Functions
- STM32446E EVAL EEPROM
    - STM32446E EVAL EEPROM Private Types
    - STM32446E EVAL EEPROM Private Defines
    - STM32446E EVAL EEPROM Private Macros
    - STM32446E EVAL EEPROM Private Variables
    - STM32446E EVAL EEPROM Private Function Prototypes
    - STM32446E EVAL EEPROM Private Functions
    - STM32446E EVAL EEPROM Exported Types
    - STM32446E EVAL EEPROM Exported Constants
    - STM32446E EVAL EEPROM Exported Macros
    - STM32446E EVAL EEPROM Exported Functions
- STM32446E EVAL IO

    - STM32446E EVAL IO Private Types Definitions
    - STM32446E EVAL IO Private Defines
    - STM32446E EVAL IO Private Macros
    - STM32446E EVAL IO Private Variables
    - STM32446E EVAL IO Private Function Prototypes
    - STM32446E EVAL IO Private Functions
    - STM32446E EVAL IO Exported Types
    - STM32446E EVAL IO Exported Constants
    - STM32446E EVAL IO Exported Macro
    - STM32446E EVAL IO Exported Functions
- STM32446E EVAL LCD
    - STM32446E EVAL LCD Private TypesDefinitions
    - STM32446E EVAL LCD Private Defines
    - STM32446E EVAL LCD Private Macros
    - STM32446E EVAL LCD Private Variables
    - STM32446E EVAL LCD Private FunctionPrototypes
    - STM32446E EVAL LCD Private Functions
    - STM32446E EVAL LCD Exported Types
    - STM32446E EVAL LCD Exported Constants
    - STM32446E EVAL LCD Exported Functions
- STM32446E EVAL QSPI
    - STM32446E EVAL QSPI Private Variables
    - STM32446E EVAL QSPI Private Functions
    - STM32446E EVAL QSPI Exported Functions
    - STM32446E EVAL QSPI Exported Constants
    - STM32446E EVAL QSPI Exported Types
- STM32446E EVAL SD
    - STM32446E EVAL SD Private TypesDefinitions
    - STM32446E EVAL SD Private Defines
    - STM32446E EVAL SD Private Macros
    - STM32446E EVAL SD Private Variables
    - STM32446E EVAL SD Private FunctionPrototypes
    - STM32446E EVAL SD Private Functions
    - STM32446E EVAL SD Exported Types
    - STM32446E EVAL SD Exported Constants
    - STM32446E EVAL SD Exported Macro

    - STM32446E EVAL SD Exported Functions
- STM32446E EVAL SDRAM
    - STM32446E EVAL SDRAM Private Types Definitions
    - STM32446E EVAL SDRAM Private Defines
    - STM32446E EVAL SDRAM Private Macros
    - STM32446E EVAL SDRAM Private Variables
    - STM32446E EVAL SDRAM Private Function Prototypes
    - STM32446E EVAL SDRAM Private Functions
    - STM32446E EVAL SDRAM Exported Types
    - STM32446E EVAL SDRAM Exported Constants
    - STM32446E EVAL SDRAM Exported Macro
    - STM32446E EVAL SDRAM Exported Functions
- STM32446E EVAL TS
    - STM32446E EVAL TS Private Types Definitions
    - STM32446E EVAL TS Private Defines
    - STM32446E EVAL TS Private Macros
    - STM32446E EVAL TS Private Variables
    - STM32446E EVAL TS Private Function Prototypes
    - STM32446E EVAL TS Private Functions
    - STM32446E EVAL TS Exported Types
    - STM32446E EVAL TS Exported Constants
    - STM32446E EVAL TS Exported Macros
    - STM32446E EVAL TS Exported Functions

STM32446E_EVAL BSP User Manual

Data Structures

Here are the data structures with brief descriptions:

LCD_CONTROLLER_TypeDef	
LCD_DrawPropTypeDef	
Point	
QSPI_Info	
TS_StateTypeDef	

STM32446E_EVAL BSP User Manual

File List

Here is a list of all files with brief descriptions:

stm32446e_eval.c [code]	This file provides a set of firmware functions to manage LEDs, push-buttons and COM ports available on STM32446E-EVAL evaluation board(MB1045) RevB from STMicroelectronics
stm32446e_eval.h [code]	This file contains definitions for STM32446E_EVAL's LEDs, push-buttons and COM ports hardware resources
stm32446e_eval_audio.c [code]	This file provides the Audio driver for the STM32446E-EVAL evaluation board
stm32446e_eval_audio.h [code]	This file contains the common defines and functions prototypes for the **stm32446e_eval_audio.c** driver
stm32446e_eval_camera.c [code]	This file includes the driver for Camera modules mounted on STM32446E-EVAL evaluation board

stm32446e_eval_camera.h [code]	This file contains the common defines and functions prototypes for the **stm32446e_eval_camera.c** driver
stm32446e_eval_eeprom.c [code]	This file provides a set of functions needed to manage an I2C M24LR64 EEPROM memory. To be able to use this driver, the switch EE_M24LR64 must be defined in your toolchain compiler preprocessor
stm32446e_eval_eeprom.h [code]	This file contains all the functions prototypes for the **stm32446e_eval_eeprom.c** firmware driver
stm32446e_eval_io.c [code]	This file provides a set of functions needed to manage the IO pins on STM32446E-EVAL evaluation board
stm32446e_eval_io.h [code]	This file contains the common defines and functions prototypes for the **stm32446e_eval_io.c** driver
stm32446e_eval_lcd.c [code]	This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM32446E-EVAL evaluation board
stm32446e_eval_lcd.h [code]	This file contains the common defines and functions prototypes for the **stm32446e_eval_lcd.c** driver

stm32446e_eval_qspi.c [code]	This file includes a standard driver for the N25Q256A QSPI memory mounted on STM32446E-EVAL board
stm32446e_eval_qspi.h [code]	This file contains the common defines and functions prototypes for the **stm32446e_eval_qspi.c** driver
stm32446e_eval_sd.c [code]	This file includes the uSD card driver mounted on STM32446E-EVAL evaluation board
stm32446e_eval_sd.h [code]	This file contains the common defines and functions prototypes for the **stm32446e_eval_sd.c** driver
stm32446e_eval_sdram.c [code]	This file includes the SDRAM driver for the MT48LC4M32B2B5-7 memory device mounted on STM32446E-EVAL evaluation board
stm32446e_eval_sdram.h [code]	This file contains the common defines and functions prototypes for the **stm32446e_eval_sdram.c** driver
stm32446e_eval_ts.c [code]	This file provides a set of functions needed to manage the Touch Screen on STM32446E-EVAL evaluation board
stm32446e_eval_ts.h [code]	This file contains the common

	defines and functions prototypes for the **stm32446e_eval_ts.c** driver

STM32446E_EVAL BSP User Manual

Directories

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

- Drivers
  - BSP
    - STM32446E_EVAL

STM32446E_EVAL BSP User Manual

Data Structure Index

L | P | Q | T

L

LCD_CONTROLLER_TypeDef
LCD_DrawPropTypeDef

P

Point

Q

QSPI_Inf

L | P | Q | T

STM32446E_EVAL BSP User Manual

STM32446E EVAL LOW LEVEL Private TypesDefinitions

STM32446E EVAL LOW LEVEL

Data Structures

struct	**LCD_CONTROLLER_TypeDef**

STM32446E_EVAL BSP User Manual

stm32446e_eval.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval.c
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file provides a set of firmware functions to manage LEDs,
00008   *          push-buttons and COM ports available on STM32446E-EVAL evaluation
00009   *          board(MB1045) RevB from STMicroelectronics.
00010   ******************************************************************************
00011   * @attention
00012   *
00013   * <h2><center>&copy; COPYRIGHT(c) 2015 STMicroelectronics</center></h2>
00014   *
00015   * Redistribution and use in source and binary forms, with or without modification,
```

```
00016   * are permitted provided that the followin
g conditions are met:
00017   *   1. Redistributions of source code must
 retain the above copyright notice,
00018   *      this list of conditions and the fol
lowing disclaimer.
00019   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00020   *      this list of conditions and the fol
lowing disclaimer in the documentation
00021   *      and/or other materials provided wit
h the distribution.
00022   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00023   *      may be used to endorse or promote p
roducts derived from this software
00024   *      without specific prior written perm
ission.
00025   *
00026   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00027   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00028   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00029   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00030   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00031   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 PROCUREMENT OF SUBSTITUTE GOODS OR
00032   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00033   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00034   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
```

```
00035   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00036   *
00037   ******************************************
**********************************
00038   */
00039
00040 /* File Info: ------------------------------
-----------------------------------
00041                                       User NOTE
00042
00043    This driver requires the stm32446e_eval_i
o to manage the joystick
00044
00045 ------------------------------------------
-----------------------------------*/
00046
00047 /* Includes --------------------------------
-----------------------------------*/
00048 #include "stm32446e_eval.h"
00049 #if defined(USE_IOEXPANDER)
00050 #include "stm32446e_eval_io.h"
00051 #endif /* USE_IOEXPANDER */
00052
00053
00054 /** @defgroup BSP BSP
00055   * @{
00056   */
00057
00058 /** @defgroup STM32446E_EVAL STM32446E EVAL
00059   * @{
00060   */
00061
00062 /** @defgroup STM32446E_EVAL_LOW_LEVEL STM32
446E EVAL LOW LEVEL
00063   * @{
00064   */
```

```
00065
00066 /** @defgroup STM32446E_EVAL_LOW_LEVEL_Priva
te_TypesDefinitions STM32446E EVAL LOW LEVEL Priva
te TypesDefinitions
00067   * @{
00068   */
00069 typedef struct
00070 {
00071   __IO uint16_t REG;
00072   __IO uint16_t RAM;
00073 }LCD_CONTROLLER_TypeDef;
00074 /**
00075   * @}
00076   */
00077
00078 /** @defgroup STM32446E_EVAL_LOW_LEVEL_Priva
te_Defines STM32446E EVAL LOW LEVEL Private Defines

00079   * @{
00080   */
00081 /**
00082  * @brief STM32446E EVAL BSP Driver version
number V1.1.1
00083    */
00084 #define __STM32446E_EVAL_BSP_VERSION_MAIN
(0x01) /*!< [31:24] main version */
00085 #define __STM32446E_EVAL_BSP_VERSION_SUB1
(0x01) /*!< [23:16] sub1 version */
00086 #define __STM32446E_EVAL_BSP_VERSION_SUB2
(0x01) /*!< [15:8]  sub2 version */
00087 #define __STM32446E_EVAL_BSP_VERSION_RC
(0x00) /*!< [7:0]  release candidate */
00088 #define __STM32446E_EVAL_BSP_VERSION
 ((__STM32446E_EVAL_BSP_VERSION_MAIN << 24)\
00089
 |(__STM32446E_EVAL_BSP_VERSION_SUB1 << 16)\
00090
```

```
 |(__STM32446E_EVAL_BSP_VERSION_SUB2 << 8 )\
00091
 |(__STM32446E_EVAL_BSP_VERSION_RC))
00092
00093 /* compared to F4xG we use BANK1 rather then
 BANK3 since we use FMC_NE1 signal (not FMC_NE3) */
00094 #define FMC_BANK1_BASE  ((uint32_t)(0x600000
00 | 0x00000000))
00095 #define FMC_BANK3_BASE  ((uint32_t)(0x600000
00 | 0x08000000))
00096 #define FMC_BANK1       ((LCD_CONTROLLER_Typ
eDef *) FMC_BANK1_BASE)
00097
00098 /**
00099   * @}
00100   */
00101
00102 /** @defgroup STM32446E_EVAL_LOW_LEVEL_Priva
te_Macros STM32446E EVAL LOW LEVEL Private Macros
00103   * @{
00104   */
00105 /**
00106   * @}
00107   */
00108
00109 /** @defgroup STM32446E_EVAL_LOW_LEVEL_Priva
te_Variables STM32446E EVAL LOW LEVEL Private Vari
ables
00110   * @{
00111   */
00112
```

```
00113 #if defined(USE_IOEXPANDER)
00114 const uint32_t GPIO_PIN[LEDn] = {LED1_PIN,
00115                                  LED2_PIN,
00116                                  LED3_PIN,
00117                                  LED4_PIN};
00118 #else
00119 const uint32_t GPIO_PIN[LEDn] = {LED1_PIN,
00120                                  LED3_PIN};
00121 #endif /* USE_IOEXPANDER */
00122
00123
00124 GPIO_TypeDef* BUTTON_PORT[BUTTONn] = {WAKEUP
_BUTTON_GPIO_PORT,
00125                                       TAMPER
_BUTTON_GPIO_PORT,
00126                                       KEY_BU
TTON_GPIO_PORT};
00127
00128 const uint16_t BUTTON_PIN[BUTTONn] = {WAKEUP
_BUTTON_PIN,
00129                                       TAMPER
_BUTTON_PIN,
00130                                       KEY_BU
TTON_PIN};
00131
00132 const uint16_t BUTTON_IRQn[BUTTONn] = {WAKEU
P_BUTTON_EXTI_IRQn,
00133                                        TAMPE
R_BUTTON_EXTI_IRQn,
00134                                        KEY_B
UTTON_EXTI_IRQn};
00135
00136 USART_TypeDef* COM_USART[COMn] = {EVAL_COM1}
;
00137
00138 GPIO_TypeDef* COM_TX_PORT[COMn] = {EVAL_COM1
_TX_GPIO_PORT};
```

```
00139
00140 GPIO_TypeDef* COM_RX_PORT[COMn] = {EVAL_COM1
_RX_GPIO_PORT};
00141
00142 const uint16_t COM_TX_PIN[COMn] = {EVAL_COM1
_TX_PIN};
00143
00144 const uint16_t COM_RX_PIN[COMn] = {EVAL_COM1
_RX_PIN};
00145
00146 const uint16_t COM_TX_AF[COMn] = {EVAL_COM1_
TX_AF};
00147
00148 const uint16_t COM_RX_AF[COMn] = {EVAL_COM1_
RX_AF};
00149
00150 static FMPI2C_HandleTypeDef hEvalI2c;
00151
00152 /**
00153   * @}
00154   */
00155
00156 /** @defgroup STM32446E_EVAL_LOW_LEVEL_Priva
te_FunctionPrototypes STM32446E EVAL LOW LEVEL Pri
vate FunctionPrototypes
00157   * @{
00158   */
00159 static void     I2Cx_MspInit(void);
00160 static void     I2Cx_Init(void);
00161 #if defined(USE_IOEXPANDER)
00162 static void     I2Cx_Write(uint8_t Addr, uin
t8_t Reg, uint8_t Value);
00163 static uint8_t  I2Cx_Read(uint8_t Addr, uint
8_t Reg);
00164 #endif /* USE_IOEXPANDER */
00165
00166 static HAL_StatusTypeDef I2Cx_ReadMultiple(u
```

```
int8_t Addr, uint16_t Reg, uint16_t MemAddSize, ui
nt8_t *Buffer, uint16_t Length);
00167 static HAL_StatusTypeDef I2Cx_WriteMultiple(
uint8_t Addr, uint16_t Reg, uint16_t MemAddSize, u
int8_t *Buffer, uint16_t Length);
00168 static HAL_StatusTypeDef I2Cx_IsDeviceReady(
uint16_t DevAddress, uint32_t Trials);
00169 static void     I2Cx_Error(uint8_t Addr);
00170
00171 static void     FMC_BANK1_WriteData(uint16_t
 Data);
00172 static void     FMC_BANK1_WriteReg(uint8_t R
eg);
00173 static uint16_t FMC_BANK1_ReadData(void);
00174 static void     FMC_BANK1_Init(void);
00175 static void     FMC_BANK1_MspInit(void);
00176
00177 /* IOExpander IO functions */
00178 #if defined(USE_IOEXPANDER)
00179 void            MFX_IO_Init(void);
00180 void            MFX_IO_DeInit(void);
00181 void            MFX_IO_ITConfig(void);
00182 void            MFX_IO_Delay(uint32_t Delay)
;
00183 void            MFX_IO_Write(uint16_t Addr,
uint8_t Reg, uint8_t Value);
00184 uint8_t         MFX_IO_Read(uint16_t Addr, u
int8_t Reg);
00185 uint16_t        MFX_IO_ReadMultiple(uint16_t
 Addr, uint8_t Reg, uint8_t *Buffer, uint16_t Leng
th);
00186 void            MFX_IO_Wakeup(void);
00187 void            MFX_IO_EnableWakeupPin(void)
;
00188 #endif /* USE_IOEXPANDER */
00189
00190 /* LCD IO functions */
```

```
00191 void              LCD_IO_Init(void);
00192 void              LCD_IO_WriteData(uint16_t Re
gValue);
00193 void              LCD_IO_WriteReg(uint8_t Reg)
;
00194 uint16_t          LCD_IO_ReadData(void);
00195
00196 /* AUDIO IO functions */
00197 void              AUDIO_IO_Init(void);
00198 void              AUDIO_IO_DeInit(void);
00199 void              AUDIO_IO_Write(uint8_t Addr,
 uint16_t Reg, uint16_t Value);
00200 uint16_t          AUDIO_IO_Read(uint8_t Addr,
uint16_t Reg);
00201 void              AUDIO_IO_Delay(uint32_t Dela
y);
00202
00203 /* CAMERA IO functions */
00204 void              CAMERA_IO_Init(void);
00205 void              CAMERA_Delay(uint32_t Delay)
;
00206 void              CAMERA_IO_Write(uint8_t Addr
, uint16_t Reg, uint16_t Value);
00207 uint16_t          CAMERA_IO_Read(uint8_t Addr,
 uint16_t Reg);
00208
00209 /* I2C EEPROM IO function */
00210 void                  EEPROM_IO_Init(void);
00211 HAL_StatusTypeDef     EEPROM_IO_WriteData(uint
16_t DevAddress, uint16_t MemAddress, uint8_t* pBu
ffer, uint32_t BufferSize);
00212 HAL_StatusTypeDef     EEPROM_IO_ReadData(uint1
6_t DevAddress, uint16_t MemAddress, uint8_t* pBuf
fer, uint32_t BufferSize);
00213 HAL_StatusTypeDef     EEPROM_IO_IsDeviceReady(
uint16_t DevAddress, uint32_t Trials);
00214
```

```
00215
00216 /**
00217   * @}
00218   */
00219
00220
00221
00222 /** @defgroup STM32446E_EVAL_LOW_LEVEL_Private_Functions STM32446E EVAL LOW LEVEL Private Functions
00223   * @{
00224   */
00225
00226   /**
00227   * @brief  This method returns the STM32446E EVAL BSP Driver revision
00228   * @retval version: 0xXYZR (8bits for each decimal, R for RC)
00229   */
00230 uint32_t BSP_GetVersion(void)
00231 {
00232   return __STM32446E_EVAL_BSP_VERSION;
00233 }
00234
00235 /**
00236   * @brief  Configures LEDs.
00237   * @param  Led: LED to be configured.
00238   *          This parameter can be one of the following values:
00239   *            @arg  LED1
00240   *            @arg  LED2
00241   *            @arg  LED3
00242   *            @arg  LED4
00243   */
00244 void BSP_LED_Init(Led_TypeDef Led)
00245 {
00246 #if !defined(USE_STM32446E_EVAL_REVA)
```

```
00247   GPIO_InitTypeDef  gpio_init_structure;
00248
00249   /* On RevB led1 and Led3 are on GPIO while
 Led2 and Led4 on Mfx*/
00250   if ((Led == LED1) || (Led == LED3))
00251   {
00252     /* Enable the GPIO_LED clock */
00253     LEDx_GPIO_CLK_ENABLE();
00254
00255     /* Configure the GPIO_LED pin */
00256     gpio_init_structure.Pin = GPIO_PIN[Led];
00257     gpio_init_structure.Mode = GPIO_MODE_OUT
PUT_PP;
00258     gpio_init_structure.Pull = GPIO_PULLUP;
00259     gpio_init_structure.Speed = GPIO_SPEED_H
IGH;
00260
00261     HAL_GPIO_Init(LEDx_GPIO_PORT, &gpio_init
_structure);
00262
00263     /* By default, turn off LED */
00264     HAL_GPIO_WritePin(LEDx_GPIO_PORT, GPIO_P
IN[Led], GPIO_PIN_SET);
00265   }
00266   else  /* Led2 and Led4 */
00267   {
00268 #endif /* !USE_STM32446E_EVAL_REVA */
00269
00270 #if defined(USE_IOEXPANDER)  /* (USE_IOEXPAN
DER always defined for RevA) */
00271     /* Initialize the IO functionalities (Mf
x) */
00272     BSP_IO_Init();
00273     /* GPIO_PIN[Led]  depends on the board r
evision: */
00274     /*  - in case of RevA all leds are on IO
EXPANDER (Mfx)  */
```

```
00275     /*  - in case of RevB just led 2 and led
4 on IOEXPANDER (Mfx) */
00276     BSP_IO_ConfigPin(GPIO_PIN[Led], IO_MODE_
OUTPUT_PP_PU);
00277     BSP_IO_WritePin(GPIO_PIN[Led], BSP_IO_PI
N_SET);
00278 #endif /* USE_IOEXPANDER */
00279
00280 #if !defined(USE_STM32446E_EVAL_REVA)
00281   }
00282 #endif /* !USE_STM32446E_EVAL_REVA */
00283 }
00284
00285
00286 /**
00287   * @brief  DeInit LEDs.
00288   * @param  Led: LED to be configured.
00289   *          This parameter can be one of th
e following values:
00290   *            @arg  LED1
00291   *            @arg  LED2
00292   *            @arg  LED3
00293   *            @arg  LED4
00294   * @note Led DeInit does not disable the GP
IO clock nor disable the Mfx
00295   */
00296 void BSP_LED_DeInit(Led_TypeDef Led)
00297 {
00298 #if !defined(USE_STM32446E_EVAL_REVA)
00299   GPIO_InitTypeDef  gpio_init_structure;
00300
00301   /* On RevB led1 and Led3 are on GPIO while
 Led2 and Led4 on Mfx*/
00302   if ((Led == LED1) || (Led == LED3))
00303   {
00304     /* Turn off LED */
00305     HAL_GPIO_WritePin(LEDx_GPIO_PORT, GPIO_P
```

```
IN[Led], GPIO_PIN_RESET);
00306     /* DeInit the GPIO_LED pin */
00307     gpio_init_structure.Pin = GPIO_PIN[Led];
00308     HAL_GPIO_DeInit(LEDx_GPIO_PORT, gpio_ini
t_structure.Pin);
00309   }
00310   else
00311   {
00312 #endif /* !USE_STM32446E_EVAL_REVA */
00313
00314 #if defined(USE_IOEXPANDER)   /* (USE_IOEXPA
NDER always defined for RevA) */
00315     /* GPIO_PIN[Led]  depends on the board r
evision: */
00316     /*  - in case of RevA all leds are on IO
EXPANDER (Mfx)  */
00317     /*  - in case of RevB just led 2 and led
4 on IOEXPANDER (Mfx) */
00318     BSP_IO_ConfigPin(GPIO_PIN[Led], IO_MODE_
OFF);
00319 #endif /* USE_IOEXPANDER */
00320
00321 #if !defined(USE_STM32446E_EVAL_REVA)
00322   }
00323 #endif /* !USE_STM32446E_EVAL_REVA */
00324 }
00325
00326 /**
00327   * @brief  Turns selected LED On.
00328   * @param  Led: LED to be set on
00329   *          This parameter can be one of th
e following values:
00330   *            @arg  LED1
00331   *            @arg  LED2
00332   *            @arg  LED3
00333   *            @arg  LED4
00334   */
```

```
00335 void BSP_LED_On(Led_TypeDef Led)
00336 {
00337
00338 #if !defined(USE_STM32446E_EVAL_REVA)
00339   /* On RevB led1 and Led3 are on GPIO while
 Led2 and Led4 on Mfx*/
00340   if ((Led == LED1) || (Led == LED3))
00341   {
00342     HAL_GPIO_WritePin(LEDx_GPIO_PORT, GPIO_
PIN[Led], GPIO_PIN_RESET);
00343   }
00344   else
00345   {
00346 #endif /* !USE_STM32446E_EVAL_REVA */
00347
00348 #if defined(USE_IOEXPANDER) /* (USE_IOEXPAND
ER always defined for RevA) */
00349     /* GPIO_PIN[Led]  depends on the board r
evision: */
00350     /*  - in case of RevA all leds are on IO
EXPANDER (Mfx)  */
00351     /*  - in case of RevB just led 2 and led
4 on IOEXPANDER (Mfx) */
00352     BSP_IO_WritePin(GPIO_PIN[Led], BSP_IO_PI
N_RESET);
00353 #endif /* USE_IOEXPANDER */
00354
00355 #if !defined(USE_STM32446E_EVAL_REVA)
00356   }
00357 #endif /* !USE_STM32446E_EVAL_REVA */
00358 }
00359
00360 /**
00361   * @brief  Turns selected LED Off.
00362   * @param  Led: LED to be set off
00363   *          This parameter can be one of th
e following values:
```

```
00364   *             @arg  LED1
00365   *             @arg  LED2
00366   *             @arg  LED3
00367   *             @arg  LED4
00368   */
00369 void BSP_LED_Off(Led_TypeDef Led)
00370 {
00371
00372 #if !defined(USE_STM32446E_EVAL_REVA)
00373   /* On RevB led1 and Led3 are on GPIO while
 Led2 and Led4 on Mfx*/
00374   if ((Led == LED1) || (Led == LED3))
00375   {
00376     HAL_GPIO_WritePin(LEDx_GPIO_PORT, GPIO_
PIN[Led], GPIO_PIN_SET);
00377   }
00378   else
00379   {
00380 #endif /* !USE_STM32446E_EVAL_REVA */
00381
00382 #if defined(USE_IOEXPANDER) /* (USE_IOEXPAND
ER always defined for RevA) */
00383     /* GPIO_PIN[Led]  depends on the board r
evision: */
00384     /*  - in case of RevA all leds are on IO
EXPANDER (Mfx)  */
00385     /*  - in case of RevB just led 2 and led
4 on IOEXPANDER (Mfx) */
00386     BSP_IO_WritePin(GPIO_PIN[Led], BSP_IO_PI
N_SET);
00387 #endif /* USE_IOEXPANDER */
00388
00389 #if !defined(USE_STM32446E_EVAL_REVA)
00390   }
00391 #endif /* !USE_STM32446E_EVAL_REVA */
00392
00393 }
```

```
00394
00395 /**
00396   * @brief  Toggles the selected LED.
00397   * @param  Led: LED to be toggled
00398   *          This parameter can be one of th
e following values:
00399   *            @arg  LED1
00400   *            @arg  LED2
00401   *            @arg  LED3
00402   *            @arg  LED4
00403   */
00404 void BSP_LED_Toggle(Led_TypeDef Led)
00405 {
00406
00407 #if !defined(USE_STM32446E_EVAL_REVA)
00408   /* On RevB led1 and Led3 are on GPIO while
 Led2 and Led4 on Mfx*/
00409   if ((Led == LED1) || (Led == LED3))
00410   {
00411     HAL_GPIO_TogglePin(LEDx_GPIO_PORT, GPIO
_PIN[Led]);
00412   }
00413   else
00414   {
00415 #endif /* !USE_STM32446E_EVAL_REVA */
00416
00417 #if defined(USE_IOEXPANDER) /* (USE_IOEXPAND
ER always defined for RevA) */
00418     /* GPIO_PIN[Led]  depends on the board r
evision: */
00419     /*  - in case of RevA all leds are on IO
EXPANDER (Mfx)  */
00420     /*  - in case of RevB just led 2 and led
4 on IOEXPANDER (Mfx) */
00421     BSP_IO_TogglePin(GPIO_PIN[Led]);
00422 #endif /* USE_IOEXPANDER */
00423
```

```
00424 #if !defined(USE_STM32446E_EVAL_REVA)
00425   }
00426 #endif /* !USE_STM32446E_EVAL_REVA */
00427 }
00428
00429 /**
00430   * @brief  Configures button GPIO and EXTI
Line.
00431   * @param  Button: Button to be configured
00432   *          This parameter can be one of th
e following values:
00433   *            @arg  BUTTON_WAKEUP: Wakeup P
ush Button
00434   *            @arg  BUTTON_TAMPER: Tamper P
ush Button
00435   *            @arg  BUTTON_KEY: Key Push Bu
tton
00436   * @param  ButtonMode: Button mode
00437   *          This parameter can be one of th
e following values:
00438   *            @arg  BUTTON_MODE_GPIO: Butto
n will be used as simple IO
00439   *            @arg  BUTTON_MODE_EXTI: Butto
n will be connected to EXTI line
00440   *                                    with
interrupt generation capability
00441   * @note On STM32446E-EVAL evaluation board
, the three buttons (Wakeup, Tamper
00442   *       and key buttons) are mapped on the
 same push button named "Wakeup/Tamper"
00443   *       on the board serigraphy.
00444   */
00445 void BSP_PB_Init(Button_TypeDef Button, Butt
onMode_TypeDef ButtonMode)
00446 {
00447   GPIO_InitTypeDef gpio_init_structure;
00448
```

```
00449   /* Enable the BUTTON clock */
00450   BUTTONx_GPIO_CLK_ENABLE(Button);
00451
00452   if(ButtonMode == BUTTON_MODE_GPIO)
00453   {
00454     /* Configure Button pin as input */
00455     gpio_init_structure.Pin = BUTTON_PIN[But
ton];
00456     gpio_init_structure.Mode = GPIO_MODE_INP
UT;
00457     gpio_init_structure.Pull = GPIO_NOPULL;
00458     gpio_init_structure.Speed = GPIO_SPEED_F
AST;
00459     HAL_GPIO_Init(BUTTON_PORT[Button], &gpio
_init_structure);
00460   }
00461
00462   if(ButtonMode == BUTTON_MODE_EXTI)
00463   {
00464     /* Configure Button pin as input with Ex
ternal interrupt */
00465     gpio_init_structure.Pin = BUTTON_PIN[But
ton];
00466     gpio_init_structure.Pull = GPIO_NOPULL;
00467     gpio_init_structure.Speed = GPIO_SPEED_F
AST;
00468
00469     if(Button != BUTTON_WAKEUP)
00470     {
00471       gpio_init_structure.Mode = GPIO_MODE_I
T_FALLING;
00472     }
00473     else
00474     {
00475       gpio_init_structure.Mode = GPIO_MODE_I
T_RISING;
00476     }
```

```
00477
00478     HAL_GPIO_Init(BUTTON_PORT[Button], &gpio
_init_structure);
00479
00480     /* Enable and set Button EXTI Interrupt
to the lowest priority */
00481     HAL_NVIC_SetPriority((IRQn_Type)(BUTTON_
IRQn[Button]), 0x0F, 0x00);
00482     HAL_NVIC_EnableIRQ((IRQn_Type)(BUTTON_IR
Qn[Button]));
00483   }
00484 }
00485
00486 /**
00487   * @brief  Push Button DeInit.
00488   * @param  Button: Button to be configured
00489   *          This parameter can be one of th
e following values:
00490   *            @arg  BUTTON_WAKEUP: Wakeup P
ush Button
00491   *            @arg  BUTTON_TAMPER: Tamper P
ush Button
00492   *            @arg  BUTTON_KEY: Key Push Bu
tton
00493   * @note On STM32446E-EVAL evaluation board
, the three buttons (Wakeup, Tamper
00494   *       and key buttons) are mapped on the
 same push button nammed "Wakeup/Tamper"
00495   *       on the board serigraphy.
00496   * @note PB DeInit does not disable the GPI
O clock
00497   */
00498 void BSP_PB_DeInit(Button_TypeDef Button)
00499 {
00500     GPIO_InitTypeDef gpio_init_structure;
00501
00502     gpio_init_structure.Pin = BUTTON_PIN[But
```

```
ton];
00503     HAL_NVIC_DisableIRQ((IRQn_Type)(BUTTON_I
RQn[Button]));
00504     HAL_GPIO_DeInit(BUTTON_PORT[Button], gpi
o_init_structure.Pin);
00505 }
00506
00507
00508 /**
00509   * @brief  Returns the selected button stat
e.
00510   * @param  Button: Button to be checked
00511   *          This parameter can be one of th
e following values:
00512   *            @arg  BUTTON_WAKEUP: Wakeup P
ush Button
00513   *            @arg  BUTTON_TAMPER: Tamper P
ush Button
00514   *            @arg  BUTTON_KEY: Key Push Bu
tton
00515   * @note On STM32446E-EVAL evaluation board
, the three buttons (Wakeup, Tamper
00516   *       and key buttons) are mapped on the
 same push button nammed "Wakeup/Tamper"
00517   *       on the board serigraphy.
00518   * @retval The Button GPIO pin value
00519   */
00520 uint32_t BSP_PB_GetState(Button_TypeDef Butt
on)
00521 {
00522   return HAL_GPIO_ReadPin(BUTTON_PORT[Button
], BUTTON_PIN[Button]);
00523 }
00524
00525 /**
00526   * @brief  Configures COM port.
00527   * @param  COM: COM port to be configured.
```

```
00528   *           This parameter can be one of th
e following values:
00529   *             @arg  COM1
00530   *             @arg  COM2
00531   * @param  huart: Pointer to a UART_HandleT
ypeDef structure that contains the
00532   *                configuration information
 for the specified USART peripheral.
00533   */
00534 void BSP_COM_Init(COM_TypeDef COM, UART_Hand
leTypeDef *huart)
00535 {
00536   GPIO_InitTypeDef gpio_init_structure;
00537
00538   /* Enable GPIO clock */
00539   EVAL_COMx_TX_GPIO_CLK_ENABLE(COM);
00540   EVAL_COMx_RX_GPIO_CLK_ENABLE(COM);
00541
00542   /* Enable USART clock */
00543   EVAL_COMx_CLK_ENABLE(COM);
00544
00545   /* Configure USART Tx as alternate functio
n */
00546   gpio_init_structure.Pin = COM_TX_PIN[COM];
00547   gpio_init_structure.Mode = GPIO_MODE_AF_PP
;
00548   gpio_init_structure.Speed = GPIO_SPEED_FAS
T;
00549   gpio_init_structure.Pull = GPIO_PULLUP;
00550   gpio_init_structure.Alternate = COM_TX_AF[
COM];
00551   HAL_GPIO_Init(COM_TX_PORT[COM], &gpio_init
_structure);
00552
00553   /* Configure USART Rx as alternate functio
n */
00554   gpio_init_structure.Pin = COM_RX_PIN[COM];
```

```
00555   gpio_init_structure.Mode = GPIO_MODE_AF_PP
;
00556   gpio_init_structure.Alternate = COM_RX_AF[
COM];
00557   HAL_GPIO_Init(COM_RX_PORT[COM], &gpio_init
_structure);
00558
00559   /* USART configuration */
00560   huart->Instance = COM_USART[COM];
00561   HAL_UART_Init(huart);
00562 }
00563
00564 /**
00565   * @brief  DeInit COM port.
00566   * @param  COM: COM port to be configured.
00567   *          This parameter can be one of th
e following values:
00568   *            @arg  COM1
00569   *            @arg  COM2
00570   * @param  huart: Pointer to a UART_HandleT
ypeDef structure that contains the
00571   *                configuration information
 for the specified USART peripheral.
00572   */
00573 void BSP_COM_DeInit(COM_TypeDef COM, UART_Ha
ndleTypeDef *huart)
00574 {
00575   /* USART configuration */
00576   huart->Instance = COM_USART[COM];
00577   HAL_UART_DeInit(huart);
00578
00579   /* Enable USART clock */
00580   EVAL_COMx_CLK_DISABLE(COM);
00581
00582   /* DeInit GPIO pins can be done in the app
lication
00583      (by surcharging this __weak function) */
```

```
00584
00585   /* GPIO pins clock, FMC clock and DMA cloc
k can be shut down in the application
00586      by surcharging this __weak function */
00587 }
00588
00589 #if defined(USE_IOEXPANDER)
00590
00591 /**
00592   * @brief  Configures joystick GPIO and EXT
I modes.
00593   * @param  JoyMode: Button mode.
00594   *           This parameter can be one of th
e following values:
00595   *             @arg  JOY_MODE_GPIO: Joystick
 pins will be used as simple IOs
00596   *             @arg  JOY_MODE_EXTI: Joystick
 pins will be connected to EXTI line
00597   *                                  with int
errupt generation capability
00598   * @retval IO_OK: if all initializations ar
e OK. Other value if error.
00599   */
00600 uint8_t BSP_JOY_Init(JOYMode_TypeDef JoyMode
)
00601 {
00602   uint8_t ret = 0;
00603
00604   /* Initialize the IO functionalities */
00605   ret = BSP_IO_Init();
00606
00607   /* Configure joystick pins in IT mode */
00608   if(JoyMode == JOY_MODE_EXTI)
00609   {
00610     /* Configure IO interrupt acquisition mo
de */
```

```
00611     BSP_IO_ConfigPin(JOY_ALL_PINS, IO_MODE_I
T_FALLING_EDGE_PU);
00612   }
00613   else
00614   {
00615     BSP_IO_ConfigPin(JOY_ALL_PINS, IO_MODE_I
NPUT_PU);
00616   }
00617
00618   return ret;
00619 }
00620
00621
00622 /**
00623   * @brief  DeInit joystick GPIOs.
00624   * @note   JOY DeInit does not disable the
Mfx, just set the Mfx pins in Off mode
00625   */
00626 void BSP_JOY_DeInit(void)
00627 {
00628     BSP_IO_ConfigPin(JOY_ALL_PINS, IO_MODE_O
FF);
00629 }
00630
00631 /**
00632   * @brief  Returns the current joystick sta
tus.
00633   * @retval Code of the joystick key pressed
00634   *          This code can be one of the fol
lowing values:
00635   *            @arg  JOY_NONE
00636   *            @arg  JOY_SEL
00637   *            @arg  JOY_DOWN
00638   *            @arg  JOY_LEFT
00639   *            @arg  JOY_RIGHT
00640   *            @arg  JOY_UP
00641   */
```

```
00642 JOYState_TypeDef BSP_JOY_GetState(void)
00643 {
00644   uint16_t pin_status = 0;
00645
00646   /* Read the status joystick pins */
00647   pin_status = BSP_IO_ReadPin(JOY_ALL_PINS);
00648
00649   /* Check the pressed keys */
00650   if((pin_status & JOY_NONE_PIN) == JOY_NONE
)
00651   {
00652     return(JOYState_TypeDef) JOY_NONE;
00653   }
00654   else if(!(pin_status & JOY_SEL_PIN))
00655   {
00656     return(JOYState_TypeDef) JOY_SEL;
00657   }
00658   else if(!(pin_status & JOY_DOWN_PIN))
00659   {
00660     return(JOYState_TypeDef) JOY_DOWN;
00661   }
00662   else if(!(pin_status & JOY_LEFT_PIN))
00663   {
00664     return(JOYState_TypeDef) JOY_LEFT;
00665   }
00666   else if(!(pin_status & JOY_RIGHT_PIN))
00667   {
00668     return(JOYState_TypeDef) JOY_RIGHT;
00669   }
00670   else if(!(pin_status & JOY_UP_PIN))
00671   {
00672     return(JOYState_TypeDef) JOY_UP;
00673   }
00674   else
00675   {
00676     return(JOYState_TypeDef) JOY_NONE;
00677   }
```

```
}

#endif /* USE_IOEXPANDER */


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

/****************************** I2C Routines ******************************/
/**
  * @brief  Initializes I2C MSP.
  */
static void I2Cx_MspInit(void)
{
  GPIO_InitTypeDef  gpio_init_structure;

  /*** Configure the GPIOs ***/
  /* Enable GPIO clock */
  EVAL_I2Cx_SCL_SDA_GPIO_CLK_ENABLE();

  /* Configure I2C Tx as alternate function */
  gpio_init_structure.Pin = EVAL_I2Cx_SCL_PIN;
  gpio_init_structure.Mode = GPIO_MODE_AF_OD;
  gpio_init_structure.Pull = GPIO_NOPULL;
  gpio_init_structure.Speed = GPIO_SPEED_FAST;
  gpio_init_structure.Alternate = EVAL_I2Cx_SCL_SDA_AF;
  HAL_GPIO_Init(EVAL_I2Cx_SCL_SDA_GPIO_PORT, &gpio_init_structure);
```

```
00706
00707   /* Configure I2C Rx as alternate function
*/
00708   gpio_init_structure.Pin = EVAL_I2Cx_SDA_PIN
;
00709   HAL_GPIO_Init(EVAL_I2Cx_SCL_SDA_GPIO_PORT,
 &gpio_init_structure);
00710
00711   /*** Configure the I2C peripheral ***/
00712   /* Enable I2C clock */
00713   EVAL_I2Cx_CLK_ENABLE();
00714
00715   /* Force the I2C peripheral clock reset */

00716   EVAL_I2Cx_FORCE_RESET();
00717
00718   /* Release the I2C peripheral clock reset
*/
00719   EVAL_I2Cx_RELEASE_RESET();
00720
00721   /* Enable and set I2Cx Interrupt to a lowe
r priority */
00722   HAL_NVIC_SetPriority(EVAL_I2Cx_EV_IRQn, 0x
05, 0);
00723   HAL_NVIC_EnableIRQ(EVAL_I2Cx_EV_IRQn);
00724
00725   /* Enable and set I2Cx Interrupt to a lowe
r priority */
00726   HAL_NVIC_SetPriority(EVAL_I2Cx_ER_IRQn, 0x
05, 0);
00727   HAL_NVIC_EnableIRQ(EVAL_I2Cx_ER_IRQn);
00728 }
00729
00730 /**
00731   * @brief  Initializes I2C HAL.
00732   */
00733 static void I2Cx_Init(void)
```

```
00734 {
00735   if(HAL_FMPI2C_GetState(&hEvalI2c) == HAL_F
MPI2C_STATE_RESET)
00736   {
00737     hEvalI2c.Instance              = EVAL_I2
Cx;
00738     hEvalI2c.Init.Timing           = EVAL_I2
Cx_TIMING;
00739     hEvalI2c.Init.OwnAddress1     = 0;
00740     hEvalI2c.Init.AddressingMode  = FMPI2C_A
DDRESSINGMODE_7BIT;
00741     hEvalI2c.Init.DualAddressMode = FMPI2C_D
UALADDRESS_DISABLE;
00742     hEvalI2c.Init.OwnAddress2     = 0;
00743     hEvalI2c.Init.GeneralCallMode = FMPI2C_G
ENERALCALL_DISABLE;
00744     hEvalI2c.Init.NoStretchMode   = FMPI2C_N
OSTRETCH_DISABLE;
00745
00746     /* Init the I2C */
00747     I2Cx_MspInit();
00748     HAL_FMPI2C_Init(&hEvalI2c);
00749   }
00750 }
00751
00752 #if defined(USE_IOEXPANDER)
00753
00754
00755 /**
00756   * @brief  Writes a single data.
00757   * @param  Addr: I2C address
00758   * @param  Reg: Register address
00759   * @param  Value: Data to be written
00760   */
00761 static void I2Cx_Write(uint8_t Addr, uint8_t
 Reg, uint8_t Value)
00762 {
```

```
00763   HAL_StatusTypeDef status = HAL_OK;
00764
00765   status = HAL_FMPI2C_Mem_Write(&hEvalI2c, A
ddr, (uint16_t)Reg, FMPI2C_MEMADD_SIZE_8BIT, &Valu
e, 1, 100);
00766
00767   /* Check the communication status */
00768   if(status != HAL_OK)
00769   {
00770     /* Execute user timeout callback */
00771     I2Cx_Error(Addr);
00772   }
00773 }
00774
00775 /**
00776   * @brief  Reads a single data.
00777   * @param  Addr: I2C address
00778   * @param  Reg: Register address
00779   * @retval Read data
00780   */
00781 static uint8_t I2Cx_Read(uint8_t Addr, uint8
_t Reg)
00782 {
00783   HAL_StatusTypeDef status = HAL_OK;
00784   uint8_t Value = 0;
00785
00786   status = HAL_FMPI2C_Mem_Read(&hEvalI2c, Ad
dr, Reg, FMPI2C_MEMADD_SIZE_8BIT, &Value, 1, 1000)
;
00787
00788   /* Check the communication status */
00789   if(status != HAL_OK)
00790   {
00791     /* Execute user timeout callback */
00792     I2Cx_Error(Addr);
00793   }
00794   return Value;
```

```
00795 }
00796
00797 #endif /* USE_IOEXPANDER */
00798
00799 /**
00800   * @brief  Reads multiple data.
00801   * @param  Addr: I2C address
00802   * @param  Reg: Reg address
00803   * @param  MemAddSize: address size
00804   * @param  Buffer: Pointer to data buffer
00805   * @param  Length: Length of the data
00806   * @retval Number of read data
00807   */
00808 static HAL_StatusTypeDef I2Cx_ReadMultiple(u
int8_t Addr, uint16_t Reg, uint16_t MemAddSize, ui
nt8_t *Buffer, uint16_t Length)
00809 {
00810   HAL_StatusTypeDef status = HAL_OK;
00811
00812   status = HAL_FMPI2C_Mem_Read(&hEvalI2c, Ad
dr, (uint16_t)Reg, MemAddSize, Buffer, Length, 100
0);
00813
00814   /* Check the communication status */
00815   if(status != HAL_OK)
00816   {
00817     /* I2C error occurred */
00818     I2Cx_Error(Addr);
00819   }
00820   return status;
00821 }
00822
00823 /**
00824   * @brief  Writes a value in a register of
the device through BUS in using DMA mode.
00825   * @param  Addr: Device address on BUS Bus.
```

```
00826   * @param  Reg: The target register address
 to write
00827   * @param  MemAddSize: address size
00828   * @param  Buffer: The target register valu
e to be written
00829   * @param  Length: buffer size to be written

00830   * @retval HAL status
00831   */
00832 static HAL_StatusTypeDef I2Cx_WriteMultiple(
uint8_t Addr, uint16_t Reg, uint16_t MemAddSize, u
int8_t *Buffer, uint16_t Length)
00833 {
00834   HAL_StatusTypeDef status = HAL_OK;
00835
00836   status = HAL_FMPI2C_Mem_Write(&hEvalI2c, A
ddr, (uint16_t)Reg, MemAddSize, Buffer, Length, 10
00);
00837
00838   /* Check the communication status */
00839   if(status != HAL_OK)
00840   {
00841     /* Re-Initialize the I2C Bus */
00842     I2Cx_Error(Addr);
00843   }
00844   return status;
00845 }
00846
00847 /**
00848   * @brief  Checks if target device is ready
 for communication.
00849   * @note   This function is used with Memor
y devices
00850   * @param  DevAddress: Target device address

00851   * @param  Trials: Number of trials
00852   * @retval HAL status
```

```
00853   */
00854 static HAL_StatusTypeDef I2Cx_IsDeviceReady(
uint16_t DevAddress, uint32_t Trials)
00855 {
00856   return (HAL_FMPI2C_IsDeviceReady(&hEvalI2c
, DevAddress, Trials, 1000));
00857 }
00858
00859 /**
00860   * @brief  Manages error callback by re-ini
tializing I2C.
00861   * @param  Addr: I2C Address
00862   */
00863 static void I2Cx_Error(uint8_t Addr)
00864 {
00865   /* De-initialize the I2C communication bus
 */
00866   HAL_FMPI2C_DeInit(&hEvalI2c);
00867
00868   /* Re-Initialize the I2C communication bus
 */
00869   I2Cx_Init();
00870 }
00871
00872 /****************************** FMC Routines **
**********************************/
00873 /**
00874   * @brief  Initializes FMC_BANK1 MSP.
00875   */
00876 static void FMC_BANK1_MspInit(void)
00877 {
00878   GPIO_InitTypeDef gpio_init_structure;
00879
00880   /* Enable FMC clock */
00881   __HAL_RCC_FMC_CLK_ENABLE();
00882
00883   /* Enable GPIOs clock */
```

```
00884   __HAL_RCC_GPIOD_CLK_ENABLE();
00885   __HAL_RCC_GPIOE_CLK_ENABLE();
00886   __HAL_RCC_GPIOF_CLK_ENABLE();
00887
00888   /* Common GPIO configuration */
00889   gpio_init_structure.Mode      = GPIO_MODE_
AF_PP;
00890   gpio_init_structure.Pull      = GPIO_PULLU
P;
00891   gpio_init_structure.Speed     = GPIO_SPEED
_HIGH;
00892   gpio_init_structure.Alternate = GPIO_AF12_
FMC;
00893
00894   /* GPIOD configuration */ /* GPIO_PIN_7 is
  FMC_NE1 */
00895   gpio_init_structure.Pin   = GPIO_PIN_0 | G
PIO_PIN_1 | GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_8 |
\
00896                               GPIO_PIN_9 | G
PIO_PIN_10 | GPIO_PIN_14 | GPIO_PIN_15 | GPIO_PIN_
7;
00897
00898   HAL_GPIO_Init(GPIOD, &gpio_init_structure)
;
00899
00900   /* GPIOE configuration */
00901   gpio_init_structure.Pin   = GPIO_PIN_0 | G
PIO_PIN_1 | GPIO_PIN_3| GPIO_PIN_4 | GPIO_PIN_7
  |\
00902                               GPIO_PIN_8 | G
PIO_PIN_9 | GPIO_PIN_10 | GPIO_PIN_11 | GPIO_PIN_1
2 |\
00903                               GPIO_PIN_13 |
GPIO_PIN_14 | GPIO_PIN_15;
00904   HAL_GPIO_Init(GPIOE, &gpio_init_structure)
;
```

```
00905
00906   /* GPIOF configuration */
00907   gpio_init_structure.Pin   = GPIO_PIN_0 ;
00908   HAL_GPIO_Init(GPIOF, &gpio_init_structure)
;
00909 }
00910
00911
00912 /**
00913   * @brief  Initializes LCD IO.
00914   */
00915 static void FMC_BANK1_Init(void)
00916 {
00917   SRAM_HandleTypeDef hsram;
00918   FMC_NORSRAM_TimingTypeDef sram_timing;
00919
00920   /*** Configure the SRAM Bank 1 ***/
00921   /* Configure IPs */
00922   hsram.Instance  = FMC_NORSRAM_DEVICE;
00923   hsram.Extended  = FMC_NORSRAM_EXTENDED_DEV
ICE;
00924
00925   sram_timing.AddressSetupTime      = 5;
00926   sram_timing.AddressHoldTime       = 1;
00927   sram_timing.DataSetupTime         = 9;
00928   sram_timing.BusTurnAroundDuration = 0;
00929   sram_timing.CLKDivision           = 2;
00930   sram_timing.DataLatency           = 2;
00931   sram_timing.AccessMode            = FMC_AC
CESS_MODE_A;
00932
00933   hsram.Init.NSBank             = FMC_NORSRA
M_BANK1;
00934   hsram.Init.DataAddressMux     = FMC_DATA_A
DDRESS_MUX_DISABLE;
00935   hsram.Init.MemoryType         = FMC_MEMORY
_TYPE_SRAM;
```

```
00936   hsram.Init.MemoryDataWidth    = FMC_NORSRA
M_MEM_BUS_WIDTH_16;
00937   hsram.Init.BurstAccessMode    = FMC_BURST_
ACCESS_MODE_DISABLE;
00938   hsram.Init.WaitSignalPolarity = FMC_WAIT_S
IGNAL_POLARITY_LOW;
00939   hsram.Init.WrapMode           = FMC_WRAP_M
ODE_DISABLE;
00940   hsram.Init.WaitSignalActive   = FMC_WAIT_T
IMING_BEFORE_WS;
00941   hsram.Init.WriteOperation     = FMC_WRITE_
OPERATION_ENABLE;
00942   hsram.Init.WaitSignal         = FMC_WAIT_S
IGNAL_DISABLE;
00943   hsram.Init.ExtendedMode       = FMC_EXTEND
ED_MODE_DISABLE;
00944   hsram.Init.AsynchronousWait   = FMC_ASYNCH
RONOUS_WAIT_DISABLE;
00945   hsram.Init.WriteBurst         = FMC_WRITE_
BURST_DISABLE;
00946   hsram.Init.ContinuousClock    = FMC_CONTIN
UOUS_CLOCK_SYNC_ONLY;
00947   hsram.Init.WriteFifo          = FMC_WRITE_
FIFO_DISABLE;
00948   hsram.Init.PageSize           = FMC_PAGE_S
IZE_NONE;
00949
00950   /* Initialize the SRAM controller */
00951   FMC_BANK1_MspInit();
00952   HAL_SRAM_Init(&hsram, &sram_timing, &sram_
timing);
00953 }
00954
00955
00956 /**
00957   * @brief  Writes register value.
00958   * @param  Data: Data to be written
```

```
  */
static void FMC_BANK1_WriteData(uint16_t Data)
{
  /* Write 16-bit Reg */
  FMC_BANK1->RAM = Data;
}

/**
  * @brief  Writes register address.
  * @param  Reg: Register to be written
  */
static void FMC_BANK1_WriteReg(uint8_t Reg)
{
  /* Write 16-bit Index, then write register */
  FMC_BANK1->REG = Reg;
}

/**
  * @brief  Reads register value.
  * @retval Read value
  */
static uint16_t FMC_BANK1_ReadData(void)
{
  return FMC_BANK1->RAM;
}

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

#if defined(USE_IOEXPANDER)

/******************************** LINK MFX
```

```
**********************************/
00992
00993 /**
00994   * @brief  Initializes MFX low level.
00995   */
00996 void MFX_IO_Init(void)
00997 {
00998   I2Cx_Init();
00999 }
01000
01001 /**
01002   * @brief  DeInitializes MFX low level.
01003   */
01004 void MFX_IO_DeInit(void)
01005 {
01006 }
01007
01008 /**
01009   * @brief  Configures MFX low level interru
pt.
01010   */
01011 void MFX_IO_ITConfig(void)
01012 {
01013   GPIO_InitTypeDef  gpio_init_structure;
01014
01015   /* Enable the GPIO EXTI clock */
01016   __HAL_RCC_GPIOA_CLK_ENABLE();
01017   __HAL_RCC_SYSCFG_CLK_ENABLE();
01018
01019   /* PA0 is the MFX_OUT_IRQ (often used for
EXTI_WKUP) */
01020   gpio_init_structure.Pin   = GPIO_PIN_0;
01021   gpio_init_structure.Pull  = GPIO_NOPULL;
01022   gpio_init_structure.Speed = GPIO_SPEED_LOW
;
01023   gpio_init_structure.Mode  = GPIO_MODE_IT_R
ISING;
```

```
01024   HAL_GPIO_Init(GPIOA, &gpio_init_structure)
;
01025
01026   /* Enable and set GPIO EXTI Interrupt to t
he lowest priority */
01027   HAL_NVIC_SetPriority((IRQn_Type)(EXTI0_IRQ
n), 0x0F, 0x0F);
01028   HAL_NVIC_EnableIRQ((IRQn_Type)(EXTI0_IRQn)
);
01029 }
01030
01031 /**
01032   * @brief  MFX writes single data.
01033   * @param  Addr: I2C address
01034   * @param  Reg: Register address
01035   * @param  Value: Data to be written
01036   */
01037 void MFX_IO_Write(uint16_t Addr, uint8_t Reg
, uint8_t Value)
01038 {
01039   I2Cx_Write((uint8_t) Addr, Reg, Value);
01040 }
01041
01042 /**
01043   * @brief  MFX reads single data.
01044   * @param  Addr: I2C address
01045   * @param  Reg: Register address
01046   * @retval Read data
01047   */
01048 uint8_t MFX_IO_Read(uint16_t Addr, uint8_t R
eg)
01049 {
01050   return I2Cx_Read((uint8_t) Addr, Reg);
01051 }
01052
01053 /**
01054   * @brief  MFX reads multiple data.
```

```
01055   * @param  Addr: I2C address
01056   * @param  Reg: Register address
01057   * @param  Buffer: Pointer to data buffer
01058   * @param  Length: Length of the data
01059   * @retval Number of read data
01060   */
01061 uint16_t MFX_IO_ReadMultiple(uint16_t Addr,
uint8_t Reg, uint8_t *Buffer, uint16_t Length)
01062 {
01063  return I2Cx_ReadMultiple((uint8_t) Addr, (u
int16_t)Reg, FMPI2C_MEMADD_SIZE_8BIT, Buffer, Leng
th);
01064 }
01065
01066 /**
01067   * @brief  MFX delay
01068   * @param  Delay: Delay in ms
01069   */
01070 void MFX_IO_Delay(uint32_t Delay)
01071 {
01072   HAL_Delay(Delay);
01073 }
01074
01075 /**
01076   * @brief  Used by Lx family but requested
for MFX component compatibility.
01077   */
01078 void MFX_IO_Wakeup(void)
01079 {
01080 }
01081
01082 /**
01083   * @brief  Used by Lx family but requested
for MXF component compatibility.
01084   */
01085 void MFX_IO_EnableWakeupPin(void)
01086 {
```

```
01087 }
01088
01089 #endif /* USE_IOEXPANDER */
01090
01091 /********************************** LINK LCD
**********************************/
01092
01093 /**
01094   * @brief  Initializes LCD low level.
01095   */
01096 void LCD_IO_Init(void)
01097 {
01098     FMC_BANK1_Init();
01099 }
01100
01101 /**
01102   * @brief  Writes data on LCD data register.

01103   * @param  RegValue: Data to be written
01104   */
01105 void LCD_IO_WriteData(uint16_t RegValue)
01106 {
01107   /* Write 16-bit Reg */
01108   FMC_BANK1_WriteData(RegValue);
01109 }
01110
01111 /**
01112   * @brief  Writes register on LCD register.
01113   * @param  Reg: Register to be written
01114   */
01115 void LCD_IO_WriteReg(uint8_t Reg)
01116 {
01117   /* Write 16-bit Index, then Write Reg */
01118   FMC_BANK1_WriteReg(Reg);
01119 }
01120
01121 /**
```

```
01122   * @brief  Reads data from LCD data registe
r.
01123   * @retval Read data.
01124   */
01125 uint16_t LCD_IO_ReadData(void)
01126 {
01127   return FMC_BANK1_ReadData();
01128 }
01129
01130 /********************************* LINK AUDI
O *********************************/
01131
01132 /**
01133   * @brief  Initializes Audio low level.
01134   */
01135 void AUDIO_IO_Init(void)
01136 {
01137   I2Cx_Init();
01138 }
01139
01140 /**
01141   * @brief  DeInitializes Audio low level.
01142   */
01143 void AUDIO_IO_DeInit(void)
01144 {
01145
01146 }
01147
01148 /**
01149   * @brief  Writes a single data.
01150   * @param  Addr: I2C address
01151   * @param  Reg: Reg address
01152   * @param  Value: Data to be written
01153   */
01154 void AUDIO_IO_Write(uint8_t Addr, uint16_t R
eg, uint16_t Value)
01155 {
```

```
01156   uint16_t tmp = Value;
01157
01158   Value = ((uint16_t)(tmp >> 8) & 0x00FF);
01159
01160   Value |= ((uint16_t)(tmp << 8)& 0xFF00);
01161
01162   I2Cx_WriteMultiple(Addr, Reg, FMPI2C_MEMAD
D_SIZE_16BIT,(uint8_t*)&Value, 2);
01163 }
01164
01165 /**
01166   * @brief  Reads a single data.
01167   * @param  Addr: I2C address
01168   * @param  Reg: Reg address
01169   * @retval Data to be read
01170   */
01171 uint16_t AUDIO_IO_Read(uint8_t Addr, uint16_
t Reg)
01172 {
01173   uint16_t read_value = 0, tmp = 0;
01174
01175   I2Cx_ReadMultiple(Addr, Reg, FMPI2C_MEMADD
_SIZE_16BIT, (uint8_t*)&read_value, 2);
01176
01177   tmp = ((uint16_t)(read_value >> 8) & 0x00F
F);
01178
01179   tmp |= ((uint16_t)(read_value << 8)& 0xFF0
0);
01180
01181   read_value = tmp;
01182
01183   return read_value;
01184 }
01185
01186 /**
01187   * @brief  AUDIO Codec delay
```

```
01188   * @param  Delay: Delay in ms
01189   */
01190 void AUDIO_IO_Delay(uint32_t Delay)
01191 {
01192   HAL_Delay(Delay);
01193 }
01194
01195 /********************************** LINK CAME
RA ***************************************/
01196
01197 /**
01198   * @brief  Initializes Camera low level.
01199   */
01200 void CAMERA_IO_Init(void)
01201 {
01202   I2Cx_Init();
01203 }
01204
01205 /**
01206   * @brief  Camera writes single data.
01207   * @param  Addr: I2C address
01208   * @param  Reg: Register address
01209   * @param  Value: Data to be written
01210   */
01211 void CAMERA_IO_Write(uint8_t Addr, uint16_t
Reg, uint16_t Value)
01212 {
01213   uint16_t tmp = Value;
01214   /* For S5K5CAG sensor, 16 bits accesses ar
e used */
01215   Value = ((uint16_t)(tmp >> 8) & 0x00FF);
01216   Value |= ((uint16_t)(tmp << 8)& 0xFF00);
01217   I2Cx_WriteMultiple(Addr, Reg, FMPI2C_MEMAD
D_SIZE_16BIT,(uint8_t*)&Value, 2);
01218 }
01219
01220 /**
```

```
01221   * @brief  Camera reads single data.
01222   * @param  Addr: I2C address
01223   * @param  Reg: Register address
01224   * @retval Read data
01225   */
01226 uint16_t CAMERA_IO_Read(uint8_t Addr, uint16
_t Reg)
01227 {
01228   uint16_t read_value = 0, tmp = 0;
01229   /* For S5K5CAG sensor, 16 bits accesses ar
e used */
01230   I2Cx_ReadMultiple(Addr, Reg, FMPI2C_MEMADD
_SIZE_16BIT, (uint8_t*)&read_value, 2);
01231   tmp = ((uint16_t)(read_value >> 8) & 0x00F
F);
01232   tmp |= ((uint16_t)(read_value << 8)& 0xFF0
0);
01233   read_value = tmp;
01234   return read_value;
01235 }
01236
01237 /**
01238   * @brief  Camera delay
01239   * @param  Delay: Delay in ms
01240   */
01241 void CAMERA_Delay(uint32_t Delay)
01242 {
01243   HAL_Delay(Delay);
01244 }
01245
01246 /******************************* LINK I2C E
EPROM *****************************/
01247
01248 /**
01249   * @brief  Initializes peripherals used by
the I2C EEPROM driver.
01250   */
```

```
01251 void EEPROM_IO_Init(void)
01252 {
01253   I2Cx_Init();
01254 }
01255
01256 /**
01257   * @brief  Write data to I2C EEPROM driver
in using DMA channel.
01258   * @param  DevAddress: Target device address

01259   * @param  MemAddress: Internal memory addr
ess
01260   * @param  pBuffer: Pointer to data buffer
01261   * @param  BufferSize: Amount of data to be
 sent
01262   * @retval HAL status
01263   */
01264 HAL_StatusTypeDef EEPROM_IO_WriteData(uint16
_t DevAddress, uint16_t MemAddress, uint8_t* pBuff
er, uint32_t BufferSize)
01265 {
01266   return (I2Cx_WriteMultiple(DevAddress, Mem
Address, FMPI2C_MEMADD_SIZE_16BIT, pBuffer, Buffer
Size));
01267 }
01268
01269 /**
01270   * @brief  Read data from I2C EEPROM driver
 in using DMA channel.
01271   * @param  DevAddress: Target device address

01272   * @param  MemAddress: Internal memory addr
ess
01273   * @param  pBuffer: Pointer to data buffer
01274   * @param  BufferSize: Amount of data to be
 read
01275   * @retval HAL status
```

```
01276   */
01277 HAL_StatusTypeDef EEPROM_IO_ReadData(uint16_
t DevAddress, uint16_t MemAddress, uint8_t* pBuffe
r, uint32_t BufferSize)
01278 {
01279   return (I2Cx_ReadMultiple(DevAddress, MemA
ddress, FMPI2C_MEMADD_SIZE_16BIT, pBuffer, BufferS
ize));
01280 }
01281
01282 /**
01283   * @brief  Checks if target device is ready
 for communication.
01284   * @note   This function is used with Memor
y devices
01285   * @param  DevAddress: Target device address

01286   * @param  Trials: Number of trials
01287   * @retval HAL status
01288   */
01289 HAL_StatusTypeDef EEPROM_IO_IsDeviceReady(ui
nt16_t DevAddress, uint32_t Trials)
01290 {
01291   return (I2Cx_IsDeviceReady(DevAddress, Tri
als));
01292 }
01293
01294
01295
01296 /**
01297   * @}
01298   */
01299
01300 /**
01301   * @}
01302   */
01303
```

```
01304 /**
01305   * @}
01306   */
01307
01308 /**
01309   * @}
01310   */
01311
01312 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM32446E_EVAL BSP User Manual

STM32446E EVAL LOW LEVEL

STM32446E EVAL

Modules

- STM32446E EVAL LOW LEVEL Private TypesDefinitions
- STM32446E EVAL LOW LEVEL Private Defines
- STM32446E EVAL LOW LEVEL Private Macros
- STM32446E EVAL LOW LEVEL Private Variables
- STM32446E EVAL LOW LEVEL Private FunctionPrototypes
- STM32446E EVAL LOW LEVEL Private Functions
- STM32446E EVAL LOW LEVEL Exported Types
- STM32446E EVAL LOW LEVEL Exported Constants
- STM32446E EVAL LOW LEVEL Exported Macros
- STM32446E EVAL LOW LEVEL Exported Functions

STM32446E_EVAL BSP User Manual

STM32446E EVAL AUDIO

This file includes the low layer driver for wm8994 Audio Codec available on STM32446E-EVAL evaluation board(MB1045). More...

Modules

- STM32446E EVAL AUDIO Private Types
- STM32446E EVAL AUDIO Private Defines
- STM32446E EVAL AUDIO Private Macros
- STM32446E EVAL AUDIO Private Variables
- STM32446E EVAL AUDIO Private Function Prototypes
- STM32446E EVAL AUDIO OUT Private Functions
- STM32446E EVAL AUDIO IN Private Functions
- STM32446E EVAL AUDIO Exported Types
- STM32446E EVAL AUDIO Exported Constants
- STM32446E EVAL AUDIO Exported Variables
- STM32446E EVAL AUDIO Exported Macros
- STM32446E EVAL AUDIO OUT Exported Functions
- STM32446E EVAL AUDIO IN Exported Functions

Detailed Description

This file includes the low layer driver for wm8994 Audio Codec available on STM32446E-EVAL evaluation board(MB1045).

STM32446E_EVAL BSP User Manual

STM32446E-EVAL CAMERA

STM32446E EVAL

Modules

- STM32446E EVAL CAMERA Private TypesDefinitions
- STM32446E EVAL CAMERA Private Defines
- STM32446E EVAL CAMERA Private Macros
- STM32446E EVAL CAMERA Private Variables
- STM32446E EVAL CAMERA Private FunctionPrototypes
- STM32446E EVAL CAMERA Private Functions
- STM32446E EVAL CAMERA Exported Types
- STM32446E EVAL CAMERA Exported Constants
- STM32446E EVAL CAMERA Exported Functions

STM32446E_EVAL BSP User Manual

STM32446E EVAL EEPROM

STM32446E EVAL

This file includes the I2C EEPROM driver of STM32446E-EVAL evaluation board. More...

Modules

- STM32446E EVAL EEPROM Private Types
- STM32446E EVAL EEPROM Private Defines
- STM32446E EVAL EEPROM Private Macros
- STM32446E EVAL EEPROM Private Variables
- STM32446E EVAL EEPROM Private Function Prototypes
- STM32446E EVAL EEPROM Private Functions
- STM32446E EVAL EEPROM Exported Types
- STM32446E EVAL EEPROM Exported Constants
- STM32446E EVAL EEPROM Exported Macros
- STM32446E EVAL EEPROM Exported Functions

Detailed Description

This file includes the I2C EEPROM driver of STM32446E-EVAL evaluation board.

STM32446E_EVAL BSP User Manual

STM32446E EVAL IO

STM32446E EVAL

Modules

- STM32446E EVAL IO Private Types Definitions
- STM32446E EVAL IO Private Defines
- STM32446E EVAL IO Private Macros
- STM32446E EVAL IO Private Variables
- STM32446E EVAL IO Private Function Prototypes
- STM32446E EVAL IO Private Functions
- STM32446E EVAL IO Exported Types
- STM32446E EVAL IO Exported Constants
- STM32446E EVAL IO Exported Macro
- STM32446E EVAL IO Exported Functions

STM32446E_EVAL BSP User Manual

STM32446E EVAL LCD

STM32446E EVAL

Modules

- STM32446E EVAL LCD Private TypesDefinitions
- STM32446E EVAL LCD Private Defines
- STM32446E EVAL LCD Private Macros
- STM32446E EVAL LCD Private Variables
- STM32446E EVAL LCD Private FunctionPrototypes
- STM32446E EVAL LCD Private Functions
- STM32446E EVAL LCD Exported Types
- STM32446E EVAL LCD Exported Constants
- STM32446E EVAL LCD Exported Functions

STM32446E_EVAL BSP User Manual

STM32446E EVAL LCD Exported Types

STM32446E EVAL LCD

Data Structures

struct	**LCD_DrawPropTypeDef**

STM32446E_EVAL BSP User Manual

stm32446e_eval_lcd.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval_lcd.h
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm32446e_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 /* Define to prevent recursive inclusion ---
-----------------------------------*/
00040 #ifndef __STM32446E_EVAL_LCD_H
00041 #define __STM32446E_EVAL_LCD_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------
-----------------------------------*/
00048 #include "stm32446e_eval.h"
00049 #include "../Components/ili9325/ili9325.h"
00050 #include "../../../Utilities/Fonts/fonts.h"
00051
00052 /** @addtogroup BSP
00053   * @{
00054   */
00055
00056 /** @addtogroup STM32446E_EVAL
00057   * @{
00058   */
00059
00060 /** @addtogroup STM32446E_EVAL_LCD
00061   * @{
00062   */
00063
00064 /** @defgroup STM32446E_EVAL_LCD_Exported_Ty
pes STM32446E EVAL LCD Exported Types
00065   * @{
00066   */
00067 typedef struct
```

```
00068 {
00069   uint32_t TextColor;
00070   uint32_t BackColor;
00071   sFONT    *pFont;
00072 }LCD_DrawPropTypeDef;
00073 /**
00074   * @}
00075   */
00076
00077 /** @defgroup STM32446E_EVAL_LCD_Exported_Co
nstants STM32446E EVAL LCD Exported Constants
00078   * @{
00079   */
00080 /**
00081   * @brief  LCD status structure definition

00082   */
00083 #define LCD_OK         ((uint8_t)0x00)
00084 #define LCD_ERROR      ((uint8_t)0x01)
00085 #define LCD_TIMEOUT    ((uint8_t)0x02)
00086
00087 typedef struct
00088 {
00089   int16_t X;
00090   int16_t Y;
00091 }Point, * pPoint;
00092
00093 /**
00094   * @brief  Line mode structures definition
00095   */
00096 typedef enum
00097 {
00098   CENTER_MODE           = 0x01,    /* Center
mode */
00099   RIGHT_MODE            = 0x02,    /* Right m
ode  */
00100   LEFT_MODE             = 0x03     /* Left mo
```

```
de   */
00101 }Line_ModeTypdef;
00102
00103 /**
00104   * @brief  LCD color
00105   */
00106 #define LCD_COLOR_BLUE          ((uint16_t)0
x001F)
00107 #define LCD_COLOR_GREEN         ((uint16_t)0
x07E0)
00108 #define LCD_COLOR_RED           ((uint16_t)0
xF800)
00109 #define LCD_COLOR_CYAN          ((uint16_t)0
x07FF)
00110 #define LCD_COLOR_MAGENTA       ((uint16_t)0
xF81F)
00111 #define LCD_COLOR_YELLOW        ((uint16_t)0
xFFE0)
00112 #define LCD_COLOR_LIGHTBLUE     ((uint16_t)0
x841F)
00113 #define LCD_COLOR_LIGHTGREEN    ((uint16_t)0
x87F0)
00114 #define LCD_COLOR_LIGHTRED      ((uint16_t)0
xFC10)
00115 #define LCD_COLOR_LIGHTMAGENTA  ((uint16_t)0
xFC1F)
00116 #define LCD_COLOR_LIGHTYELLOW   ((uint16_t)0
xFFF0)
00117 #define LCD_COLOR_DARKBLUE      ((uint16_t)0
x0010)
00118 #define LCD_COLOR_DARKGREEN     ((uint16_t)0
x0400)
00119 #define LCD_COLOR_DARKRED       ((uint16_t)0
x8000)
00120 #define LCD_COLOR_DARKCYAN      ((uint16_t)0
x0410)
00121 #define LCD_COLOR_DARKMAGENTA   ((uint16_t)0
```

```
x8010)
00122 #define LCD_COLOR_DARKYELLOW    ((uint16_t)0
x8400)
00123 #define LCD_COLOR_WHITE         ((uint16_t)0
xFFFF)
00124 #define LCD_COLOR_LIGHTGRAY     ((uint16_t)0
xD69A)
00125 #define LCD_COLOR_GRAY          ((uint16_t)0
x8410)
00126 #define LCD_COLOR_DARKGRAY      ((uint16_t)0
x4208)
00127 #define LCD_COLOR_BLACK         ((uint16_t)0
x0000)
00128 #define LCD_COLOR_BROWN         ((uint16_t)0
xA145)
00129 #define LCD_COLOR_ORANGE        ((uint16_t)0
xFD20)
00130
00131 /**
00132   * @brief LCD default font
00133   */
00134 #define LCD_DEFAULT_FONT         Font24
00135
00136 /**
00137   * @}
00138   */
00139
00140 /** @defgroup STM32446E_EVAL_LCD_Exported_Fu
nctions STM32446E EVAL LCD Exported Functions
00141   * @{
00142   */
00143 uint8_t  BSP_LCD_Init(void);
00144 uint8_t  BSP_LCD_DeInit(void);
00145 uint32_t BSP_LCD_GetXSize(void);
00146 uint32_t BSP_LCD_GetYSize(void);
00147
00148 uint16_t BSP_LCD_GetTextColor(void);
```

```
00149 uint16_t BSP_LCD_GetBackColor(void);
00150 void     BSP_LCD_SetTextColor(__IO uint16_t
Color);
00151 void     BSP_LCD_SetBackColor(__IO uint16_t
Color);
00152 void     BSP_LCD_SetFont(sFONT *fonts);
00153 sFONT    *BSP_LCD_GetFont(void);
00154
00155 void     BSP_LCD_Clear(uint16_t Color);
00156 void     BSP_LCD_ClearStringLine(uint16_t Li
ne);
00157 void     BSP_LCD_DisplayStringAtLine(uint16_
t Line, uint8_t *ptr);
00158 void     BSP_LCD_DisplayStringAt(uint16_t Xp
os, uint16_t Ypos, uint8_t *Text, Line_ModeTypdef
Mode);
00159 void     BSP_LCD_DisplayChar(uint16_t Xpos,
uint16_t Ypos, uint8_t Ascii);
00160
00161 uint16_t BSP_LCD_ReadPixel(uint16_t Xpos, ui
nt16_t Ypos);
00162 void     BSP_LCD_DrawPixel(uint16_t Xpos, ui
nt16_t Ypos, uint16_t RGB_Code);
00163 void     BSP_LCD_DrawHLine(uint16_t Xpos, ui
nt16_t Ypos, uint16_t Length);
00164 void     BSP_LCD_DrawVLine(uint16_t Xpos, ui
nt16_t Ypos, uint16_t Length);
00165 void     BSP_LCD_DrawLine(uint16_t x1, uint1
6_t y1, uint16_t x2, uint16_t y2);
00166 void     BSP_LCD_DrawRect(uint16_t Xpos, uin
t16_t Ypos, uint16_t Width, uint16_t Height);
00167 void     BSP_LCD_DrawCircle(uint16_t Xpos, u
int16_t Ypos, uint16_t Radius);
00168 void     BSP_LCD_DrawPolygon(pPoint Points,
uint16_t PointCount);
00169 void     BSP_LCD_DrawEllipse(int Xpos, int Y
pos, int XRadius, int YRadius);
```

```
00170 void     BSP_LCD_DrawBitmap(uint16_t Xpos, u
int16_t Ypos, uint8_t *pbmp);
00171 void     BSP_LCD_DrawRGBImage(uint16_t Xpos,
 uint16_t Ypos, uint16_t Xsize, uint16_t Ysize, ui
nt8_t *pbmp);
00172 void     BSP_LCD_FillRect(uint16_t Xpos, uin
t16_t Ypos, uint16_t Width, uint16_t Height);
00173 void     BSP_LCD_FillCircle(uint16_t Xpos, u
int16_t Ypos, uint16_t Radius);
00174 void     BSP_LCD_FillPolygon(pPoint Points,
uint16_t PointCount);
00175 void     BSP_LCD_FillEllipse(int Xpos, int Y
pos, int XRadius, int YRadius);
00176
00177 void     BSP_LCD_DisplayOff(void);
00178 void     BSP_LCD_DisplayOn(void);
00179
00180 /**
00181   * @}
00182   */
00183
00184 /**
00185   * @}
00186   */
00187
00188 /**
00189   * @}
00190   */
00191
00192 /**
00193   * @}
00194   */
00195
00196 #ifdef __cplusplus
00197 }
00198 #endif
00199
```

```
00200 #endif /* __STM32446E_EVAL_LCD_H */
00201
00202 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM32446E_EVAL BSP User Manual

stm32446e_eval_lcd.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval_lcd.c
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file includes the driver for Liquid Crystal Display (LCD) module
00008   *          mounted on STM32446E-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 indirectly
 an LCD TFT.
00044    - This driver supports the ILI9325 LCD mo
unted on MB785 daughter board
00045    - The ILI9325 component driver MUST be in
cluded with this driver.
00046
00047 2. Driver description:
00048 ---------------------
00049   + Initialization steps:
00050      o Initialize the LCD using the BSP_LCD_
Init() function.
00051
00052   + Display on LCD
00053      o Clear the hole LCD using BSP_LCD_Clea
r() function or only one specified string
00054        line using the BSP_LCD_ClearStringLin
e() function.
00055      o Display a character on the specified
line and column using the BSP_LCD_DisplayChar()
00056        function or a complete string line us
ing the BSP_LCD_DisplayStringAtLine() function.
00057      o Display a string line on the specifie
d position (x,y in pixel) and align mode
00058        using the BSP_LCD_DisplayStringAtLine
() function.
```

```
00059      o Draw and fill a basic shapes (dot, li
ne, rectangle, circle, ellipse, .. bitmap)
00060        on LCD using the available set of fun
ctions.
00061
00062 ----------------------------------------
---------------------------------*/
00063
00064 /* Includes ------------------------------
---------------------------------*/
00065 #include "stm32446e_eval_lcd.h"
00066 #include "../../../Utilities/Fonts/fonts.h"
00067 #include "../../../Utilities/Fonts/font24.c"
00068 #include "../../../Utilities/Fonts/font20.c"
00069 #include "../../../Utilities/Fonts/font16.c"
00070 #include "../../../Utilities/Fonts/font12.c"
00071 #include "../../../Utilities/Fonts/font8.c"
00072
00073 /** @addtogroup BSP
00074   * @{
00075   */
00076
00077 /** @addtogroup STM32446E_EVAL
00078   * @{
00079   */
00080
00081 /** @defgroup STM32446E_EVAL_LCD STM32446E E
VAL LCD
00082   * @{
00083   */
00084
00085 /** @defgroup STM32446E_EVAL_LCD_Private_Typ
esDefinitions STM32446E EVAL LCD Private TypesDefi
nitions
00086   * @{
00087   */
00088 /**
```

```
00089   * @}
00090   */
00091
00092 /** @defgroup STM32446E_EVAL_LCD_Private_Def
ines STM32446E EVAL LCD Private Defines
00093   * @{
00094   */
00095 #define POLY_X(Z)              ((int32_t)((P
oints + Z)->X))
00096 #define POLY_Y(Z)              ((int32_t)((P
oints + Z)->Y))
00097 /**
00098   * @}
00099   */
00100
00101 /** @defgroup STM32446E_EVAL_LCD_Private_Mac
ros STM32446E EVAL LCD Private Macros
00102   * @{
00103   */
00104 #define ABS(X)  ((X) > 0 ? (X) : -(X))
00105 /**
00106   * @}
00107   */
00108
00109 /** @defgroup STM32446E_EVAL_LCD_Private_Var
iables STM32446E EVAL LCD Private Variables
00110   * @{
00111   */
00112 LCD_DrawPropTypeDef DrawProp;
00113 static LCD_DrvTypeDef  *LcdDrv;
00114 /**
00115   * @}
00116   */
00117
00118 /** @defgroup STM32446E_EVAL_LCD_Private_Fun
ctionPrototypes STM32446E EVAL LCD Private Functio
nPrototypes
```

```
00119   * @{
00120   */
00121 static void DrawChar(uint16_t Xpos, uint16_t
 Ypos, const uint8_t *c);
00122 static void SetDisplayWindow(uint16_t Xpos,
uint16_t Ypos, uint16_t Width, uint16_t Height);
00123 static void FillTriangle(uint16_t x1, uint16
_t x2, uint16_t x3, uint16_t y1, uint16_t y2, uint
16_t y3);
00124 /**
00125   * @}
00126   */
00127
00128 /** @defgroup STM32446E_EVAL_LCD_Private_Fun
ctions STM32446E EVAL LCD Private Functions
00129   * @{
00130   */
00131
00132 /**
00133   * @brief  Initializes the LCD.
00134   * @retval LCD state
00135   */
00136 uint8_t BSP_LCD_Init(void)
00137 {
00138   uint8_t ret = LCD_ERROR;
00139
00140   /* Default value for draw propriety */
00141   DrawProp.BackColor = 0xFFFF;
00142   DrawProp.pFont     = &Font24;
00143   DrawProp.TextColor = 0x0000;
00144
00145   if(ili9325_drv.ReadID() == ILI9325_ID)
00146   {
00147     LcdDrv = &ili9325_drv;
00148
00149     /* LCD Init */
00150     LcdDrv->Init();
```

```
00151
00152     /* Initialize the font */
00153     BSP_LCD_SetFont(&LCD_DEFAULT_FONT);
00154
00155     ret = LCD_OK;
00156   }
00157
00158   return ret;
00159 }
00160
00161 /**
00162   * @brief  DeInitializes the LCD.
00163   * @retval LCD state
00164   */
00165 uint8_t BSP_LCD_DeInit(void)
00166 {
00167   /* Actually LcdDrv does not provide a DeIn
it function */
00168   return LCD_OK;
00169 }
00170
00171 /**
00172   * @brief  Gets the LCD X size.
00173   * @retval Used LCD X size
00174   */
00175 uint32_t BSP_LCD_GetXSize(void)
00176 {
00177   return(LcdDrv->GetLcdPixelWidth());
00178 }
00179
00180 /**
00181   * @brief  Gets the LCD Y size.
00182   * @retval Used LCD Y size
00183   */
00184 uint32_t BSP_LCD_GetYSize(void)
00185 {
00186   return(LcdDrv->GetLcdPixelHeight());
```

```
00187 }
00188
00189 /**
00190   * @brief  Gets the LCD text color.
00191   * @retval Used text color.
00192   */
00193 uint16_t BSP_LCD_GetTextColor(void)
00194 {
00195   return DrawProp.TextColor;
00196 }
00197
00198 /**
00199   * @brief  Gets the LCD background color.
00200   * @retval Used background color
00201   */
00202 uint16_t BSP_LCD_GetBackColor(void)
00203 {
00204   return DrawProp.BackColor;
00205 }
00206
00207 /**
00208   * @brief  Sets the LCD text color.
00209   * @param  Color: Text color code RGB(5-6-5)

00210   */
00211 void BSP_LCD_SetTextColor(uint16_t Color)
00212 {
00213   DrawProp.TextColor = Color;
00214 }
00215
00216 /**
00217   * @brief  Sets the LCD background color.
00218   * @param  Color: Background color code RGB
(5-6-5)
00219   */
00220 void BSP_LCD_SetBackColor(uint16_t Color)
00221 {
```

```
00222   DrawProp.BackColor = Color;
00223 }
00224
00225 /**
00226   * @brief  Sets the LCD text font.
00227   * @param  fonts: Font to be used
00228   */
00229 void BSP_LCD_SetFont(sFONT *fonts)
00230 {
00231   DrawProp.pFont = fonts;
00232 }
00233
00234 /**
00235   * @brief  Gets the LCD text font.
00236   * @retval Used font
00237   */
00238 sFONT *BSP_LCD_GetFont(void)
00239 {
00240   return DrawProp.pFont;
00241 }
00242
00243 /**
00244   * @brief  Clears the hole LCD.
00245   * @param  Color: Color of the background
00246   */
00247 void BSP_LCD_Clear(uint16_t Color)
00248 {
00249   uint32_t counter = 0;
00250   uint32_t y_size = 0;
00251   uint32_t color_backup = DrawProp.TextColor
;
00252
00253   DrawProp.TextColor = Color;
00254   y_size =  BSP_LCD_GetYSize();
00255
00256   for(counter = 0; counter < y_size; counter
++)
```

```
00257   {
00258     BSP_LCD_DrawHLine(0, counter, BSP_LCD_Ge
tXSize());
00259   }
00260   DrawProp.TextColor = color_backup;
00261   BSP_LCD_SetTextColor(DrawProp.TextColor);
00262 }
00263
00264 /**
00265   * @brief  Clears the selected line.
00266   * @param  Line: Line to be cleared
00267   *          This parameter can be one of th
e following values:
00268   *            @arg  0..9: if the Current fo
nts is Font16x24
00269   *            @arg  0..19: if the Current f
onts is Font12x12 or Font8x12
00270   *            @arg  0..29: if the Current f
onts is Font8x8
00271   */
00272 void BSP_LCD_ClearStringLine(uint16_t Line)
00273 {
00274   uint32_t color_backup = DrawProp.TextColor
;
00275
00276   DrawProp.TextColor = DrawProp.BackColor;;
00277
00278   /* Draw a rectangle with background color
*/
00279   BSP_LCD_FillRect(0, (Line * DrawProp.pFont
->Height), BSP_LCD_GetXSize(), DrawProp.pFont->Hei
ght);
00280
00281   DrawProp.TextColor = color_backup;
00282   BSP_LCD_SetTextColor(DrawProp.TextColor);
00283 }
00284
```

```
00285 /**
00286   * @brief  Displays one character.
00287   * @param  Xpos: Start column address
00288   * @param  Ypos: Line where to display the
character shape.
00289   * @param  Ascii: Character ascii code
00290   *           This parameter must be a numbe
r between Min_Data = 0x20 and Max_Data = 0x7E
00291   */
00292 void BSP_LCD_DisplayChar(uint16_t Xpos, uint
16_t Ypos, uint8_t Ascii)
00293 {
00294   DrawChar(Xpos, Ypos, &DrawProp.pFont->tabl
e[(Ascii-' ') *\
00295     DrawProp.pFont->Height * ((DrawProp.pFont
->Width + 7) / 8)]);
00296 }
00297
00298 /**
00299   * @brief  Displays characters on the LCD.
00300   * @param  Xpos: X position (in pixel)
00301   * @param  Ypos: Y position (in pixel)
00302   * @param  Text: Pointer to string to displ
ay on LCD
00303   * @param  Mode: Display mode
00304   *          This parameter can be one of th
e following values:
00305   *            @arg  CENTER_MODE
00306   *            @arg  RIGHT_MODE
00307   *            @arg  LEFT_MODE
00308   */
00309 void BSP_LCD_DisplayStringAt(uint16_t Xpos,
uint16_t Ypos, uint8_t *Text, Line_ModeTypdef Mode
)
00310 {
00311   uint16_t refcolumn = 1, i = 0;
00312   uint32_t size = 0, xsize = 0;
```

```
00313   uint8_t  *ptr = Text;
00314
00315   /* Get the text size */
00316   while (*ptr++) size ++ ;
00317
00318   /* Characters number per line */
00319   xsize = (BSP_LCD_GetXSize()/DrawProp.pFont
->Width);
00320
00321   switch (Mode)
00322   {
00323   case CENTER_MODE:
00324     {
00325       refcolumn = Xpos + ((xsize - size)* Dr
awProp.pFont->Width) / 2;
00326       break;
00327     }
00328   case LEFT_MODE:
00329     {
00330       refcolumn = Xpos;
00331       break;
00332     }
00333   case RIGHT_MODE:
00334     {
00335       refcolumn =  - Xpos + ((xsize - size)*
DrawProp.pFont->Width);
00336       break;
00337     }
00338   default:
00339     {
00340       refcolumn = Xpos;
00341       break;
00342     }
00343   }
00344
00345   /* Check that the Start column is located
in the screen */
```

```
00346   if ((refcolumn < 1) || (refcolumn >= 0x800
0))
00347   {
00348     refcolumn = 1;
00349   }
00350
00351   /* Send the string character by character
on lCD */
00352   while ((*Text != 0) & (((BSP_LCD_GetXSize(
) - (i*DrawProp.pFont->Width)) & 0xFFFF) >= DrawPr
op.pFont->Width))
00353   {
00354     /* Display one character on LCD */
00355     BSP_LCD_DisplayChar(refcolumn, Ypos, *Te
xt);
00356     /* Decrement the column position by 16 */

00357     refcolumn += DrawProp.pFont->Width;
00358     /* Point on the next character */
00359     Text++;
00360     i++;
00361   }
00362 }
00363
00364 /**
00365   * @brief  Displays a character on the LCD.
00366   * @param  Line: Line where to display the
character shape
00367   *          This parameter can be one of th
e following values:
00368   *            @arg  0..9: if the Current fo
nts is Font16x24
00369   *            @arg  0..19: if the Current f
onts is Font12x12 or Font8x12
00370   *            @arg  0..29: if the Current f
onts is Font8x8
00371   * @param  ptr: Pointer to string to displa
```

```
y on LCD
00372   */
00373 void BSP_LCD_DisplayStringAtLine(uint16_t Li
ne, uint8_t *ptr)
00374 {
00375   BSP_LCD_DisplayStringAt(0, LINE(Line), ptr
, LEFT_MODE);
00376 }
00377
00378 /**
00379   * @brief  Reads an LCD pixel.
00380   * @param  Xpos: X position
00381   * @param  Ypos: Y position
00382   * @retval RGB pixel color
00383   */
00384 uint16_t BSP_LCD_ReadPixel(uint16_t Xpos, ui
nt16_t Ypos)
00385 {
00386   uint16_t ret = 0;
00387
00388   if(LcdDrv->ReadPixel != NULL)
00389   {
00390     ret = LcdDrv->ReadPixel(Xpos, Ypos);
00391   }
00392
00393   return ret;
00394 }
00395
00396 /**
00397   * @brief  Draws a pixel on LCD.
00398   * @param  Xpos: X position
00399   * @param  Ypos: Y position
00400   * @param  RGB_Code: Pixel color in RGB mod
e (5-6-5)
00401   */
00402 void BSP_LCD_DrawPixel(uint16_t Xpos, uint16
_t Ypos, uint16_t RGB_Code)
```

```
00403 {
00404   if(LcdDrv->WritePixel != NULL)
00405   {
00406     LcdDrv->WritePixel(Xpos, Ypos, RGB_Code)
;
00407   }
00408 }
00409
00410 /**
00411   * @brief  Draws an horizontal line.
00412   * @param  Xpos: X position
00413   * @param  Ypos: Y position
00414   * @param  Length: Line length
00415   */
00416 void BSP_LCD_DrawHLine(uint16_t Xpos, uint16
_t Ypos, uint16_t Length)
00417 {
00418   uint32_t index = 0;
00419
00420   if(LcdDrv->DrawHLine != NULL)
00421   {
00422     LcdDrv->DrawHLine(DrawProp.TextColor, Xp
os, Ypos, Length);
00423   }
00424   else
00425   {
00426     for(index = 0; index < Length; index++)
00427     {
00428       BSP_LCD_DrawPixel((Xpos + index), Ypos
, DrawProp.TextColor);
00429     }
00430   }
00431 }
00432
00433 /**
00434   * @brief  Draws a vertical line.
00435   * @param  Xpos: X position
```

```
00436   * @param  Ypos: Y position
00437   * @param  Length: Line length
00438   */
00439 void BSP_LCD_DrawVLine(uint16_t Xpos, uint16
_t Ypos, uint16_t Length)
00440 {
00441   uint32_t index = 0;
00442
00443   if(LcdDrv->DrawVLine != NULL)
00444   {
00445     LcdDrv->DrawVLine(DrawProp.TextColor, Xp
os, Ypos, Length);
00446   }
00447   else
00448   {
00449     for(index = 0; index < Length; index++)
00450     {
00451       BSP_LCD_DrawPixel(Xpos, Ypos + index,
DrawProp.TextColor);
00452     }
00453   }
00454 }
00455
00456 /**
00457   * @brief  Draws an uni-line (between two p
oints).
00458   * @param  x1: Point 1 X position
00459   * @param  y1: Point 1 Y position
00460   * @param  x2: Point 2 X position
00461   * @param  y2: Point 2 Y position
00462   */
00463 void BSP_LCD_DrawLine(uint16_t x1, uint16_t
y1, uint16_t x2, uint16_t y2)
00464 {
00465   int16_t deltax = 0, deltay = 0, x = 0, y =
 0, xinc1 = 0, xinc2 = 0,
00466   yinc1 = 0, yinc2 = 0, den = 0, num = 0, nu
```

```
madd = 0, numpixels = 0,
00467   curpixel = 0;
00468
00469   deltax = ABS(x2 - x1);          /* The diffe
rence between the x's */
00470   deltay = ABS(y2 - y1);          /* The diffe
rence between the y's */
00471   x = x1;                         /* Start x o
ff at the first pixel */
00472   y = y1;                         /* Start y o
ff at the first pixel */
00473
00474   if (x2 >= x1)                   /* The x-val
ues are increasing */
00475   {
00476     xinc1 = 1;
00477     xinc2 = 1;
00478   }
00479   else                            /* The x-val
ues are decreasing */
00480   {
00481     xinc1 = -1;
00482     xinc2 = -1;
00483   }
00484
00485   if (y2 >= y1)                   /* The y-val
ues are increasing */
00486   {
00487     yinc1 = 1;
00488     yinc2 = 1;
00489   }
00490   else                            /* The y-val
ues are decreasing */
00491   {
00492     yinc1 = -1;
00493     yinc2 = -1;
00494   }
```

```
00495
00496   if (deltax >= deltay)         /* There is
at least one x-value for every y-value */
00497   {
00498     xinc1 = 0;                  /* Don't cha
nge the x when numerator >= denominator */
00499     yinc2 = 0;                  /* Don't cha
nge the y for every iteration */
00500     den = deltax;
00501     num = deltax / 2;
00502     numadd = deltay;
00503     numpixels = deltax;         /* There are
 more x-values than y-values */
00504   }
00505   else                          /* There is
at least one y-value for every x-value */
00506   {
00507     xinc2 = 0;                  /* Don't cha
nge the x for every iteration */
00508     yinc1 = 0;                  /* Don't cha
nge the y when numerator >= denominator */
00509     den = deltay;
00510     num = deltay / 2;
00511     numadd = deltax;
00512     numpixels = deltay;         /* There are
 more y-values than x-values */
00513   }
00514
00515   for (curpixel = 0; curpixel <= numpixels;
curpixel++)
00516   {
00517     BSP_LCD_DrawPixel(x, y, DrawProp.TextCol
or);  /* Draw the current pixel */
00518     num += numadd;
  /* Increase the numerator by the top of the frac
tion */
00519     if (num >= den)
```

```
  /* Check if numerator >= denominator */
00520     {
00521       num -= den;
  /* Calculate the new numerator value */
00522       x += xinc1;
  /* Change the x as appropriate */
00523       y += yinc1;
  /* Change the y as appropriate */
00524     }
00525     x += xinc2;
  /* Change the x as appropriate */
00526     y += yinc2;
  /* Change the y as appropriate */
00527   }
00528 }
00529
00530 /**
00531   * @brief  Draws a rectangle.
00532   * @param  Xpos: X position
00533   * @param  Ypos: Y position
00534   * @param  Width: Rectangle width
00535   * @param  Height: Rectangle height
00536   */
00537 void BSP_LCD_DrawRect(uint16_t Xpos, uint16_
t Ypos, uint16_t Width, uint16_t Height)
00538 {
00539   /* Draw horizontal lines */
00540   BSP_LCD_DrawHLine(Xpos, Ypos, Width);
00541   BSP_LCD_DrawHLine(Xpos, (Ypos+ Height), Wi
dth);
00542
00543   /* Draw vertical lines */
00544   BSP_LCD_DrawVLine(Xpos, Ypos, Height);
00545   BSP_LCD_DrawVLine((Xpos + Width), Ypos, He
ight);
00546 }
00547
```

```
00548 /**
00549   * @brief  Draws a circle.
00550   * @param  Xpos: X position
00551   * @param  Ypos: Y position
00552   * @param  Radius: Circle radius
00553   */
00554 void BSP_LCD_DrawCircle(uint16_t Xpos, uint1
6_t Ypos, uint16_t Radius)
00555 {
00556   int32_t  decision;       /* Decision Varia
ble */
00557   uint32_t  current_x;   /* Current X Value
*/
00558   uint32_t  current_y;   /* Current Y Value
*/
00559
00560   decision = 3 - (Radius << 1);
00561   current_x = 0;
00562   current_y = Radius;
00563
00564   while (current_x <= current_y)
00565   {
00566     BSP_LCD_DrawPixel((Xpos + current_x), (Y
pos - current_y), DrawProp.TextColor);
00567
00568     BSP_LCD_DrawPixel((Xpos - current_x), (Y
pos - current_y), DrawProp.TextColor);
00569
00570     BSP_LCD_DrawPixel((Xpos + current_y), (Y
pos - current_x), DrawProp.TextColor);
00571
00572     BSP_LCD_DrawPixel((Xpos - current_y), (Y
pos - current_x), DrawProp.TextColor);
00573
00574     BSP_LCD_DrawPixel((Xpos + current_x), (Y
pos + current_y), DrawProp.TextColor);
00575
```

```
00576     BSP_LCD_DrawPixel((Xpos - current_x), (Y
pos + current_y), DrawProp.TextColor);
00577
00578     BSP_LCD_DrawPixel((Xpos + current_y), (Y
pos + current_x), DrawProp.TextColor);
00579
00580     BSP_LCD_DrawPixel((Xpos - current_y), (Y
pos + current_x), DrawProp.TextColor);
00581
00582     /* Initialize the font */
00583     BSP_LCD_SetFont(&LCD_DEFAULT_FONT);
00584
00585     if (decision < 0)
00586     {
00587       decision += (current_x << 2) + 6;
00588     }
00589     else
00590     {
00591       decision += ((current_x - current_y) <
< 2) + 10;
00592       current_y--;
00593     }
00594     current_x++;
00595   }
00596 }
00597
00598 /**
00599   * @brief  Draws an poly-line (between many
 points).
00600   * @param  Points: Pointer to the points ar
ray
00601   * @param  PointCount: Number of points
00602   */
00603 void BSP_LCD_DrawPolygon(pPoint Points, uint
16_t PointCount)
00604 {
00605   int16_t x = 0, y = 0;
```

```
00606
00607   if(PointCount < 2)
00608   {
00609     return;
00610   }
00611
00612   BSP_LCD_DrawLine(Points->X, Points->Y, (Po
ints+PointCount-1)->X, (Points+PointCount-1)->Y);
00613
00614   while(--PointCount)
00615   {
00616     x = Points->X;
00617     y = Points->Y;
00618     Points++;
00619     BSP_LCD_DrawLine(x, y, Points->X, Points
->Y);
00620   }
00621 }
00622
00623 /**
00624   * @brief  Draws an ellipse on LCD.
00625   * @param  Xpos: X position
00626   * @param  Ypos: Y position
00627   * @param  XRadius: Ellipse X radius
00628   * @param  YRadius: Ellipse Y radius
00629   */
00630 void BSP_LCD_DrawEllipse(int Xpos, int Ypos,
int XRadius, int YRadius)
00631 {
00632   int x = 0, y = -YRadius, err = 2-2*XRadius
, e2;
00633   float k = 0, rad1 = 0, rad2 = 0;
00634
00635   rad1 = XRadius;
00636   rad2 = YRadius;
00637
00638   k = (float)(rad2/rad1);
```

```
00639
00640   do {
00641     BSP_LCD_DrawPixel((Xpos-(uint16_t)(x/k))
, (Ypos+y), DrawProp.TextColor);
00642     BSP_LCD_DrawPixel((Xpos+(uint16_t)(x/k))
, (Ypos+y), DrawProp.TextColor);
00643     BSP_LCD_DrawPixel((Xpos+(uint16_t)(x/k))
, (Ypos-y), DrawProp.TextColor);
00644     BSP_LCD_DrawPixel((Xpos-(uint16_t)(x/k))
, (Ypos-y), DrawProp.TextColor);
00645
00646     e2 = err;
00647     if (e2 <= x) {
00648       err += ++x*2+1;
00649       if (-y == x && e2 <= y) e2 = 0;
00650     }
00651     if (e2 > y) err += ++y*2+1;
00652   }
00653   while (y <= 0);
00654 }
00655
00656 /**
00657   * @brief  Draws a bitmap picture (16 bpp).
00658   * @param  Xpos: Bmp X position in the LCD
00659   * @param  Ypos: Bmp Y position in the LCD
00660   * @param  pbmp: Pointer to Bmp picture add
ress.
00661   */
00662 void BSP_LCD_DrawBitmap(uint16_t Xpos, uint1
6_t Ypos, uint8_t *pbmp)
00663 {
00664   uint32_t height = 0;
00665   uint32_t width  = 0;
00666
00667
00668   /* Read bitmap width */
00669   width = *(uint16_t *) (pbmp + 18);
```

```
00670   width |= (*(uint16_t *) (pbmp + 20)) << 16
;
00671
00672   /* Read bitmap height */
00673   height = *(uint16_t *) (pbmp + 22);
00674   height |= (*(uint16_t *) (pbmp + 24)) << 1
6;
00675
00676   SetDisplayWindow(Xpos, Ypos, width, height
);
00677
00678   if(LcdDrv->DrawBitmap != NULL)
00679   {
00680     LcdDrv->DrawBitmap(Xpos, Ypos, pbmp);
00681   }
00682   SetDisplayWindow(0, 0, BSP_LCD_GetXSize(),
BSP_LCD_GetYSize());
00683 }
00684
00685 /**
00686   * @brief  Draws RGB Image (16 bpp).
00687   * @param  Xpos:  X position in the LCD
00688   * @param  Ypos:  Y position in the LCD
00689   * @param  Xsize: X size in the LCD
00690   * @param  Ysize: Y size in the LCD
00691   * @param  pdata: Pointer to the RGB Image
address.
00692   */
00693 void BSP_LCD_DrawRGBImage(uint16_t Xpos, uin
t16_t Ypos, uint16_t Xsize, uint16_t Ysize, uint8_
t *pdata)
00694 {
00695
00696   SetDisplayWindow(Xpos, Ypos, Xsize, Ysize)
;
00697
00698   if(LcdDrv->DrawRGBImage != NULL)
```

```
00699   {
00700     LcdDrv->DrawRGBImage(Xpos, Ypos, Xsize,
Ysize, pdata);
00701   }
00702   SetDisplayWindow(0, 0, BSP_LCD_GetXSize(),
BSP_LCD_GetYSize());
00703 }
00704
00705 /**
00706   * @brief  Draws a full rectangle.
00707   * @param  Xpos: X position
00708   * @param  Ypos: Y position
00709   * @param  Width: Rectangle width
00710   * @param  Height: Rectangle height
00711   */
00712 void BSP_LCD_FillRect(uint16_t Xpos, uint16_
t Ypos, uint16_t Width, uint16_t Height)
00713 {
00714   BSP_LCD_SetTextColor(DrawProp.TextColor);
00715   do
00716   {
00717     BSP_LCD_DrawHLine(Xpos, Ypos++, Width);

00718   }
00719   while(Height--);
00720 }
00721
00722 /**
00723   * @brief  Draws a full circle.
00724   * @param  Xpos: X position
00725   * @param  Ypos: Y position
00726   * @param  Radius: Circle radius
00727   */
00728 void BSP_LCD_FillCircle(uint16_t Xpos, uint1
6_t Ypos, uint16_t Radius)
00729 {
00730   int32_t  decision;        /* Decision Vari
```

```
able */
00731   uint32_t  current_x;    /* Current X Value
 */
00732   uint32_t  current_y;    /* Current Y Value
 */
00733
00734   decision = 3 - (Radius << 1);
00735
00736   current_x = 0;
00737   current_y = Radius;
00738
00739   BSP_LCD_SetTextColor(DrawProp.TextColor);
00740
00741   while (current_x <= current_y)
00742   {
00743     if(current_y > 0)
00744     {
00745       BSP_LCD_DrawHLine(Xpos - current_y, Yp
os + current_x, 2*current_y);
00746       BSP_LCD_DrawHLine(Xpos - current_y, Yp
os - current_x, 2*current_y);
00747     }
00748
00749     if(current_x > 0)
00750     {
00751       BSP_LCD_DrawHLine(Xpos - current_x, Yp
os - current_y, 2*current_x);
00752       BSP_LCD_DrawHLine(Xpos - current_x, Yp
os + current_y, 2*current_x);
00753     }
00754     if (decision < 0)
00755     {
00756       decision += (current_x << 2) + 6;
00757     }
00758     else
00759     {
00760       decision += ((current_x - current_y) <
```

```
< 2) + 10;
00761       current_y--;
00762     }
00763     current_x++;
00764   }
00765
00766   BSP_LCD_SetTextColor(DrawProp.TextColor);
00767   BSP_LCD_DrawCircle(Xpos, Ypos, Radius);
00768 }
00769
00770 /**
00771   * @brief  Draws a full poly-line (between
many points).
00772   * @param  Points: Pointer to the points ar
ray
00773   * @param  PointCount: Number of points
00774   */
00775 void BSP_LCD_FillPolygon(pPoint Points, uint
16_t PointCount)
00776 {
00777   int16_t X = 0, Y = 0, X2 = 0, Y2 = 0, X_ce
nter = 0, Y_center = 0, X_first = 0, Y_first = 0,
pixelX = 0, pixelY = 0, counter = 0;
00778   uint16_t  IMAGE_LEFT = 0, IMAGE_RIGHT = 0,
 IMAGE_TOP = 0, IMAGE_BOTTOM = 0;
00779
00780   IMAGE_LEFT = IMAGE_RIGHT = Points->X;
00781   IMAGE_TOP= IMAGE_BOTTOM = Points->Y;
00782
00783   for(counter = 1; counter < PointCount; cou
nter++)
00784   {
00785     pixelX = POLY_X(counter);
00786     if(pixelX < IMAGE_LEFT)
00787     {
00788       IMAGE_LEFT = pixelX;
00789     }
```

```
00790     if(pixelX > IMAGE_RIGHT)
00791     {
00792       IMAGE_RIGHT = pixelX;
00793     }
00794
00795     pixelY = POLY_Y(counter);
00796     if(pixelY < IMAGE_TOP)
00797     {
00798       IMAGE_TOP = pixelY;
00799     }
00800     if(pixelY > IMAGE_BOTTOM)
00801     {
00802       IMAGE_BOTTOM = pixelY;
00803     }
00804   }
00805
00806   if(PointCount < 2)
00807   {
00808     return;
00809   }
00810
00811   X_center = (IMAGE_LEFT + IMAGE_RIGHT)/2;
00812   Y_center = (IMAGE_BOTTOM + IMAGE_TOP)/2;
00813
00814   X_first = Points->X;
00815   Y_first = Points->Y;
00816
00817   while(--PointCount)
00818   {
00819     X = Points->X;
00820     Y = Points->Y;
00821     Points++;
00822     X2 = Points->X;
00823     Y2 = Points->Y;
00824
00825     FillTriangle(X, X2, X_center, Y, Y2, Y_c
enter);
```

```
00826     FillTriangle(X, X_center, X2, Y, Y_cente
r, Y2);
00827     FillTriangle(X_center, X2, X, Y_center,
Y2, Y);
00828   }
00829
00830   FillTriangle(X_first, X2, X_center, Y_firs
t, Y2, Y_center);
00831   FillTriangle(X_first, X_center, X2, Y_firs
t, Y_center, Y2);
00832   FillTriangle(X_center, X2, X_first, Y_cent
er, Y2, Y_first);
00833 }
00834
00835 /**
00836   * @brief  Draws a full ellipse.
00837   * @param  Xpos: X position
00838   * @param  Ypos: Y position
00839   * @param  XRadius: Ellipse X radius
00840   * @param  YRadius: Ellipse Y radius
00841   */
00842 void BSP_LCD_FillEllipse(int Xpos, int Ypos,
int XRadius, int YRadius)
00843 {
00844   int x = 0, y = -YRadius, err = 2-2*XRadius
, e2;
00845   float k = 0, rad1 = 0, rad2 = 0;
00846
00847   rad1 = XRadius;
00848   rad2 = YRadius;
00849
00850   k = (float)(rad2/rad1);
00851
00852   do
00853   {
00854     BSP_LCD_DrawHLine((Xpos-(uint16_t)(x/k))
, (Ypos+y), (2*(uint16_t)(x/k) + 1));
```

```
    BSP_LCD_DrawHLine((Xpos-(uint16_t)(x/k)), (Ypos-y), (2*(uint16_t)(x/k) + 1));

    e2 = err;
    if (e2 <= x)
    {
      err += ++x*2+1;
      if (-y == x && e2 <= y) e2 = 0;
    }
    if (e2 > y) err += ++y*2+1;
  }
  while (y <= 0);
}

/**
  * @brief  Enables the display.
  */
void BSP_LCD_DisplayOn(void)
{
  LcdDrv->DisplayOn();
}

/**
  * @brief  Disables the display.
  */
void BSP_LCD_DisplayOff(void)
{
  LcdDrv->DisplayOff();
}

/*****************************************************************************
                                Static Functions
*****************************************************************************/

/**
```

```
00889   * @brief  Draws a character on LCD.
00890   * @param  Xpos: Line where to display the
character shape
00891   * @param  Ypos: Start column address
00892   * @param  c: Pointer to the character data
00893   */
00894 static void DrawChar(uint16_t Xpos, uint16_t
 Ypos, const uint8_t *c)
00895 {
00896   uint32_t i = 0, j = 0;
00897   uint16_t height, width;
00898   uint8_t offset;
00899   uint8_t *pchar;
00900   uint32_t line;
00901
00902   height = DrawProp.pFont->Height;
00903   width  = DrawProp.pFont->Width;
00904
00905   offset =  8 *((width + 7)/8) -  width ;
00906
00907   for(i = 0; i < height; i++)
00908   {
00909     pchar = ((uint8_t *)c + (width + 7)/8 *
i);
00910
00911     switch(((width + 7)/8))
00912     {
00913     case 1:
00914       line =  pchar[0];
00915       break;
00916
00917     case 2:
00918       line =  (pchar[0]<< 8) | pchar[1];
00919       break;
00920
00921     case 3:
00922     default:
```

```
00923       line =  (pchar[0]<< 16) | (pchar[1]<<
8) | pchar[2];
00924       break;
00925     }
00926
00927     for (j = 0; j < width; j++)
00928     {
00929       if(line & (1 << (width- j + offset- 1)
))
00930       {
00931         BSP_LCD_DrawPixel((Xpos + j), Ypos,
DrawProp.TextColor);
00932       }
00933       else
00934       {
00935         BSP_LCD_DrawPixel((Xpos + j), Ypos,
DrawProp.BackColor);
00936       }
00937     }
00938     Ypos++;
00939   }
00940 }
00941
00942 /**
00943   * @brief  Sets display window.
00944   * @param  Xpos: LCD X position
00945   * @param  Ypos: LCD Y position
00946   * @param  Width: LCD window width
00947   * @param  Height: LCD window height
00948   */
00949 static void SetDisplayWindow(uint16_t Xpos,
uint16_t Ypos, uint16_t Width, uint16_t Height)
00950 {
00951   if(LcdDrv->SetDisplayWindow != NULL)
00952   {
00953     LcdDrv->SetDisplayWindow(Xpos, Ypos, Wid
th, Height);
```

```
00954   }
00955 }
00956
00957 /**
00958   * @brief  Fills a triangle (between 3 poin
ts).
00959   * @param  x1: Point 1 X position
00960   * @param  y1: Point 1 Y position
00961   * @param  x2: Point 2 X position
00962   * @param  y2: Point 2 Y position
00963   * @param  x3: Point 3 X position
00964   * @param  y3: Point 3 Y position
00965   */
00966 static void FillTriangle(uint16_t x1, uint16
_t x2, uint16_t x3, uint16_t y1, uint16_t y2, uint
16_t y3)
00967 {
00968   int16_t deltax = 0, deltay = 0, x = 0, y =
 0, xinc1 = 0, xinc2 = 0,
00969   yinc1 = 0, yinc2 = 0, den = 0, num = 0, nu
madd = 0, numpixels = 0,
00970   curpixel = 0;
00971
00972   deltax = ABS(x2 - x1);        /* The diffe
rence between the x's */
00973   deltay = ABS(y2 - y1);        /* The diffe
rence between the y's */
00974   x = x1;                       /* Start x o
ff at the first pixel */
00975   y = y1;                       /* Start y o
ff at the first pixel */
00976
00977   if (x2 >= x1)                 /* The x-val
ues are increasing */
00978   {
00979     xinc1 = 1;
00980     xinc2 = 1;
```

```
00981   }
00982   else                            /* The x-val
ues are decreasing */
00983   {
00984     xinc1 = -1;
00985     xinc2 = -1;
00986   }
00987
00988   if (y2 >= y1)                   /* The y-val
ues are increasing */
00989   {
00990     yinc1 = 1;
00991     yinc2 = 1;
00992   }
00993   else                            /* The y-val
ues are decreasing */
00994   {
00995     yinc1 = -1;
00996     yinc2 = -1;
00997   }
00998
00999   if (deltax >= deltay)           /* There is
at least one x-value for every y-value */
01000   {
01001     xinc1 = 0;                    /* Don't cha
nge the x when numerator >= denominator */
01002     yinc2 = 0;                    /* Don't cha
nge the y for every iteration */
01003     den = deltax;
01004     num = deltax / 2;
01005     numadd = deltay;
01006     numpixels = deltax;           /* There are
 more x-values than y-values */
01007   }
01008   else                            /* There is
at least one y-value for every x-value */
01009   {
```

```
01010     xinc2 = 0;                    /* Don't cha
nge the x for every iteration */
01011     yinc1 = 0;                    /* Don't cha
nge the y when numerator >= denominator */
01012     den = deltay;
01013     num = deltay / 2;
01014     numadd = deltax;
01015     numpixels = deltay;           /* There are
 more y-values than x-values */
01016   }
01017
01018   for (curpixel = 0; curpixel <= numpixels;
curpixel++)
01019   {
01020     BSP_LCD_DrawLine(x, y, x3, y3);
01021
01022     num += numadd;                /* Increase
the numerator by the top of the fraction */
01023     if (num >= den)               /* Check if
numerator >= denominator */
01024     {
01025       num -= den;                 /* Calculate
 the new numerator value */
01026       x += xinc1;                 /* Change th
e x as appropriate */
01027       y += yinc1;                 /* Change th
e y as appropriate */
01028     }
01029     x += xinc2;                   /* Change th
e x as appropriate */
01030     y += yinc2;                   /* Change th
e y as appropriate */
01031   }
01032 }
01033
01034 /**
01035   * @}
```

```
01036   */
01037
01038 /**
01039   * @}
01040   */
01041
01042 /**
01043   * @}
01044   */
01045
01046 /**
01047   * @}
01048   */
01049
01050 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM32446E_EVAL BSP User Manual

Data Structures | Defines | Typedefs | Enumerations

STM32446E EVAL LCD Exported Constants

STM32446E EVAL LCD

Data Structures

struct	**Point**

Defines

#define	**LCD_OK** ((uint8_t)0x00) LCD status structure definition.
#define	**LCD_ERROR** ((uint8_t)0x01)
#define	**LCD_TIMEOUT** ((uint8_t)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_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 127 of file stm32446e_eval_lcd.h.

#define LCD_COLOR_BLUE ((uint16_t)0x001F)

LCD color.

Definition at line 106 of file stm32446e_eval_lcd.h.

#define LCD_COLOR_BROWN ((uint16_t)0xA145)

Definition at line 128 of file stm32446e_eval_lcd.h.

#define LCD_COLOR_CYAN ((uint16_t)0x07FF)

Definition at line 109 of file stm32446e_eval_lcd.h.

#define LCD_COLOR_DARKBLUE ((uint16_t)0x0010)

Definition at line 117 of file stm32446e_eval_lcd.h.

#define LCD_COLOR_DARKCYAN ((uint16_t)0x0410)

Definition at line 120 of file stm32446e_eval_lcd.h.

#define LCD_COLOR_DARKGRAY ((uint16_t)0x4208)

Definition at line **126** of file **stm32446e_eval_lcd.h**.

#define LCD_COLOR_DARKGREEN ((uint16_t)0x0400)

Definition at line **118** of file **stm32446e_eval_lcd.h**.

#define LCD_COLOR_DARKMAGENTA ((uint16_t)0x8010)

Definition at line **121** of file **stm32446e_eval_lcd.h**.

#define LCD_COLOR_DARKRED ((uint16_t)0x8000)

Definition at line **119** of file **stm32446e_eval_lcd.h**.

#define LCD_COLOR_DARKYELLOW ((uint16_t)0x8400)

Definition at line **122** of file **stm32446e_eval_lcd.h**.

#define LCD_COLOR_GRAY ((uint16_t)0x8410)

Definition at line **125** of file **stm32446e_eval_lcd.h**.

#define LCD_COLOR_GREEN ((uint16_t)0x07E0)

Definition at line **107** of file **stm32446e_eval_lcd.h**.

#define LCD_COLOR_LIGHTBLUE ((uint16_t)0x841F)

Definition at line **112** of file **stm32446e_eval_lcd.h**.

#define LCD_COLOR_LIGHTGRAY ((uint16_t)0xD69A)

Definition at line 124 of file stm32446e_eval_lcd.h.

#define LCD_COLOR_LIGHTGREEN ((uint16_t)0x87F0)

Definition at line 113 of file stm32446e_eval_lcd.h.

#define LCD_COLOR_LIGHTMAGENTA ((uint16_t)0xFC1F)

Definition at line 115 of file stm32446e_eval_lcd.h.

#define LCD_COLOR_LIGHTRED ((uint16_t)0xFC10)

Definition at line 114 of file stm32446e_eval_lcd.h.

#define LCD_COLOR_LIGHTYELLOW ((uint16_t)0xFFF0)

Definition at line 116 of file stm32446e_eval_lcd.h.

#define LCD_COLOR_MAGENTA ((uint16_t)0xF81F)

Definition at line 110 of file stm32446e_eval_lcd.h.

#define LCD_COLOR_ORANGE ((uint16_t)0xFD20)

Definition at line 129 of file stm32446e_eval_lcd.h.

#define LCD_COLOR_RED ((uint16_t)0xF800)

Definition at line 108 of file stm32446e_eval_lcd.h.

#define LCD_COLOR_WHITE ((uint16_t)0xFFFF)

Definition at line 123 of file stm32446e_eval_lcd.h.

#define LCD_COLOR_YELLOW ((uint16_t)0xFFE0)

Definition at line 111 of file stm32446e_eval_lcd.h.

#define LCD_DEFAULT_FONT Font24

LCD default font.

Definition at line 134 of file stm32446e_eval_lcd.h.

Referenced by BSP_LCD_DrawCircle(), and BSP_LCD_Init().

#define LCD_ERROR ((uint8_t)0x01)

Definition at line 84 of file stm32446e_eval_lcd.h.

Referenced by BSP_LCD_Init().

#define LCD_OK ((uint8_t)0x00)

LCD status structure definition.

Definition at line 83 of file stm32446e_eval_lcd.h.

Referenced by **BSP_LCD_DeInit()**, and **BSP_LCD_Init()**.

#define LCD_TIMEOUT ((uint8_t)0x02)

Definition at line **85** of file **stm32446e_eval_lcd.h**.

Typedef Documentation

typedef struct Point * pPoint

Enumeration Type Documentation

enum Line_ModeTypdef

Line mode structures definition.

Enumerator:

CENTER_MODE

RIGHT_MODE

LEFT_MODE

Definition at line **96** of file **stm32446e_eval_lcd.h**.

STM32446E_EVAL BSP User Manual

STM32446E EVAL QSPI Exported Types

STM32446E EVAL QSPI

Data Structures

struct	**QSPI_Info**

STM32446E_EVAL BSP User Manual

stm32446e_eval_qspi.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval_qspi.h
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm32446e_eval_qspi.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 STM32446E_EVAL
00044   * @{
00045   */
00046
00047 /* Define to prevent recursive inclusion ---
-----------------------------------*/
00048 #ifndef __STM32446E_EVAL_QSPI_H
00049 #define __STM32446E_EVAL_QSPI_H
00050
00051 #ifdef __cplusplus
00052  extern "C" {
00053 #endif
00054
00055 /* Includes ------------------------------
-----------------------------------*/
00056 #include "stm32f4xx_hal.h"
00057 #include "../Components/n25q256a/n25q256a.h"
00058
00059 /** @defgroup STM32446E_EVAL_QSPI STM32446E
EVAL QSPI
00060   * @{
00061   */
00062
00063
00064 /* Exported constants --------------------
-----------------------------------*/
00065 /** @defgroup STM32446E_EVAL_QSPI_Exported_C
onstants STM32446E EVAL QSPI Exported Constants
```

```
00066   * @{
00067   */
00068 /* QSPI Error codes */
00069 #define QSPI_OK            ((uint8_t)0x00)
00070 #define QSPI_ERROR         ((uint8_t)0x01)
00071 #define QSPI_BUSY          ((uint8_t)0x02)
00072 #define QSPI_NOT_SUPPORTED ((uint8_t)0x04)
00073 #define QSPI_SUSPENDED     ((uint8_t)0x08)
00074
00075
00076 /* Definition for QSPI clock resources */
00077 #define QSPI_CLK_ENABLE()           __HAL_RC
C_QSPI_CLK_ENABLE()
00078 #define QSPI_CLK_DISABLE()          __HAL_RC
C_QSPI_CLK_DISABLE()
00079 #define QSPI_CS_GPIO_CLK_ENABLE()   __HAL_RC
C_GPIOG_CLK_ENABLE()
00080 #define QSPI_CS_GPIO_CLK_DISABLE()  __HAL_RC
C_GPIOG_CLK_DISABLE()
00081 #if !defined(USE_STM32446E_EVAL_REVA)
00082 #define QSPI_CLK_GPIO_CLK_ENABLE()  __HAL_RC
C_GPIOD_CLK_ENABLE()
00083 #define QSPI_CLK_GPIO_CLK_DISABLE() __HAL_RC
C_GPIOD_CLK_DISABLE()
00084 #else
00085 #define QSPI_CLK_GPIO_CLK_ENABLE()  __HAL_RC
C_GPIOB_CLK_ENABLE()
00086 #define QSPI_CLK_GPIO_CLK_DISABLE() __HAL_RC
C_GPIOB_CLK_DISABLE()
00087 #endif // !USE_STM32446E_EVAL_REVA
00088 #define QSPI_Dx_GPIO_CLK_ENABLE()   __HAL_RC
C_GPIOF_CLK_ENABLE()
00089 #define QSPI_Dx_GPIO_CLK_DISABLE()  __HAL_RC
C_GPIOF_CLK_DISABLE()
00090
00091 #define QSPI_FORCE_RESET()          __HAL_RC
C_QSPI_FORCE_RESET()
```

```
00092 #define QSPI_RELEASE_RESET()          __HAL_RC
C_QSPI_RELEASE_RESET()
00093
00094 /* Definition for QSPI Pins */
00095 #define QSPI_CS_PIN                GPIO_PIN_6

00096 #define QSPI_CS_GPIO_PORT          GPIOG
00097 #if !defined(USE_STM32446E_EVAL_REVA)
00098 #define QSPI_CLK_PIN               GPIO_PIN_3

00099 #define QSPI_CLK_GPIO_PORT         GPIOD
00100 #else
00101 #define QSPI_CLK_PIN               GPIO_PIN_2

00102 #define QSPI_CLK_GPIO_PORT         GPIOB
00103 #endif // !USE_STM32446E_EVAL_REVA
00104 #define QSPI_D0_PIN                GPIO_PIN_8

00105 #define QSPI_D0_GPIO_PORT          GPIOF
00106 #define QSPI_D1_PIN                GPIO_PIN_9

00107 #define QSPI_D1_GPIO_PORT          GPIOF
00108 #define QSPI_D2_PIN                GPIO_PIN_7

00109 #define QSPI_D2_GPIO_PORT          GPIOF
00110 #define QSPI_D3_PIN                GPIO_PIN_6

00111 #define QSPI_D3_GPIO_PORT          GPIOF
00112
00113
00114 /**
00115   * @}
00116   */
00117
00118 /* Exported types ------------------------
-----------------------------------*/
00119 /** @defgroup STM32446E_EVAL_QSPI_Exported_T
```

```
ypes STM32446E EVAL QSPI Exported Types
00120   * @{
00121   */
00122 /* QSPI Info */
00123 typedef struct {
00124   uint32_t FlashSize;          /*!< Size of
the flash */
00125   uint32_t EraseSectorSize;    /*!< Size of
sectors for the erase operation */
00126   uint32_t EraseSectorsNumber; /*!< Number o
f sectors for the erase operation */
00127   uint32_t ProgPageSize;       /*!< Size of
pages for the program operation */
00128   uint32_t ProgPagesNumber;    /*!< Number o
f pages for the program operation */
00129 } QSPI_Info;
00130
00131 /**
00132   * @}
00133   */
00134
00135
00136 /* Exported functions ---------------------
-----------------------------------*/
00137 /** @defgroup STM32446E_EVAL_QSPI_Exported_F
unctions  STM32446E EVAL QSPI Exported Functions
00138   * @{
00139   */
00140 uint8_t BSP_QSPI_Init       (void);
00141 uint8_t BSP_QSPI_DeInit     (void);
00142 uint8_t BSP_QSPI_Read       (uint8_t* pData,
 uint32_t ReadAddr, uint32_t Size);
00143 uint8_t BSP_QSPI_Write      (uint8_t* pData,
 uint32_t WriteAddr, uint32_t Size);
00144 uint8_t BSP_QSPI_Erase_Block(uint32_t BlockA
ddress);
00145 uint8_t BSP_QSPI_Erase_Chip (void);
```

```
00146 uint8_t BSP_QSPI_GetStatus  (void);
00147 uint8_t BSP_QSPI_GetInfo    (QSPI_Info* pInf
o);
00148 uint8_t BSP_QSPI_MemoryMappedMode(void);
00149
00150 /* These function can be modified in case th
e current settings (e.g. DMA stream)
00151    need to be changed for specific applicati
on needs */
00152 void BSP_QSPI_MspInit(QSPI_HandleTypeDef *hq
spi, void *Params);
00153 void BSP_QSPI_MspDeInit(QSPI_HandleTypeDef *
hqspi, void *Params);
00154
00155 /**
00156   * @}
00157   */
00158
00159 /**
00160   * @}
00161   */
00162
00163 #ifdef __cplusplus
00164 }
00165 #endif
00166
00167 #endif /* __STM32446E_EVAL_QSPI_H */
00168 /**
00169   * @}
00170   */
00171
00172 /**
00173   * @}
00174   */
00175
00176 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```



Generated on Fri Jan 15 2016 10:06:21 for STM32446E_EVAL BSP
User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

stm32446e_eval_qspi.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval_qspi.c
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file includes a standard driver for the N25Q256A QSPI
00008   *          memory mounted on STM32446E-EVAL board.
00009   @verbatim
00010   ==============================================================================
00011                         ##### How to use this driver #####
00012   ==============================================================================
00013   [..]
00014    (#) This driver is used to drive the N25Q256A QSPI external
00015        memory mounted on STM32446E-EVAL eval
```

```
uation board.
00016
00017    (#) This driver need a specific component
 driver (N25Q256A) to be included with.
00018
00019    (#) Initialization steps:
00020        (++) Initialize the QPSI external mem
ory using the BSP_QSPI_Init() function. This
00021             function includes the MSP layer
hardware resources initialization and the
00022             QSPI interface with the external
 memory.
00023
00024    (#) QSPI memory operations
00025        (++) QSPI memory can be accessed with
 read/write operations once it is
00026             initialized.
00027             Read/write operation can be perf
ormed with AHB access using the functions
00028             BSP_QSPI_Read()/BSP_QSPI_Write()
.
00029        (++) The function BSP_QSPI_GetInfo()
returns the configuration of the QSPI memory.
00030             (see the QSPI memory data sheet)
00031        (++) Perform erase block operation us
ing the function BSP_QSPI_Erase_Block() and by
00032             specifying the block address. Yo
u can perform an erase operation of the whole
00033             chip by calling the function BSP
_QSPI_Erase_Chip().
00034        (++) The function BSP_QSPI_GetStatus(
) returns the current status of the QSPI memory.
00035             (see the QSPI memory data sheet)
00036   @endverbatim
00037   ******************************************
**********************************
00038   * @attention
```

```
00039   *
00040   * <h2><center>&copy; COPYRIGHT(c) 2015 STM
icroelectronics</center></h2>
00041   *
00042   * Redistribution and use in source and bin
ary forms, with or without modification,
00043   * are permitted provided that the followin
g conditions are met:
00044   *   1. Redistributions of source code must
 retain the above copyright notice,
00045   *      this list of conditions and the fol
lowing disclaimer.
00046   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00047   *      this list of conditions and the fol
lowing disclaimer in the documentation
00048   *      and/or other materials provided wit
h the distribution.
00049   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00050   *      may be used to endorse or promote p
roducts derived from this software
00051   *      without specific prior written perm
ission.
00052   *
00053   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00054   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00055   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00056   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00057   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00058   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
```

```
00059   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00060   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00061   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00062   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00063   *
00064   ****************************************
**********************************
00065   */
00066
00067 /* Includes ------------------------------
-----------------------------------*/
00068 #include "stm32446e_eval_qspi.h"
00069
00070 /** @addtogroup BSP
00071   * @{
00072   */
00073
00074 /** @addtogroup STM32446E_EVAL
00075   * @{
00076   */
00077
00078 /** @defgroup STM32446E_EVAL_QSPI STM32446E
EVAL QSPI
00079   * @{
00080   */
00081
00082
00083 /* Private variables ---------------------
-----------------------------------*/
00084
00085 /** @defgroup STM32446E_EVAL_QSPI_Private_Va
riables STM32446E EVAL QSPI Private Variables
00086   * @{
```

```
00087   */
00088 QSPI_HandleTypeDef QSPIHandle;
00089
00090 /**
00091   * @}
00092   */
00093
00094
00095
00096 /* Private functions ---------------------
---------------------------------*/
00097
00098 /** @defgroup STM32446E_EVAL_QSPI_Private_Fu
nctions STM32446E EVAL QSPI Private Functions
00099   * @{
00100   */
00101 static uint8_t QSPI_ResetMemory          (QS
PI_HandleTypeDef *hqspi);
00102 static uint8_t QSPI_EnterFourBytesAddress(QS
PI_HandleTypeDef *hqspi);
00103 static uint8_t QSPI_DummyCyclesCfg       (QS
PI_HandleTypeDef *hqspi);
00104 static uint8_t QSPI_WriteEnable          (QS
PI_HandleTypeDef *hqspi);
00105 static uint8_t QSPI_AutoPollingMemReady(QSPI
_HandleTypeDef *hqspi, uint32_t Timeout);
00106
00107 /**
00108   * @}
00109   */
00110
00111 /** @defgroup STM32446E_EVAL_QSPI_Exported_F
unctions STM32446E EVAL QSPI Exported Functions
00112   * @{
00113   */
00114
00115 /**
```

```
00116   * @brief  Initializes the QSPI interface.
00117   * @retval QSPI memory status
00118   */
00119 uint8_t BSP_QSPI_Init(void)
00120 {
00121   QSPIHandle.Instance = QUADSPI;
00122
00123   /* Call the DeInit function to reset the d
river */
00124   if (HAL_QSPI_DeInit(&QSPIHandle) != HAL_OK
)
00125   {
00126     return QSPI_ERROR;
00127   }
00128
00129   /* System level initialization */
00130   BSP_QSPI_MspInit(&QSPIHandle, NULL);
00131
00132   /* QSPI initialization */
00133   QSPIHandle.Init.ClockPrescaler     = 1; /*
 QSPI freq = 180 MHz/(1+1) = 90 Mhz */
00134   QSPIHandle.Init.FifoThreshold      = 4;
00135   QSPIHandle.Init.SampleShifting     = QSPI_
SAMPLE_SHIFTING_HALFCYCLE;
00136   QSPIHandle.Init.FlashSize          = POSIT
ION_VAL(N25Q256A_FLASH_SIZE) - 1;
00137   QSPIHandle.Init.ChipSelectHighTime = QSPI_
CS_HIGH_TIME_2_CYCLE;
00138   QSPIHandle.Init.ClockMode          = QSPI_
CLOCK_MODE_0;
00139   QSPIHandle.Init.FlashID            = QSPI_
FLASH_ID_1;
00140   QSPIHandle.Init.DualFlash          = QSPI_
DUALFLASH_DISABLE;
00141
00142   if (HAL_QSPI_Init(&QSPIHandle) != HAL_OK)
00143   {
```

```
00144     return QSPI_ERROR;
00145   }
00146
00147   /* QSPI memory reset */
00148   if (QSPI_ResetMemory(&QSPIHandle) != QSPI_
OK)
00149   {
00150     return QSPI_NOT_SUPPORTED;
00151   }
00152
00153   /* Set the QSPI memory in 4-bytes address
mode */
00154   if (QSPI_EnterFourBytesAddress(&QSPIHandle
) != QSPI_OK)
00155   {
00156     return QSPI_NOT_SUPPORTED;
00157   }
00158
00159   /* Configuration of the dummy cycles on QS
PI memory side */
00160   if (QSPI_DummyCyclesCfg(&QSPIHandle) != QS
PI_OK)
00161   {
00162     return QSPI_NOT_SUPPORTED;
00163   }
00164
00165   return QSPI_OK;
00166 }
00167
00168 /**
00169   * @brief  De-Initializes the QSPI interfac
e.
00170   * @retval QSPI memory status
00171   */
00172 uint8_t BSP_QSPI_DeInit(void)
00173 {
00174   QSPIHandle.Instance = QUADSPI;
```

```
00175
00176   /* Call the DeInit function to reset the d
river */
00177   if (HAL_QSPI_DeInit(&QSPIHandle) != HAL_OK
)
00178   {
00179     return QSPI_ERROR;
00180   }
00181
00182   /* System level De-initialization */
00183   BSP_QSPI_MspDeInit(&QSPIHandle, NULL);
00184
00185   return QSPI_OK;
00186 }
00187
00188 /**
00189   * @brief  Reads an amount of data from the
 QSPI memory.
00190   * @param  pData: Pointer to data to be read

00191   * @param  ReadAddr: Read start address
00192   * @param  Size: Size of data to read
00193   * @retval QSPI memory status
00194   */
00195 uint8_t BSP_QSPI_Read(uint8_t* pData, uint32
_t ReadAddr, uint32_t Size)
00196 {
00197   QSPI_CommandTypeDef s_command;
00198
00199   /* Initialize the read command */
00200   s_command.InstructionMode   = QSPI_INSTRUC
TION_1_LINE;
00201   s_command.Instruction       = QUAD_INOUT_F
AST_READ_CMD;
00202   s_command.AddressMode       = QSPI_ADDRESS
_4_LINES;
00203   s_command.AddressSize       = QSPI_ADDRESS
```

```
_32_BITS;
00204   s_command.Address           = ReadAddr;
00205   s_command.AlternateByteMode = QSPI_ALTERNA
TE_BYTES_NONE;
00206   s_command.DataMode          = QSPI_DATA_4_
LINES;
00207   s_command.DummyCycles       = N25Q256A_DUM
MY_CYCLES_READ_QUAD;
00208   s_command.NbData            = Size;
00209   s_command.DdrMode           = QSPI_DDR_MOD
E_DISABLE;
00210   s_command.DdrHoldHalfCycle  = QSPI_DDR_HHC
_ANALOG_DELAY;
00211   s_command.SIOOMode          = QSPI_SIOO_IN
ST_EVERY_CMD;
00212
00213   /* Configure the command */
00214   if (HAL_QSPI_Command(&QSPIHandle, &s_comma
nd, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00215   {
00216     return QSPI_ERROR;
00217   }
00218
00219   /* Reception of the data */
00220   if (HAL_QSPI_Receive(&QSPIHandle, pData, H
AL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00221   {
00222     return QSPI_ERROR;
00223   }
00224
00225   return QSPI_OK;
00226 }
00227
00228 /**
00229   * @brief  Writes an amount of data to the
QSPI memory.
00230   * @param  pData: Pointer to data to be wri
```

```
tten
00231   * @param  WriteAddr: Write start address
00232   * @param  Size: Size of data to write
00233   * @retval QSPI memory status
00234   */
00235 uint8_t BSP_QSPI_Write(uint8_t* pData, uint3
2_t WriteAddr, uint32_t Size)
00236 {
00237   QSPI_CommandTypeDef s_command;
00238   uint32_t end_addr, current_size, current_a
ddr;
00239
00240   /* Calculation of the size between the wri
te address and the end of the page */
00241   current_addr = 0;
00242
00243   while (current_addr <= WriteAddr)
00244   {
00245     current_addr += N25Q256A_PAGE_SIZE;
00246   }
00247   current_size = current_addr - WriteAddr;
00248
00249   /* Check if the size of the data is less t
han the remaining place in the page */
00250   if (current_size > Size)
00251   {
00252     current_size = Size;
00253   }
00254
00255   /* Initialize the adress variables */
00256   current_addr = WriteAddr;
00257   end_addr = WriteAddr + Size;
00258
00259   /* Initialize the program command */
00260   s_command.InstructionMode   = QSPI_INSTRUC
TION_1_LINE;
00261   s_command.Instruction       = EXT_QUAD_IN_
```

```
FAST_PROG_CMD;
00262   s_command.AddressMode       = QSPI_ADDRESS
_4_LINES;
00263   s_command.AddressSize       = QSPI_ADDRESS
_32_BITS;
00264   s_command.AlternateByteMode = QSPI_ALTERNA
TE_BYTES_NONE;
00265   s_command.DataMode          = QSPI_DATA_4_
LINES;
00266   s_command.DummyCycles       = 0;
00267   s_command.DdrMode           = QSPI_DDR_MOD
E_DISABLE;
00268   s_command.DdrHoldHalfCycle  = QSPI_DDR_HHC
_ANALOG_DELAY;
00269   s_command.SIOOMode          = QSPI_SIOO_IN
ST_EVERY_CMD;
00270
00271   /* Perform the write page by page */
00272   do
00273   {
00274     s_command.Address = current_addr;
00275     s_command.NbData  = current_size;
00276
00277     /* Enable write operations */
00278     if (QSPI_WriteEnable(&QSPIHandle) != QSP
I_OK)
00279     {
00280       return QSPI_ERROR;
00281     }
00282
00283     /* Configure the command */
00284     if (HAL_QSPI_Command(&QSPIHandle, &s_com
mand, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00285     {
00286       return QSPI_ERROR;
00287     }
00288
```

```
00289     /* Transmission of the data */
00290     if (HAL_QSPI_Transmit(&QSPIHandle, pData
, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00291     {
00292       return QSPI_ERROR;
00293     }
00294
00295     /* Configure automatic polling mode to w
ait for end of program */
00296     if (QSPI_AutoPollingMemReady(&QSPIHandle
, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != QSPI_OK)
00297     {
00298       return QSPI_ERROR;
00299     }
00300
00301     /* Update the address and size variables
 for next page programming */
00302     current_addr += current_size;
00303     pData += current_size;
00304     current_size = ((current_addr + N25Q256A
_PAGE_SIZE) > end_addr) ? (end_addr - current_addr
) : N25Q256A_PAGE_SIZE;
00305   } while (current_addr < end_addr);
00306
00307   return QSPI_OK;
00308 }
00309
00310 /**
00311   * @brief  Erases the specified block of th
e QSPI memory.
00312   * @param  BlockAddress: Block address to e
rase
00313   * @retval QSPI memory status
00314   */
00315 uint8_t BSP_QSPI_Erase_Block(uint32_t BlockA
ddress)
00316 {
```

```
00317   QSPI_CommandTypeDef s_command;
00318
00319   /* Initialize the erase command */
00320   s_command.InstructionMode   = QSPI_INSTRUC
TION_1_LINE;
00321   s_command.Instruction       = SUBSECTOR_ER
ASE_CMD;
00322   s_command.AddressMode       = QSPI_ADDRESS
_1_LINE;
00323   s_command.AddressSize       = QSPI_ADDRESS
_32_BITS;
00324   s_command.Address           = BlockAddress
;
00325   s_command.AlternateByteMode = QSPI_ALTERNA
TE_BYTES_NONE;
00326   s_command.DataMode          = QSPI_DATA_NO
NE;
00327   s_command.DummyCycles       = 0;
00328   s_command.DdrMode           = QSPI_DDR_MOD
E_DISABLE;
00329   s_command.DdrHoldHalfCycle  = QSPI_DDR_HHC
_ANALOG_DELAY;
00330   s_command.SIOOMode          = QSPI_SIOO_IN
ST_EVERY_CMD;
00331
00332   /* Enable write operations */
00333   if (QSPI_WriteEnable(&QSPIHandle) != QSPI_
OK)
00334   {
00335     return QSPI_ERROR;
00336   }
00337
00338   /* Send the command */
00339   if (HAL_QSPI_Command(&QSPIHandle, &s_comma
nd, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00340   {
00341     return QSPI_ERROR;
```

```
00342   }
00343
00344   /* Configure automatic polling mode to wai
t for end of erase */
00345   if (QSPI_AutoPollingMemReady(&QSPIHandle,
N25Q256A_SUBSECTOR_ERASE_MAX_TIME) != QSPI_OK)
00346   {
00347     return QSPI_ERROR;
00348   }
00349
00350   return QSPI_OK;
00351 }
00352
00353 /**
00354   * @brief  Erases the entire QSPI memory.
00355   * @retval QSPI memory status
00356   */
00357 uint8_t BSP_QSPI_Erase_Chip(void)
00358 {
00359   QSPI_CommandTypeDef s_command;
00360
00361   /* Initialize the erase command */
00362   s_command.InstructionMode   = QSPI_INSTRUC
TION_1_LINE;
00363   s_command.Instruction       = BULK_ERASE_C
MD;
00364   s_command.AddressMode       = QSPI_ADDRESS
_NONE;
00365   s_command.AlternateByteMode = QSPI_ALTERNA
TE_BYTES_NONE;
00366   s_command.DataMode          = QSPI_DATA_NO
NE;
00367   s_command.DummyCycles       = 0;
00368   s_command.DdrMode           = QSPI_DDR_MOD
E_DISABLE;
00369   s_command.DdrHoldHalfCycle  = QSPI_DDR_HHC
_ANALOG_DELAY;
```

```
00370   s_command.SIOOMode          = QSPI_SIOO_IN
ST_EVERY_CMD;
00371
00372   /* Enable write operations */
00373   if (QSPI_WriteEnable(&QSPIHandle) != QSPI_
OK)
00374   {
00375     return QSPI_ERROR;
00376   }
00377
00378   /* Send the command */
00379   if (HAL_QSPI_Command(&QSPIHandle, &s_comma
nd, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00380   {
00381     return QSPI_ERROR;
00382   }
00383
00384   /* Configure automatic polling mode to wai
t for end of erase */
00385   if (QSPI_AutoPollingMemReady(&QSPIHandle,
N25Q256A_BULK_ERASE_MAX_TIME) != QSPI_OK)
00386   {
00387     return QSPI_ERROR;
00388   }
00389
00390   return QSPI_OK;
00391 }
00392
00393 /**
00394   * @brief  Reads current status of the QSPI
 memory.
00395   * @retval QSPI memory status
00396   */
00397 uint8_t BSP_QSPI_GetStatus(void)
00398 {
00399   QSPI_CommandTypeDef s_command;
00400   uint8_t reg;
```

```
00401
00402   /* Initialize the read flag status registe
r command */
00403   s_command.InstructionMode   = QSPI_INSTRUC
TION_1_LINE;
00404   s_command.Instruction       = READ_FLAG_ST
ATUS_REG_CMD;
00405   s_command.AddressMode       = QSPI_ADDRESS
_NONE;
00406   s_command.AlternateByteMode = QSPI_ALTERNA
TE_BYTES_NONE;
00407   s_command.DataMode          = QSPI_DATA_1_
LINE;
00408   s_command.DummyCycles       = 0;
00409   s_command.NbData            = 1;
00410   s_command.DdrMode           = QSPI_DDR_MOD
E_DISABLE;
00411   s_command.DdrHoldHalfCycle  = QSPI_DDR_HHC
_ANALOG_DELAY;
00412   s_command.SIOOMode          = QSPI_SIOO_IN
ST_EVERY_CMD;
00413
00414   /* Configure the command */
00415   if (HAL_QSPI_Command(&QSPIHandle, &s_comma
nd, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00416   {
00417     return QSPI_ERROR;
00418   }
00419
00420   /* Reception of the data */
00421   if (HAL_QSPI_Receive(&QSPIHandle, &reg, HA
L_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00422   {
00423     return QSPI_ERROR;
00424   }
00425
00426   /* Check the value of the register */
```

```
00427   if ((reg & (N25Q256A_FSR_PRERR | N25Q256A_
FSR_VPPERR | N25Q256A_FSR_PGERR | N25Q256A_FSR_ERE
RR)) != 0)
00428   {
00429     return QSPI_ERROR;
00430   }
00431   else if ((reg & (N25Q256A_FSR_PGSUS | N25Q
256A_FSR_ERSUS)) != 0)
00432   {
00433     return QSPI_SUSPENDED;
00434   }
00435   else if ((reg & N25Q256A_FSR_READY) != 0)
00436   {
00437     return QSPI_OK;
00438   }
00439   else
00440   {
00441     return QSPI_BUSY;
00442   }
00443 }
00444
00445 /**
00446   * @brief  Return the configuration of the
QSPI memory.
00447   * @param  pInfo: pointer on the configurat
ion structure
00448   * @retval QSPI memory status
00449   */
00450 uint8_t BSP_QSPI_GetInfo(QSPI_Info* pInfo)
00451 {
00452   /* Configure the structure with the memory
 configuration */
00453   pInfo->FlashSize          = N25Q256A_FLASH
_SIZE;
00454   pInfo->EraseSectorSize    = N25Q256A_SUBSE
CTOR_SIZE;
00455   pInfo->EraseSectorsNumber = (N25Q256A_FLAS
```

```
H_SIZE/N25Q256A_SUBSECTOR_SIZE);
00456   pInfo->ProgPageSize        = N25Q256A_PAGE_
SIZE;
00457   pInfo->ProgPagesNumber     = (N25Q256A_FLAS
H_SIZE/N25Q256A_PAGE_SIZE);
00458
00459   return QSPI_OK;
00460 }
00461
00462 /**
00463   * @brief  Configure the QSPI in memory-map
ped mode
00464   * @retval QSPI memory status
00465   */
00466 uint8_t BSP_QSPI_MemoryMappedMode(void)
00467 {
00468   QSPI_CommandTypeDef      s_command;
00469   QSPI_MemoryMappedTypeDef s_mem_mapped_cfg;
00470
00471   /* Configure the command for the read inst
ruction */
00472   s_command.InstructionMode   = QSPI_INSTRUC
TION_1_LINE;
00473   s_command.Instruction       = QUAD_INOUT_F
AST_READ_CMD;
00474   s_command.AddressMode       = QSPI_ADDRESS
_4_LINES;
00475   s_command.AddressSize       = QSPI_ADDRESS
_32_BITS;
00476   s_command.AlternateByteMode = QSPI_ALTERNA
TE_BYTES_NONE;
00477   s_command.DataMode          = QSPI_DATA_4_
LINES;
00478   s_command.DummyCycles       = N25Q256A_DUM
MY_CYCLES_READ_QUAD;
00479   s_command.DdrMode           = QSPI_DDR_MOD
E_DISABLE;
```

```
00480   s_command.DdrHoldHalfCycle  = QSPI_DDR_HHC
_ANALOG_DELAY;
00481   s_command.SIOOMode          = QSPI_SIOO_IN
ST_EVERY_CMD;
00482
00483   /* Configure the memory mapped mode */
00484   s_mem_mapped_cfg.TimeOutActivation = QSPI_
TIMEOUT_COUNTER_ENABLE;
00485   s_mem_mapped_cfg.TimeOutPeriod     = 1;
00486
00487   if (HAL_QSPI_MemoryMapped(&QSPIHandle, &s_
command, &s_mem_mapped_cfg) != HAL_OK)
00488   {
00489     return QSPI_ERROR;
00490   }
00491
00492   return QSPI_OK;
00493 }
00494
00495 /**
00496   * @}
00497   */
00498
00499 /** @addtogroup STM32446E_EVAL_QSPI_Private_
Functions
00500   * @{
00501   */
00502
00503 /**
00504   * @brief QSPI MSP Initialization
00505   *        This function configures the hard
ware resources used in this example:
00506   *           - Peripheral's clock enable
00507   *           - Peripheral's GPIO Configurat
ion
00508   *           - NVIC configuration for QSPI
interrupt
```

```
00509   * @retval None
00510   */
00511 __weak void BSP_QSPI_MspInit(QSPI_HandleType
Def *hqspi, void *Params)
00512 {
00513   GPIO_InitTypeDef gpio_init_structure;
00514
00515   /*##-1- Enable peripherals and GPIO Clocks
 #################################*/
00516   /* Enable the QuadSPI memory interface clo
ck */
00517   QSPI_CLK_ENABLE();
00518   /* Reset the QuadSPI memory interface */
00519   QSPI_FORCE_RESET();
00520   QSPI_RELEASE_RESET();
00521   /* Enable GPIO clocks */
00522   QSPI_CS_GPIO_CLK_ENABLE();
00523   QSPI_CLK_GPIO_CLK_ENABLE();
00524   QSPI_Dx_GPIO_CLK_ENABLE();
00525
00526   /*##-2- Configure peripheral GPIO ########
#################################*/
00527   /* QSPI CS GPIO pin configuration  */
00528   gpio_init_structure.Pin       = QSPI_CS_PIN
;
00529   gpio_init_structure.Mode      = GPIO_MODE_
AF_PP;
00530   gpio_init_structure.Pull      = GPIO_PULLU
P;
00531   gpio_init_structure.Speed     = GPIO_SPEED
_HIGH;
00532   gpio_init_structure.Alternate = GPIO_AF10_
QSPI;
00533   HAL_GPIO_Init(QSPI_CS_GPIO_PORT, &gpio_ini
t_structure);
00534
00535   /* QSPI CLK GPIO pin configuration  */
```

```
00536   gpio_init_structure.Pin       = QSPI_CLK_P
IN;
00537   gpio_init_structure.Pull      = GPIO_NOPUL
L;
00538   gpio_init_structure.Alternate = GPIO_AF9_Q
SPI;
00539   HAL_GPIO_Init(QSPI_CLK_GPIO_PORT, &gpio_in
it_structure);
00540
00541   /* QSPI D0 GPIO pin configuration  */
00542   gpio_init_structure.Pin       = QSPI_D0_PIN
;
00543   gpio_init_structure.Alternate = GPIO_AF10_
QSPI;
00544   HAL_GPIO_Init(QSPI_D0_GPIO_PORT, &gpio_ini
t_structure);
00545
00546   /* QSPI D1 GPIO pin configuration  */
00547   gpio_init_structure.Pin       = QSPI_D1_PIN
;
00548   gpio_init_structure.Alternate = GPIO_AF10_
QSPI;
00549   HAL_GPIO_Init(QSPI_D1_GPIO_PORT, &gpio_ini
t_structure);
00550
00551   /* QSPI D2 GPIO pin configuration  */
00552   gpio_init_structure.Pin       = QSPI_D2_PIN
;
00553   gpio_init_structure.Alternate = GPIO_AF9_Q
SPI;
00554   HAL_GPIO_Init(QSPI_D2_GPIO_PORT, &gpio_ini
t_structure);
00555
00556   /* QSPI D3 GPIO pin configuration  */
00557   gpio_init_structure.Pin       = QSPI_D3_PIN
;
00558   gpio_init_structure.Alternate = GPIO_AF9_Q
```

```
SPI;
00559   HAL_GPIO_Init(QSPI_D3_GPIO_PORT, &gpio_ini
t_structure);
00560 }
00561
00562
00563 /**
00564   * @brief QSPI MSP De-Initialization
00565   *        This function frees the hardware
resources used in this example:
00566   *          - Disable the Peripheral's clock

00567   *          - Revert GPIO and NVIC configur
ation to their default state
00568   * @retval None
00569   */
00570 __weak void BSP_QSPI_MspDeInit(QSPI_HandleTy
peDef *hqspi, void *Params)
00571 {
00572     /*##-1- Disable the NVIC for QSPI ######
#####################################*/
00573     HAL_NVIC_DisableIRQ(QUADSPI_IRQn);
00574
00575     /*##-2- Disable peripherals and GPIO Clo
cks ################################*/
00576     /* De-Configure QSPI specific pins (not
common to any other blocks) */
00577     HAL_GPIO_DeInit(QSPI_D0_GPIO_PORT, QSPI_
D0_PIN);
00578     HAL_GPIO_DeInit(QSPI_D1_GPIO_PORT, QSPI_
D1_PIN);
00579     HAL_GPIO_DeInit(QSPI_D2_GPIO_PORT, QSPI_
D2_PIN);
00580     HAL_GPIO_DeInit(QSPI_D3_GPIO_PORT, QSPI_
D3_PIN);
00581
00582     /*##-3- Reset peripherals ##############
```

```
####################################*/
00583     /* Reset the QuadSPI memory interface */
00584     QSPI_FORCE_RESET();
00585     QSPI_RELEASE_RESET();
00586
00587     /* Disable the QuadSPI memory interface
clock */
00588     QSPI_CLK_DISABLE();
00589 }
00590
00591 /**
00592   * @brief  This function reset the QSPI mem
ory.
00593   * @param  hqspi: QSPI handle
00594   * @retval None
00595   */
00596 static uint8_t QSPI_ResetMemory(QSPI_HandleT
ypeDef *hqspi)
00597 {
00598   QSPI_CommandTypeDef s_command;
00599
00600   /* Initialize the reset enable command */
00601   s_command.InstructionMode   = QSPI_INSTRUC
TION_1_LINE;
00602   s_command.Instruction       = RESET_ENABLE
_CMD;
00603   s_command.AddressMode       = QSPI_ADDRESS
_NONE;
00604   s_command.AlternateByteMode = QSPI_ALTERNA
TE_BYTES_NONE;
00605   s_command.DataMode          = QSPI_DATA_NO
NE;
00606   s_command.DummyCycles       = 0;
00607   s_command.DdrMode           = QSPI_DDR_MOD
E_DISABLE;
00608   s_command.DdrHoldHalfCycle  = QSPI_DDR_HHC
_ANALOG_DELAY;
```

```
00609   s_command.SIOOMode          = QSPI_SIOO_IN
ST_EVERY_CMD;
00610
00611   /* Send the command */
00612   if (HAL_QSPI_Command(&QSPIHandle, &s_comma
nd, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00613   {
00614     return QSPI_ERROR;
00615   }
00616
00617   /* Send the reset memory command */
00618   s_command.Instruction = RESET_MEMORY_CMD;
00619   if (HAL_QSPI_Command(&QSPIHandle, &s_comma
nd, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00620   {
00621     return QSPI_ERROR;
00622   }
00623
00624   /* Configure automatic polling mode to wai
t the memory is ready */
00625   if (QSPI_AutoPollingMemReady(&QSPIHandle,
HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != QSPI_OK)
00626   {
00627     return QSPI_ERROR;
00628   }
00629
00630   return QSPI_OK;
00631 }
00632
00633 /**
00634   * @brief  This function set the QSPI memor
y in 4-byte address mode
00635   * @param  hqspi: QSPI handle
00636   * @retval None
00637   */
00638 static uint8_t QSPI_EnterFourBytesAddress(QS
PI_HandleTypeDef *hqspi)
```

```
00639 {
00640   QSPI_CommandTypeDef s_command;
00641
00642   /* Initialize the command */
00643   s_command.InstructionMode   = QSPI_INSTRUC
TION_1_LINE;
00644   s_command.Instruction       = ENTER_4_BYTE
_ADDR_MODE_CMD;
00645   s_command.AddressMode       = QSPI_ADDRESS
_NONE;
00646   s_command.AlternateByteMode = QSPI_ALTERNA
TE_BYTES_NONE;
00647   s_command.DataMode          = QSPI_DATA_NO
NE;
00648   s_command.DummyCycles       = 0;
00649   s_command.DdrMode           = QSPI_DDR_MOD
E_DISABLE;
00650   s_command.DdrHoldHalfCycle  = QSPI_DDR_HHC
_ANALOG_DELAY;
00651   s_command.SIOOMode          = QSPI_SIOO_IN
ST_EVERY_CMD;
00652
00653   /* Enable write operations */
00654   if (QSPI_WriteEnable(&QSPIHandle) != QSPI_
OK)
00655   {
00656     return QSPI_ERROR;
00657   }
00658
00659   /* Send the command */
00660   if (HAL_QSPI_Command(&QSPIHandle, &s_comma
nd, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00661   {
00662     return QSPI_ERROR;
00663   }
00664
00665   /* Configure automatic polling mode to wai
```

```
t the memory is ready */
00666   if (QSPI_AutoPollingMemReady(&QSPIHandle,
HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != QSPI_OK)
00667   {
00668     return QSPI_ERROR;
00669   }
00670
00671   return QSPI_OK;
00672 }
00673
00674 /**
00675   * @brief  This function configure the dumm
y cycles on memory side.
00676   * @param  hqspi: QSPI handle
00677   * @retval None
00678   */
00679 static uint8_t QSPI_DummyCyclesCfg(QSPI_Hand
leTypeDef *hqspi)
00680 {
00681   QSPI_CommandTypeDef s_command;
00682   uint8_t reg;
00683
00684   /* Initialize the read volatile configurat
ion register command */
00685   s_command.InstructionMode   = QSPI_INSTRUC
TION_1_LINE;
00686   s_command.Instruction       = READ_VOL_CFG
_REG_CMD;
00687   s_command.AddressMode       = QSPI_ADDRESS
_NONE;
00688   s_command.AlternateByteMode = QSPI_ALTERNA
TE_BYTES_NONE;
00689   s_command.DataMode          = QSPI_DATA_1_
LINE;
00690   s_command.DummyCycles       = 0;
00691   s_command.NbData            = 1;
00692   s_command.DdrMode           = QSPI_DDR_MOD
```

```
E_DISABLE;
00693   s_command.DdrHoldHalfCycle  = QSPI_DDR_HHC
_ANALOG_DELAY;
00694   s_command.SIOOMode          = QSPI_SIOO_IN
ST_EVERY_CMD;
00695
00696   /* Configure the command */
00697   if (HAL_QSPI_Command(&QSPIHandle, &s_comma
nd, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00698   {
00699     return QSPI_ERROR;
00700   }
00701
00702   /* Reception of the data */
00703   if (HAL_QSPI_Receive(&QSPIHandle, &reg, HA
L_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00704   {
00705     return QSPI_ERROR;
00706   }
00707
00708   /* Enable write operations */
00709   if (QSPI_WriteEnable(&QSPIHandle) != QSPI_
OK)
00710   {
00711     return QSPI_ERROR;
00712   }
00713
00714   /* Update volatile configuration register
(with new dummy cycles) */
00715   s_command.Instruction = WRITE_VOL_CFG_REG_
CMD;
00716   MODIFY_REG(reg, N25Q256A_VCR_NB_DUMMY, (N2
5Q256A_DUMMY_CYCLES_READ_QUAD << POSITION_VAL(N25Q
256A_VCR_NB_DUMMY)));
00717
00718   /* Configure the write volatile configurat
ion register command */
```

```
00719   if (HAL_QSPI_Command(&QSPIHandle, &s_comma
nd, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00720   {
00721     return QSPI_ERROR;
00722   }
00723
00724   /* Transmission of the data */
00725   if (HAL_QSPI_Transmit(&QSPIHandle, &reg, H
AL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00726   {
00727     return QSPI_ERROR;
00728   }
00729
00730   return QSPI_OK;
00731 }
00732
00733 /**
00734   * @brief  This function send a Write Enabl
e and wait it is effective.
00735   * @param  hqspi: QSPI handle
00736   * @retval None
00737   */
00738 static uint8_t QSPI_WriteEnable(QSPI_HandleT
ypeDef *hqspi)
00739 {
00740   QSPI_CommandTypeDef     s_command;
00741   QSPI_AutoPollingTypeDef s_config;
00742
00743   /* Enable write operations */
00744   s_command.InstructionMode   = QSPI_INSTRUC
TION_1_LINE;
00745   s_command.Instruction       = WRITE_ENABLE
_CMD;
00746   s_command.AddressMode       = QSPI_ADDRESS
_NONE;
00747   s_command.AlternateByteMode = QSPI_ALTERNA
TE_BYTES_NONE;
```

```
00748   s_command.DataMode          = QSPI_DATA_NO
NE;
00749   s_command.DummyCycles       = 0;
00750   s_command.DdrMode           = QSPI_DDR_MOD
E_DISABLE;
00751   s_command.DdrHoldHalfCycle  = QSPI_DDR_HHC
_ANALOG_DELAY;
00752   s_command.SIOOMode          = QSPI_SIOO_IN
ST_EVERY_CMD;
00753
00754   if (HAL_QSPI_Command(&QSPIHandle, &s_comma
nd, HAL_QPSI_TIMEOUT_DEFAULT_VALUE) != HAL_OK)
00755   {
00756     return QSPI_ERROR;
00757   }
00758
00759   /* Configure automatic polling mode to wai
t for write enabling */
00760   s_config.Match           = N25Q256A_SR_WRE
N;
00761   s_config.Mask            = N25Q256A_SR_WRE
N;
00762   s_config.MatchMode       = QSPI_MATCH_MODE
_AND;
00763   s_config.StatusBytesSize = 1;
00764   s_config.Interval        = 0x10;
00765   s_config.AutomaticStop   = QSPI_AUTOMATIC_
STOP_ENABLE;
00766
00767   s_command.Instruction    = READ_STATUS_REG
_CMD;
00768   s_command.DataMode       = QSPI_DATA_1_LIN
E;
00769
00770   if (HAL_QSPI_AutoPolling(&QSPIHandle, &s_c
ommand, &s_config, HAL_QPSI_TIMEOUT_DEFAULT_VALUE)
 != HAL_OK)
```

```
00771   {
00772     return QSPI_ERROR;
00773   }
00774
00775   return QSPI_OK;
00776 }
00777
00778 /**
00779   * @brief  This function read the SR of the
 memory and wait the EOP.
00780   * @param  hqspi: QSPI handle
00781   * @param  Timeout: timeout value
00782   * @retval None
00783   */
00784 static uint8_t QSPI_AutoPollingMemReady(QSPI
_HandleTypeDef *hqspi, uint32_t Timeout)
00785 {
00786   QSPI_CommandTypeDef     s_command;
00787   QSPI_AutoPollingTypeDef s_config;
00788
00789   /* Configure automatic polling mode to wai
t for memory ready */
00790   s_command.InstructionMode   = QSPI_INSTRUC
TION_1_LINE;
00791   s_command.Instruction       = READ_STATUS_
REG_CMD;
00792   s_command.AddressMode       = QSPI_ADDRESS
_NONE;
00793   s_command.AlternateByteMode = QSPI_ALTERNA
TE_BYTES_NONE;
00794   s_command.DataMode          = QSPI_DATA_1_
LINE;
00795   s_command.DummyCycles       = 0;
00796   s_command.DdrMode           = QSPI_DDR_MOD
E_DISABLE;
00797   s_command.DdrHoldHalfCycle  = QSPI_DDR_HHC
_ANALOG_DELAY;
```

```
00798   s_command.SIOOMode          = QSPI_SIOO_IN
ST_EVERY_CMD;
00799
00800   s_config.Match           = 0;
00801   s_config.Mask            = N25Q256A_SR_WIP
;
00802   s_config.MatchMode       = QSPI_MATCH_MODE
_AND;
00803   s_config.StatusBytesSize = 1;
00804   s_config.Interval        = 0x10;
00805   s_config.AutomaticStop   = QSPI_AUTOMATIC_
STOP_ENABLE;
00806
00807   if (HAL_QSPI_AutoPolling(&QSPIHandle, &s_c
ommand, &s_config, Timeout) != HAL_OK)
00808   {
00809     return QSPI_ERROR;
00810   }
00811
00812   return QSPI_OK;
00813 }
00814 /**
00815   * @}
00816   */
00817
00818 /**
00819   * @}
00820   */
00821
00822 /**
00823   * @}
00824   */
00825
00826 /**
00827   * @}
00828   */
00829
```

```
00830 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
00831
```

STM32446E_EVAL BSP User Manual

Modules | Defines

STM32446E EVAL SD

STM32446E EVAL

Modules

- STM32446E EVAL SD Private TypesDefinitions
- STM32446E EVAL SD Private Defines
- STM32446E EVAL SD Private Macros
- STM32446E EVAL SD Private Variables
- STM32446E EVAL SD Private FunctionPrototypes
- STM32446E EVAL SD Private Functions
- STM32446E EVAL SD Exported Types
- STM32446E EVAL SD Exported Constants
- STM32446E EVAL SD Exported Macro
- STM32446E EVAL SD Exported Functions

Defines

#define	**MSD_OK** ((uint8_t)0x00) SD status structure definition.
#define	**MSD_ERROR** ((uint8_t)0x01)
#define	**MSD_ERROR_SD_NOT_PRESENT** ((uint8_t)0x02)

Define Documentation

#define MSD_ERROR ((uint8_t)0x01)

Definition at line 79 of file stm32446e_eval_sd.h.

Referenced by BSP_SD_DeInit(), BSP_SD_Erase(), BSP_SD_Init(), BSP_SD_ReadBlocks(), BSP_SD_ReadBlocks_DMA(), BSP_SD_WriteBlocks(), and BSP_SD_WriteBlocks_DMA().

#define MSD_ERROR_SD_NOT_PRESENT ((uint8_t)0x02)

Definition at line 80 of file stm32446e_eval_sd.h.

Referenced by BSP_SD_Init().

#define MSD_OK ((uint8_t)0x00)

SD status structure definition.

Definition at line 78 of file stm32446e_eval_sd.h.

Referenced by BSP_SD_DeInit(), BSP_SD_Erase(), BSP_SD_Init(), BSP_SD_ReadBlocks(), BSP_SD_ReadBlocks_DMA(), BSP_SD_WriteBlocks(), and BSP_SD_WriteBlocks_DMA().

STM32446E_EVAL BSP User Manual

STM32446E EVAL SDRAM

STM32446E EVAL

Modules

- STM32446E EVAL SDRAM Private Types Definitions
- STM32446E EVAL SDRAM Private Defines
- STM32446E EVAL SDRAM Private Macros
- STM32446E EVAL SDRAM Private Variables
- STM32446E EVAL SDRAM Private Function Prototypes
- STM32446E EVAL SDRAM Private Functions
- STM32446E EVAL SDRAM Exported Types
- STM32446E EVAL SDRAM Exported Constants
- STM32446E EVAL SDRAM Exported Macro
- STM32446E EVAL SDRAM Exported Functions

STM32446E_EVAL BSP User Manual

STM32446E EVAL TS

STM32446E EVAL

Modules

- STM32446E EVAL TS Private Types Definitions
- STM32446E EVAL TS Private Defines
- STM32446E EVAL TS Private Macros
- STM32446E EVAL TS Private Variables
- STM32446E EVAL TS Private Function Prototypes
- STM32446E EVAL TS Private Functions
- STM32446E EVAL TS Exported Types
- STM32446E EVAL TS Exported Constants
- STM32446E EVAL TS Exported Macros
- STM32446E EVAL TS Exported Functions

STM32446E_EVAL BSP User Manual

STM32446E EVAL TS Exported Types

STM32446E EVAL TS

Data Structures

struct **TS_StateTypeDef**

STM32446E_EVAL BSP User Manual

stm32446e_eval_ts.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval_ts.h
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm32446e_eval_ts.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 __STM32446E_EVAL_TS_H
00041 #define __STM32446E_EVAL_TS_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------------------------------------------*/
00048 #include "stm32446e_eval.h"
00049 /* Include IOExpander(MFX) component Driver */
00050 #include "../Components/mfxstm32l152/mfxstm32l152.h"
00051
00052 /** @addtogroup BSP
00053   * @{
00054   */
00055
00056 /** @addtogroup STM32446E_EVAL
00057   * @{
00058   */
00059
00060 /** @defgroup STM32446E_EVAL_TS STM32446E EVAL TS
00061   * @{
00062   */
00063
00064 /** @defgroup STM32446E_EVAL_TS_Exported_Types STM32446E EVAL TS Exported Types
00065   * @{
```

```
00066   */
00067 typedef struct
00068 {
00069   uint16_t TouchDetected;
00070   uint16_t x;
00071   uint16_t y;
00072   uint16_t z;
00073 }TS_StateTypeDef;
00074 /**
00075   * @}
00076   */
00077
00078 /** @defgroup STM32446E_EVAL_TS_Exported_Con
stants STM32446E EVAL TS Exported Constants
00079   * @{
00080   */
00081 #define TS_SWAP_NONE                    ((ui
nt8_t)0x00)
00082 #define TS_SWAP_X                       ((ui
nt8_t)0x01)
00083 #define TS_SWAP_Y                       ((ui
nt8_t)0x02)
00084 #define TS_SWAP_XY                      ((ui
nt8_t)0x04)
00085
00086 typedef enum
00087 {
00088   TS_OK       = 0x00,
00089   TS_ERROR    = 0x01,
00090   TS_TIMEOUT  = 0x02
00091 }TS_StatusTypeDef;
00092
00093 /**
00094   * @}
00095   */
00096
00097 /** @defgroup STM32446E_EVAL_TS_Exported_Mac
```

```
ros STM32446E EVAL TS Exported Macros
00098   * @{
00099   */
00100 /**
00101   * @}
00102   */
00103
00104 /** @defgroup STM32446E_EVAL_TS_Exported_Fun
ctions  STM32446E EVAL TS Exported Functions
00105   * @{
00106   */
00107 uint8_t BSP_TS_Init(uint16_t xSize, uint16_t
 ySize);
00108 uint8_t BSP_TS_DeInit(void);
00109 uint8_t BSP_TS_GetState(TS_StateTypeDef *TS_
State);
00110 uint8_t BSP_TS_ITEnable(void);
00111 uint8_t BSP_TS_ITDisable(void);
00112 uint8_t BSP_TS_ITConfig(void);
00113 uint8_t BSP_TS_ITGetStatus(void);
00114 void    BSP_TS_ITClear(void);
00115 void    BSP_TS_FIFOClear(void);
00116
00117 /**
00118   * @}
00119   */
00120
00121 /**
00122   * @}
00123   */
00124
00125 /**
00126   * @}
00127   */
00128
00129 /**
00130   * @}
```

```
00131   */
00132
00133
00134 #ifdef __cplusplus
00135 }
00136 #endif
00137
00138 #endif /* __STM32446E_EVAL_TS_H */
00139
00140 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM32446E_EVAL BSP User Manual

stm32446e_eval_ts.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval_ts.c
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file provides a set of functions needed to manage the Touch
00008   *          Screen on STM32446E-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 touch
screen module of the STM32446E-EVAL
00044      evaluation board on AMPIRE 480x272 LCD
mounted on MB1046 daughter board.
00045    - the MFXSTM32L152 IO expander device com
ponent
00046      driver must be included in order to run
 the TS module commanded by the IO
00047      expander device, the MFX IO expander de
vice component driver must be
00048      also included in case of interrupt mode
 use of the TS.
00049
00050 2. Driver description:
00051 ---------------------
00052   + Initialization steps:
00053      o Initialize the TS module using the BS
P_TS_Init() function. This
00054        function includes the MSP layer hardw
are resources initialization and the
00055        communication layer configuration to
start the TS use. The LCD size properties
00056        (x and y) are passed as parameters.
00057      o If TS interrupt mode is desired, you
must configure the TS interrupt mode
00058        by calling the function BSP_TS_ITConf
```

```
ig(). The TS interrupt mode is generated
00059        as an external interrupt whenever a t
ouch is detected.
00060        The interrupt mode internally uses th
e IO functionalities driver driven by
00061        the IO expander, to configure the IT
line.
00062
00063   + Touch screen use
00064      o The touch screen state is captured wh
enever the function BSP_TS_GetState() is
00065        used. This function returns informati
on about the last LCD touch occurred
00066        in the TS_StateTypeDef structure.
00067      o If TS interrupt mode is used, the fun
ction BSP_TS_ITGetStatus() is needed to get
00068        the interrupt status. To clear the IT
 pending bits, you should call the
00069        function BSP_TS_ITClear().
00070      o The IT is handled using the correspon
ding external interrupt IRQ handler,
00071        the user IT callback treatment is imp
lemented on the same external interrupt
00072        callback.
00073
00074 ----------------------------------------
----------------------------------*/
00075
00076 /* Includes ------------------------------
----------------------------------*/
00077 #include "stm32446e_eval_ts.h"
00078 #include "stm32446e_eval_io.h"
00079
00080 /** @addtogroup BSP
00081   * @{
00082   */
00083
```

```
00084 /** @addtogroup STM32446E_EVAL
00085   * @{
00086   */
00087
00088 /** @defgroup STM32446E_EVAL_TS STM32446E EV
AL TS
00089   * @{
00090   */
00091
00092 /** @defgroup STM32446E_EVAL_TS_Private_Type
s_Definitions STM32446E EVAL TS Private Types Defi
nitions
00093   * @{
00094   */
00095 /**
00096   * @}
00097   */
00098
00099 /** @defgroup STM32446E_EVAL_TS_Private_Defi
nes STM32446E EVAL TS Private Defines
00100   * @{
00101   */
00102 /**
00103   * @}
00104   */
00105
00106 /** @defgroup STM32446E_EVAL_TS_Private_Macr
os STM32446E EVAL TS Private Macros
00107   * @{
00108   */
00109 /**
00110   * @}
00111   */
00112
00113 /** @defgroup STM32446E_EVAL_TS_Private_Vari
ables STM32446E EVAL TS Private Variables
00114   * @{
```

```
00115   */
00116 static TS_DrvTypeDef *ts_driver;
00117 static uint16_t tsBundaryX, tsBundaryY;
00118 static uint8_t  tsOrientation;
00119 static uint8_t  AddressI2C;
00120 /**
00121   * @}
00122   */
00123
00124 /** @defgroup STM32446E_EVAL_TS_Private_Function_Prototypes STM32446E EVAL TS Private Function Prototypes
00125   * @{
00126   */
00127 /**
00128   * @}
00129   */
00130
00131 /** @defgroup STM32446E_EVAL_TS_Private_Functions STM32446E EVAL TS Private Functions
00132   * @{
00133   */
00134
00135 /**
00136   * @brief  Initializes and configures the touch screen functionalities and
00137   *         configures all necessary hardware resources (GPIOs, clocks..).
00138   * @param  xSize: Maximum X size of the TS area on LCD
00139   * @param  ySize: Maximum Y size of the TS area on LCD
00140   * @retval TS_OK if all initializations are OK. Other value if error.
00141   */
00142 uint8_t BSP_TS_Init(uint16_t xSize, uint16_t ySize)
```

```
00143 {
00144   uint8_t mfxstm32l152_id = 0;
00145 
00146   tsBundaryX = xSize;
00147   tsBundaryY = ySize;
00148 
00149   /* Initialize IO functionalities (MFX) use
d by TS */
00150   BSP_IO_Init();
00151 
00152   /* Read ID and verify if the IO expander i
s ready */
00153   mfxstm32l152_id = mfxstm32l152_io_drv.Read
ID(IO_I2C_ADDRESS);
00154   if((mfxstm32l152_id == MFXSTM32L152_ID_1)
|| (mfxstm32l152_id == MFXSTM32L152_ID_2))
00155   {
00156     /* Initialize the TS driver structure */
00157     ts_driver = &mfxstm32l152_ts_drv;
00158     AddressI2C = TS_I2C_ADDRESS;
00159     tsOrientation = TS_SWAP_NONE;
00160   }
00161 
00162 
00163   /* Initialize the TS driver */
00164   ts_driver->Init(AddressI2C);
00165   ts_driver->Start(AddressI2C);
00166 
00167   return TS_OK;
00168 }
00169 
00170 /**
00171   * @brief  DeInitializes the TouchScreen.
00172   * @retval TS state
00173   */
00174 uint8_t BSP_TS_DeInit(void)
00175 {
```

```
00176   /* Actually ts_driver does not provide a D
eInit function */
00177   return TS_OK;
00178 }
00179
00180 /**
00181   * @brief  Enables the touch screen interru
pts.
00182   * @retval TS_OK if all initializations are
 OK. Other value if error.
00183   */
00184 uint8_t BSP_TS_ITEnable(void)
00185 {
00186   /* Enable the TS ITs */
00187   ts_driver->EnableIT(AddressI2C);
00188   return TS_OK;
00189 }
00190
00191 /**
00192   * @brief  Disables the touch screen interr
upts.
00193   * @retval TS_OK if all initializations are
 OK. Other value if error.
00194   */
00195 uint8_t BSP_TS_ITDisable(void)
00196 {
00197   /* Disable the TS ITs */
00198   ts_driver->DisableIT(AddressI2C);
00199   return TS_OK;
00200 }
00201
00202 /**
00203   * @brief  Configures and enables the touch
 screen interrupts.
00204   * @retval TS_OK if all initializations are
 OK. Other value if error.
00205   */
```

```
00206 uint8_t BSP_TS_ITConfig(void)
00207 {
00208   /* Initialize the IO */
00209   BSP_IO_Init();
00210 
00211   /* Enable the TS ITs */
00212   ts_driver->EnableIT(AddressI2C);
00213 
00214   return TS_OK;
00215 }
00216 
00217 /**
00218   * @brief  Clears all touch screen interrup
ts.
00219   */
00220 void BSP_TS_ITClear(void)
00221 {
00222   /* Clear TS IT pending bits */
00223   ts_driver->ClearIT(AddressI2C);
00224 }
00225 
00226 /**
00227   * @brief  Gets the touch screen interrupt
status.
00228   * @retval TS_OK if all initializations are
 OK. Other value if error.
00229   */
00230 uint8_t BSP_TS_ITGetStatus(void)
00231 {
00232   /* Return the TS IT status */
00233   return (ts_driver->GetITStatus(AddressI2C)
);
00234 }
00235 
00236 /**
00237   * @brief  Clears touch screen FIFO contain
ing 128 x,y values.
```

```
00238   */
00239 void BSP_TS_FIFOClear(void)
00240 {
00241   /* ts.h does not foresee an API that allow
s clearing the FIFO */
00242   /*      reading GetXY currently read one v
alue from the FIFO */
00243   /*      but the FIFO can contain up to 128
 values ...       */
00244   /*      which would mean 128 I2C read.
                  */
00245   /*      Best is to modify ts.h (not done f
or compatibility with other families) */
00246 }
00247
00248 /**
00249   * @brief  Returns status and positions of
the touch screen.
00250   * @param  TS_State: Pointer to touch scree
n current state structure
00251   * @retval TS_OK if all initializations are
 OK. Other value if error.
00252   */
00253 uint8_t BSP_TS_GetState(TS_StateTypeDef *TS_
State)
00254 {
00255   static uint32_t _x = 0, _y = 0;
00256   uint16_t x_diff, y_diff , x , y;
00257   uint16_t swap;
00258
00259   TS_State->TouchDetected = ts_driver->Detec
tTouch(AddressI2C);
00260
00261   if(TS_State->TouchDetected)
00262   {
00263     ts_driver->GetXY(AddressI2C, &x, &y);
00264
```

```
00265     if(tsOrientation & TS_SWAP_X)
00266     {
00267       x = 4096 - x;
00268     }
00269
00270     if(tsOrientation & TS_SWAP_Y)
00271     {
00272       y = 4096 - y;
00273     }
00274
00275     if(tsOrientation & TS_SWAP_XY)
00276     {
00277       swap = y;
00278       y = x;
00279       x = swap;
00280     }
00281
00282     x_diff = x > _x? (x - _x): (_x - x);
00283     y_diff = y > _y? (y - _y): (_y - y);
00284
00285     if (x_diff + y_diff > 5)
00286     {
00287       _x = x;
00288       _y = y;
00289     }
00290
00291     TS_State->x = (tsBundaryX * _x) >> 12;
00292     TS_State->y = (tsBundaryY * _y) >> 12;
00293   }
00294   return TS_OK;
00295 }
00296
00297
00298
00299 /**
00300   * @}
00301   */
```

```
00302
00303 /**
00304   * @}
00305   */
00306
00307 /**
00308   * @}
00309   */
00310
00311 /**
00312   * @}
00313   */
00314
00315 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM32446E_EVAL BSP User Manual

Defines

STM32446E EVAL SDRAM Exported Constants

STM32446E EVAL SDRAM

Defines

#define	**SDRAM_DEVICE_ADDR** ((uint32_t)0xC0000000)
#define	**SDRAM_DEVICE_SIZE** ((uint32_t)0x800000) /* SDRAM devi
#define	**SDRAM_MEMORY_WIDTH** FMC_SDRAM_MEM_BUS_WID
#define	**SDCLOCK_PERIOD** FMC_SDRAM_CLOCK_PERIOD_2
#define	**REFRESH_COUNT** ((uint32_t)0x0569) /* SDRAM refresh cou clock) */
#define	**SDRAM_TIMEOUT** ((uint32_t)0xFFFF)
#define	**__DMAx_CLK_ENABLE** __HAL_RCC_DMA2_CLK_ENABLE
#define	**__DMAx_CLK_DISABLE** __HAL_RCC_DMA2_CLK_DISABL
#define	**SDRAM_DMAx_CHANNEL** DMA_CHANNEL_0
#define	**SDRAM_DMAx_STREAM** DMA2_Stream0
#define	**SDRAM_DMAx_IRQn** DMA2_Stream0_IRQn
#define	**SDRAM_DMAx_IRQHandler** DMA2_Stream0_IRQHandler
#define	**SDRAM_MODEREG_BURST_LENGTH_1** ((uint16_t)0x0000 FMC SDRAM Mode definition register defines.
#define	**SDRAM_MODEREG_BURST_LENGTH_2** ((uint16_t)0x0001
#define	**SDRAM_MODEREG_BURST_LENGTH_4** ((uint16_t)0x0002
#define	**SDRAM_MODEREG_BURST_LENGTH_8** ((uint16_t)0x0004
#define	**SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL** ((uint16
#define	**SDRAM_MODEREG_BURST_TYPE_INTERLEAVED** ((uint1
#define	**SDRAM_MODEREG_CAS_LATENCY_2** ((uint16_t)0x0020)
#define	**SDRAM_MODEREG_CAS_LATENCY_3** ((uint16_t)0x0030)
#define	**SDRAM_MODEREG_OPERATING_MODE_STANDARD** ((ui
#define	**SDRAM_MODEREG_WRITEBURST_MODE_PROGRAMME**
#define	**SDRAM_MODEREG_WRITEBURST_MODE_SINGLE** ((uint

Define Documentation

#define __DMAx_CLK_DISABLE __HAL_RCC_DMA2_CLK_DISAB

Definition at line 94 of file stm32446e_eval_sdram.h.

#define __DMAx_CLK_ENABLE __HAL_RCC_DMA2_CLK_ENABL

Definition at line 93 of file stm32446e_eval_sdram.h.

Referenced by BSP_SDRAM_MspInit().

#define REFRESH_COUNT ((uint32_t)0x0569) /* SDRAM refresh c

Definition at line 88 of file stm32446e_eval_sdram.h.

Referenced by BSP_SDRAM_Init().

#define SDCLOCK_PERIOD FMC_SDRAM_CLOCK_PERIOD_2

Definition at line 85 of file stm32446e_eval_sdram.h.

Referenced by BSP_SDRAM_Init().

#define SDRAM_DEVICE_ADDR ((uint32_t)0xC0000000)

Definition at line 79 of file stm32446e_eval_sdram.h.

#define SDRAM_DEVICE_SIZE ((uint32_t)0x800000) /* SDRAM dev

Definition at line 80 of file stm32446e_eval_sdram.h.

#define SDRAM_DMAx_CHANNEL DMA_CHANNEL_0

Definition at line 95 of file stm32446e_eval_sdram.h.

Referenced by BSP_SDRAM_MspInit().

#define SDRAM_DMAx_IRQHandler DMA2_Stream0_IRQHandler

Definition at line 98 of file stm32446e_eval_sdram.h.

#define SDRAM_DMAx_IRQn DMA2_Stream0_IRQn

Definition at line 97 of file stm32446e_eval_sdram.h.

Referenced by BSP_SDRAM_MspDeInit(), and BSP_SDRAM_MspInit().

#define SDRAM_DMAx_STREAM DMA2_Stream0

Definition at line 96 of file stm32446e_eval_sdram.h.

Referenced by BSP_SDRAM_MspDeInit(), and BSP_SDRAM_MspInit().

#define SDRAM_MEMORY_WIDTH FMC_SDRAM_MEM_BUS_WID

Definition at line 83 of file stm32446e_eval_sdram.h.

Referenced by BSP_SDRAM_Init().

#define SDRAM_MODEREG_BURST_LENGTH_1 ((uint16_t)0x000(

FMC SDRAM Mode definition register defines.

Definition at line **104** of file **stm32446e_eval_sdram.h**.

Referenced by **BSP_SDRAM_Initialization_sequence()**.

#define SDRAM_MODEREG_BURST_LENGTH_2 ((uint16_t)0x0001

Definition at line **105** of file **stm32446e_eval_sdram.h**.

#define SDRAM_MODEREG_BURST_LENGTH_4 ((uint16_t)0x0002

Definition at line **106** of file **stm32446e_eval_sdram.h**.

#define SDRAM_MODEREG_BURST_LENGTH_8 ((uint16_t)0x0004

Definition at line **107** of file **stm32446e_eval_sdram.h**.

#define SDRAM_MODEREG_BURST_TYPE_INTERLEAVED ((uint1

Definition at line **109** of file **stm32446e_eval_sdram.h**.

#define SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL ((uint16

Definition at line **108** of file **stm32446e_eval_sdram.h**.

Referenced by **BSP_SDRAM_Initialization_sequence()**.

#define SDRAM_MODEREG_CAS_LATENCY_2 ((uint16_t)0x0020)

Definition at line **110** of file **stm32446e_eval_sdram.h**.

Referenced by **BSP_SDRAM_Initialization_sequence()**.

#define SDRAM_MODEREG_CAS_LATENCY_3 ((uint16_t)0x0030)

Definition at line **111** of file **stm32446e_eval_sdram.h**.

#define SDRAM_MODEREG_OPERATING_MODE_STANDARD ((ui

Definition at line **112** of file **stm32446e_eval_sdram.h**.

Referenced by **BSP_SDRAM_Initialization_sequence()**.

#define SDRAM_MODEREG_WRITEBURST_MODE_PROGRAMMED

Definition at line **113** of file **stm32446e_eval_sdram.h**.

#define SDRAM_MODEREG_WRITEBURST_MODE_SINGLE ((uint

Definition at line **114** of file **stm32446e_eval_sdram.h**.

Referenced by **BSP_SDRAM_Initialization_sequence()**.

#define SDRAM_TIMEOUT ((uint32_t)0xFFFF)

Definition at line **90** of file **stm32446e_eval_sdram.h**.

Referenced by **BSP_SDRAM_Initialization_sequence()**, and **BSP_SDRAM_Sendcmd()**.

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP
User Manual by  1.7.6.1

STM32446E_EVAL BSP User Manual

Defines

STM32446E EVAL SD Exported Constants

STM32446E EVAL SD

Defines

#define	**SD_PRESENT** ((uint8_t)0x01)
#define	**SD_NOT_PRESENT** ((uint8_t)0x00)
#define	**SD_DATATIMEOUT** ((uint32_t)100000000)
#define	**__DMAx_TxRx_CLK_ENABLE** __HAL_RCC_DMA2_CLK_E
#define	**SD_DMAx_Tx_CHANNEL** DMA_CHANNEL_4
#define	**SD_DMAx_Rx_CHANNEL** DMA_CHANNEL_4
#define	**SD_DMAx_Tx_STREAM** DMA2_Stream6
#define	**SD_DMAx_Rx_STREAM** DMA2_Stream3
#define	**SD_DMAx_Tx_IRQn** DMA2_Stream6_IRQn
#define	**SD_DMAx_Rx_IRQn** DMA2_Stream3_IRQn
#define	**SD_DMAx_Tx_IRQHandler** DMA2_Stream6_IRQHandler
#define	**SD_DMAx_Rx_IRQHandler** DMA2_Stream3_IRQHandler
#define	**SD_DetectIRQHandler**() HAL_GPIO_EXTI_IRQHandler(GPI

Define Documentation

#define __DMAx_TxRx_CLK_ENABLE __HAL_RCC_DMA2_CLK_E

Definition at line 91 of file stm32446e_eval_sd.h.

Referenced by BSP_SD_MspInit().

#define SD_DATATIMEOUT ((uint32_t)100000000)

Definition at line 88 of file stm32446e_eval_sd.h.

Referenced by BSP_SD_ReadBlocks_DMA(), and BSP_SD_WriteBlocks_DMA().

#define SD_DetectIRQHandler ( ) HAL_GPIO_EXTI_IRQHandler(G

Definition at line 100 of file stm32446e_eval_sd.h.

#define SD_DMAx_Rx_CHANNEL DMA_CHANNEL_4

Definition at line 93 of file stm32446e_eval_sd.h.

Referenced by BSP_SD_MspInit().

#define SD_DMAx_Rx_IRQHandler DMA2_Stream3_IRQHandler

Definition at line 99 of file stm32446e_eval_sd.h.

#define SD_DMAx_Rx_IRQn DMA2_Stream3_IRQn

Definition at line **97** of file **stm32446e_eval_sd.h**.

Referenced by **BSP_SD_MspDeInit()**, and **BSP_SD_MspInit()**.

#define SD_DMAx_Rx_STREAM DMA2_Stream3

Definition at line **95** of file **stm32446e_eval_sd.h**.

Referenced by **BSP_SD_MspDeInit()**, and **BSP_SD_MspInit()**.

#define SD_DMAx_Tx_CHANNEL DMA_CHANNEL_4

Definition at line **92** of file **stm32446e_eval_sd.h**.

Referenced by **BSP_SD_MspInit()**.

#define SD_DMAx_Tx_IRQHandler DMA2_Stream6_IRQHandler

Definition at line **98** of file **stm32446e_eval_sd.h**.

#define SD_DMAx_Tx_IRQn DMA2_Stream6_IRQn

Definition at line **96** of file **stm32446e_eval_sd.h**.

Referenced by **BSP_SD_MspDeInit()**, and **BSP_SD_MspInit()**.

#define SD_DMAx_Tx_STREAM DMA2_Stream6

Definition at line **94** of file **stm32446e_eval_sd.h**.

Referenced by **BSP_SD_MspDeInit()**, and **BSP_SD_MspInit()**.

#define SD_NOT_PRESENT ((uint8_t)0x00)

Definition at line 86 of file stm32446e_eval_sd.h.

Referenced by BSP_SD_IsDetected().

#define SD_PRESENT ((uint8_t)0x01)

Definition at line 85 of file stm32446e_eval_sd.h.

Referenced by BSP_SD_Init(), and BSP_SD_IsDetected().

STM32446E_EVAL BSP User Manual

Defines

STM32446E EVAL LOW LEVEL Private Defines

STM32446E EVAL LOW LEVEL

Defines

#define	**__STM32446E_EVAL_BSP_VERSION_MAIN** (0x01) STM32446E EVAL BSP Driver version number V1.1.1.
#define	**__STM32446E_EVAL_BSP_VERSION_SUB1** (0x01)
#define	**__STM32446E_EVAL_BSP_VERSION_SUB2** (0x01)
#define	**__STM32446E_EVAL_BSP_VERSION_RC** (0x00)
#define	**__STM32446E_EVAL_BSP_VERSION**
#define	**FMC_BANK1_BASE** ((uint32_t)(0x60000000 | 0x00000000))
#define	**FMC_BANK3_BASE** ((uint32_t)(0x60000000 | 0x08000000))
#define	**FMC_BANK1** ((**LCD_CONTROLLER_TypeDef** *) **FMC_BANK1_BASE**)

Define Documentation

#define __STM32446E_EVAL_BSP_VERSION

Value:

```
((__STM32446E_EVAL_BSP_VERSION_MAIN << 24)\
                                                |(__
STM32446E_EVAL_BSP_VERSION_SUB1 << 16)\
                                                |(__
STM32446E_EVAL_BSP_VERSION_SUB2 << 8 )\
                                                |(__
STM32446E_EVAL_BSP_VERSION_RC))
```

Definition at line **88** of file **stm32446e_eval.c**.

Referenced by **BSP_GetVersion()**.

#define __STM32446E_EVAL_BSP_VERSION_MAIN (0x01)

STM32446E EVAL BSP Driver version number V1.1.1.

[31:24] main version

Definition at line **84** of file **stm32446e_eval.c**.

#define __STM32446E_EVAL_BSP_VERSION_RC (0x00)

[7:0] release candidate

Definition at line **87** of file **stm32446e_eval.c**.

#define __STM32446E_EVAL_BSP_VERSION_SUB1 (0x01)

[23:16] sub1 version

Definition at line **85** of file **stm32446e_eval.c**.

#define __STM32446E_EVAL_BSP_VERSION_SUB2 (0x01)

[15:8] sub2 version

Definition at line **86** of file **stm32446e_eval.c**.

#define FMC_BANK1 ((LCD_CONTROLLER_TypeDef *) FMC_BAN

Definition at line **96** of file **stm32446e_eval.c**.

Referenced by **FMC_BANK1_ReadData()**, **FMC_BANK1_WriteData()**, and **FMC_BANK1_WriteReg()**.

#define FMC_BANK1_BASE ((uint32_t)(0x60000000 | 0x00000000)

Definition at line **94** of file **stm32446e_eval.c**.

#define FMC_BANK3_BASE ((uint32_t)(0x60000000 | 0x08000000)

Definition at line **95** of file **stm32446e_eval.c**.

STM32446E_EVAL BSP User Manual

Defines

STM32446E EVAL LCD Private Macros

STM32446E EVAL LCD

Defines

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

Define Documentation

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

Definition at line **104** of file **stm32446e_eval_lcd.c**.

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

STM32446E_EVAL BSP User Manual

STM32446E EVAL TS
Private Variables

STM32446E EVAL TS

Variables

static TS_DrvTypeDef *	**ts_driver**
static uint16_t	**tsBundaryX**
static uint16_t	**tsBundaryY**
static uint8_t	**tsOrientation**
static uint8_t	**AddressI2C**

Variable Documentation

uint8_t AddressI2C [static]

Definition at line 119 of file stm32446e_eval_ts.c.

Referenced by BSP_TS_GetState(), BSP_TS_Init(), BSP_TS_ITClear(), BSP_TS_ITConfig(), BSP_TS_ITDisable(), BSP_TS_ITEnable(), and BSP_TS_ITGetStatus().

TS_DrvTypeDef* ts_driver [static]

Definition at line 116 of file stm32446e_eval_ts.c.

Referenced by BSP_TS_GetState(), BSP_TS_Init(), BSP_TS_ITClear(), BSP_TS_ITConfig(), BSP_TS_ITDisable(), BSP_TS_ITEnable(), and BSP_TS_ITGetStatus().

uint16_t tsBundaryX [static]

Definition at line 117 of file stm32446e_eval_ts.c.

Referenced by BSP_TS_GetState(), and BSP_TS_Init().

uint16_t tsBundaryY [static]

Definition at line 117 of file stm32446e_eval_ts.c.

Referenced by BSP_TS_GetState(), and BSP_TS_Init().

uint8_t tsOrientation [static]

Definition at line **118** of file **stm32446e_eval_ts.c**.

Referenced by **BSP_TS_GetState()**, and **BSP_TS_Init()**.

STM32446E_EVAL BSP User Manual

Variables

STM32446E EVAL AUDIO Private Variables

STM32446E EVAL AUDIO

Variables

AUDIO_DrvTypeDef *	**audio_drv**
SAI_HandleTypeDef	**haudio_out_sai**
I2S_HandleTypeDef	**haudio_in_i2s**
TIM_HandleTypeDef	**haudio_tim**
PDMFilter_InitStruct	**Filter** [2]
uint8_t	**Channel_Demux** [128]
uint16_t __IO	**AudioInVolume** = **DEFAULT_AUDIO_IN_VOLUME**

Variable Documentation

AUDIO_DrvTypeDef* audio_drv

Definition at line 149 of file stm32446e_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().

uint16_t __IO AudioInVolume = DEFAULT_AUDIO_IN_VOLUME

Definition at line 174 of file stm32446e_eval_audio.c.

Referenced by BSP_AUDIO_IN_PDMToPCM(), and BSP_AUDIO_IN_SetVolume().

uint8_t Channel_Demux[128]

Initial value:

```
 {
    0x00, 0x01, 0x00, 0x01, 0x02, 0x03, 0x02, 0x0
3,
    0x00, 0x01, 0x00, 0x01, 0x02, 0x03, 0x02, 0x0
3,
    0x04, 0x05, 0x04, 0x05, 0x06, 0x07, 0x06, 0x0
7,
    0x04, 0x05, 0x04, 0x05, 0x06, 0x07, 0x06, 0x0
7,
    0x00, 0x01, 0x00, 0x01, 0x02, 0x03, 0x02, 0x0
3,
    0x00, 0x01, 0x00, 0x01, 0x02, 0x03, 0x02, 0x0
```

```
3,
    0x04, 0x05, 0x04, 0x05, 0x06, 0x07, 0x06, 0x0
7,
    0x04, 0x05, 0x04, 0x05, 0x06, 0x07, 0x06, 0x0
7,
    0x08, 0x09, 0x08, 0x09, 0x0a, 0x0b, 0x0a, 0x0
b,
    0x08, 0x09, 0x08, 0x09, 0x0a, 0x0b, 0x0a, 0x0
b,
    0x0c, 0x0d, 0x0c, 0x0d, 0x0e, 0x0f, 0x0e, 0x0
f,
    0x0c, 0x0d, 0x0c, 0x0d, 0x0e, 0x0f, 0x0e, 0x0
f,
    0x08, 0x09, 0x08, 0x09, 0x0a, 0x0b, 0x0a, 0x0
b,
    0x08, 0x09, 0x08, 0x09, 0x0a, 0x0b, 0x0a, 0x0
b,
    0x0c, 0x0d, 0x0c, 0x0d, 0x0e, 0x0f, 0x0e, 0x0
f,
    0x0c, 0x0d, 0x0c, 0x0d, 0x0e, 0x0f, 0x0e, 0x0
f
}
```

Definition at line **155** of file **stm32446e_eval_audio.c**.

Referenced by **BSP_AUDIO_IN_PDMToPCM()**.

PDMFilter_InitStruct Filter[2]

Definition at line **154** of file **stm32446e_eval_audio.c**.

Referenced by **BSP_AUDIO_IN_PDMToPCM()**, and **PDMDecoder_Init()**.

I2S_HandleTypeDef haudio_in_i2s

Definition at line **151** of file **stm32446e_eval_audio.c**.

Referenced by **BSP_AUDIO_IN_DeInit()**, **BSP_AUDIO_IN_Init()**, **BSP_AUDIO_IN_Pause()**, **BSP_AUDIO_IN_Record()**, **BSP_AUDIO_IN_Resume()**, **BSP_AUDIO_IN_Stop()**, **I2Sx_DeInit()**, and **I2Sx_Init()**.

SAI_HandleTypeDef haudio_out_sai

Definition at line **150** of file **stm32446e_eval_audio.c**.

Referenced by **BSP_AUDIO_OUT_ChangeBuffer()**, **BSP_AUDIO_OUT_DeInit()**, **BSP_AUDIO_OUT_Init()**, **BSP_AUDIO_OUT_Pause()**, **BSP_AUDIO_OUT_Play()**, **BSP_AUDIO_OUT_Resume()**, **BSP_AUDIO_OUT_SetAudioFrameSlot()**, **BSP_AUDIO_OUT_SetFrequency()**, **BSP_AUDIO_OUT_Stop()**, **SAIx_DeInit()**, and **SAIx_Init()**.

TIM_HandleTypeDef haudio_tim

Definition at line **152** of file **stm32446e_eval_audio.c**.

Referenced by **TIMx_DeInit()**, and **TIMx_Init()**.

STM32446E_EVAL BSP User Manual

STM32446E EVAL AUDIO Exported Constants

STM32446E EVAL AUDIO

Modules

STM32446E EVAL Audio Slot TDM mode

In W8994 codec the Audio frame contains 4 slots : TDM Mode TDM format : +-----------------|-----------------|-------------------|-----------------+ | CODEC_SLOT0 Left | CODEC_SLOT1 Left | CODEC_SLOT0 Right | CODEC_SLOT1 Right | +---+.

Defines

```
#define AUDIO_SAIx SAI2_Block_B
#define AUDIO_SAIx_CLK_ENABLE() __HAL_RCC_SAI2_CLK_EN
#define AUDIO_SAIx_CLK_DISABLE() __HAL_RCC_SAI2_CLK_DI
#define AUDIO_SAIx_SCK_AF GPIO_AF8_SAI2
#define AUDIO_SAIx_MCLK_SD_FS_AF GPIO_AF10_SAI2
#define AUDIO_SAIx_MCLK_SCK_ENABLE() __HAL_RCC_GPIOA
#define AUDIO_SAIx_MCLK_SCK_DISABLE() __HAL_RCC_GPIOA
#define AUDIO_SAIx_MCK_PIN GPIO_PIN_1 /* PA1 */
#define AUDIO_SAIx_SCK_PIN GPIO_PIN_2 /* PA2 */
#define AUDIO_SAIx_MCLK_SCK_GPIO_PORT GPIOA
#define AUDIO_SAIx_SD_FS_ENABLE() __HAL_RCC_GPIOG_CLK
#define AUDIO_SAIx_SD_FS_DISABLE() __HAL_RCC_GPIOG_CL
#define AUDIO_SAIx_FS_PIN GPIO_PIN_9 /* PG9 */
#define AUDIO_SAIx_SD_PIN GPIO_PIN_10 /* PG10 */
#define AUDIO_SAIx_SD_FS_GPIO_PORT GPIOG
#define AUDIO_SAIx_DMAx_CLK_ENABLE() __HAL_RCC_DMA2_
#define AUDIO_SAIx_DMAx_CLK_DISABLE() __HAL_RCC_DMA2_
#define AUDIO_SAIx_DMAx_STREAM DMA2_Stream6
#define AUDIO_SAIx_DMAx_CHANNEL DMA_CHANNEL_3
#define AUDIO_SAIx_DMAx_IRQ DMA2_Stream6_IRQn
#define AUDIO_SAIx_DMAx_PERIPH_DATA_SIZE DMA_PDATAAL
#define AUDIO_SAIx_DMAx_MEM_DATA_SIZE DMA_MDATAALIGN
#define DMA_MAX_SZE 0xFFFF
#define AUDIO_SAIx_DMAx_IRQHandler DMA2_Stream6_IRQHan
#define AUDIO_OUT_IRQ_PREPRIO ((uint32_t)5) /* Select the preem
#define AUDIO_I2Sx SPI3
#define AUDIO_I2Sx_CLK_ENABLE() __HAL_RCC_SPI3_CLK_EN
#define AUDIO_I2Sx_CLK_DISABLE() __HAL_RCC_SPI3_CLK_DI
#define AUDIO_I2Sx_SCK_PIN GPIO_PIN_3
#define AUDIO_I2Sx_SCK_GPIO_PORT GPIOB
#define AUDIO_I2Sx_SCK_GPIO_CLK_ENABLE() __HAL_RCC_GP
```

```
#define AUDIO_I2Sx_SCK_GPIO_CLK_DISABLE()   __HAL_RCC_G
#define AUDIO_I2Sx_SCK_AF   GPIO_AF6_SPI3
#define AUDIO_I2Sx_SD_PIN   GPIO_PIN_1
#define AUDIO_I2Sx_SD_GPIO_PORT   GPIOC
#define AUDIO_I2Sx_SD_GPIO_CLK_ENABLE()   __HAL_RCC_GPI
#define AUDIO_I2Sx_SD_GPIO_CLK_DISABLE()   __HAL_RCC_GP
#define AUDIO_I2Sx_SD_AF   GPIO_AF5_SPI3
#define AUDIO_I2Sx_DMAx_CLK_ENABLE()   __HAL_RCC_DMA1_
#define AUDIO_I2Sx_DMAx_CLK_DISABLE()   __HAL_RCC_DMA1_
#define AUDIO_I2Sx_DMAx_STREAM   DMA1_Stream2
#define AUDIO_I2Sx_DMAx_CHANNEL   DMA_CHANNEL_0
#define AUDIO_I2Sx_DMAx_IRQ   DMA1_Stream2_IRQn
#define AUDIO_I2Sx_DMAx_PERIPH_DATA_SIZE   DMA_PDATAALI
#define AUDIO_I2Sx_DMAx_MEM_DATA_SIZE   DMA_MDATAALIGN
#define AUDIO_I2Sx_DMAx_IRQHandler   DMA1_Stream2_IRQHand
#define AUDIO_IN_IRQ_PREPRIO   ((uint32_t)6) /* Select the preemp
#define AUDIO_TIMx_CLK_ENABLE()   __HAL_RCC_TIM4_CLK_EN
#define AUDIO_TIMx_CLK_DISABLE()   __HAL_RCC_TIM4_CLK_DI
#define AUDIO_TIMx   TIM4
#define AUDIO_TIMx_IN_CHANNEL   TIM_CHANNEL_1
#define AUDIO_TIMx_OUT_CHANNEL   TIM_CHANNEL_2 /* Select c
#define AUDIO_TIMx_GPIO_CLK_ENABLE()   __HAL_RCC_GPIOB_
#define AUDIO_TIMx_GPIO_CLK_DISABLE()   __HAL_RCC_GPIOB_
#define AUDIO_TIMx_GPIO_PORT   GPIOB
#define AUDIO_TIMx_IN_GPIO_PIN   GPIO_PIN_6
#define AUDIO_TIMx_OUT_GPIO_PIN   GPIO_PIN_7
#define AUDIO_TIMx_AF   GPIO_AF2_TIM4
#define AUDIODATA_SIZE   2 /* 16-bits audio data size */
#define AUDIO_OK   ((uint8_t)0)
#define AUDIO_ERROR   ((uint8_t)1)
#define AUDIO_TIMEOUT   ((uint8_t)2)
#define DEFAULT_AUDIO_IN_FREQ   I2S_AUDIOFREQ_16K
#define DEFAULT_AUDIO_IN_BIT_RESOLUTION   ((uint8_t)16)
#define DEFAULT_AUDIO_IN_CHANNEL_NBR   ((uint8_t)2) /* Mono
```

#define	**DEFAULT_AUDIO_IN_VOLUME** ((uint16_t)64)
#define	**INTERNAL_BUFF_SIZE** (128***DEFAULT_AUDIO_IN_FREQ**/
#define	**PCM_OUT_SIZE** (**DEFAULT_AUDIO_IN_FREQ**/1000*2)
#define	**CHANNEL_DEMUX_MASK** ((uint8_t)0x55)
#define	**CODEC_RESET_DELAY** ((uint8_t)5)
#define	**OUTPUT_DEVICE_HEADPHONE1** OUTPUT_DEVICE_HEA
#define	**OUTPUT_DEVICE_HEADPHONE2** OUTPUT_DEVICE_SPE output of the wm8994 */

Define Documentation

#define AUDIO_ERROR ((uint8_t)1)

Definition at line 191 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_IN_Record(), BSP_AUDIO_IN_Stop(), 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_I2Sx SPI3

Definition at line 136 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_IN_Init(), BSP_AUDIO_IN_MspDeInit(), BSP_AUDIO_IN_MspInit(), I2Sx_DeInit(), and I2Sx_Init().

#define AUDIO_I2Sx_CLK_DISABLE ( ) __HAL_RCC_SPI3_CLK_D

Definition at line 138 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_IN_MspDeInit().

#define AUDIO_I2Sx_CLK_ENABLE ( ) __HAL_RCC_SPI3_CLK_E

Definition at line 137 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_IN_MspInit().

#define AUDIO_I2Sx_DMAx_CHANNEL DMA_CHANNEL_0

Definition at line 155 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_IN_MspInit().

#define AUDIO_I2Sx_DMAx_CLK_DISABLE ( ) __HAL_RCC_DMA

Definition at line 153 of file stm32446e_eval_audio.h.

#define AUDIO_I2Sx_DMAx_CLK_ENABLE ( ) __HAL_RCC_DMA1

Definition at line 152 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_IN_MspInit().

#define AUDIO_I2Sx_DMAx_IRQ DMA1_Stream2_IRQn

Definition at line 156 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_IN_MspDeInit(), and BSP_AUDIO_IN_MspInit().

#define AUDIO_I2Sx_DMAx_IRQHandler DMA1_Stream2_IRQHan

Definition at line 160 of file stm32446e_eval_audio.h.

#define AUDIO_I2Sx_DMAx_MEM_DATA_SIZE DMA_MDATAALIGI

Definition at line 158 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_IN_MspInit().

#define AUDIO_I2Sx_DMAx_PERIPH_DATA_SIZE DMA_PDATAAL

Definition at line 157 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_IN_MspInit().

#define AUDIO_I2Sx_DMAx_STREAM DMA1_Stream2

Definition at line 154 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_IN_MspInit().

#define AUDIO_I2Sx_SCK_AF GPIO_AF6_SPI3

Definition at line 143 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_IN_MspInit().

#define AUDIO_I2Sx_SCK_GPIO_CLK_DISABLE ( ) __HAL_RCC_

Definition at line 142 of file stm32446e_eval_audio.h.

#define AUDIO_I2Sx_SCK_GPIO_CLK_ENABLE ( ) __HAL_RCC_(

Definition at line 141 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_IN_MspInit().

#define AUDIO_I2Sx_SCK_GPIO_PORT GPIOB

Definition at line 140 of file stm32446e_eval_audio.h.

Referenced by **BSP_AUDIO_IN_MspDeInit()**, and **BSP_AUDIO_IN_MspInit()**.

#define AUDIO_I2Sx_SCK_PIN GPIO_PIN_3

Definition at line **139** of file **stm32446e_eval_audio.h**.

Referenced by **BSP_AUDIO_IN_MspDeInit()**, and **BSP_AUDIO_IN_MspInit()**.

#define AUDIO_I2Sx_SD_AF GPIO_AF5_SPI3

Definition at line **149** of file **stm32446e_eval_audio.h**.

Referenced by **BSP_AUDIO_IN_MspInit()**.

#define AUDIO_I2Sx_SD_GPIO_CLK_DISABLE ( ) __HAL_RCC_G

Definition at line **148** of file **stm32446e_eval_audio.h**.

#define AUDIO_I2Sx_SD_GPIO_CLK_ENABLE ( ) __HAL_RCC_GI

Definition at line **147** of file **stm32446e_eval_audio.h**.

Referenced by **BSP_AUDIO_IN_MspInit()**.

#define AUDIO_I2Sx_SD_GPIO_PORT GPIOC

Definition at line **146** of file **stm32446e_eval_audio.h**.

Referenced by **BSP_AUDIO_IN_MspDeInit()**, and **BSP_AUDIO_IN_MspInit()**.

#define AUDIO_I2Sx_SD_PIN GPIO_PIN_1

Definition at line 145 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_IN_MspDeInit(), and BSP_AUDIO_IN_MspInit().

#define AUDIO_IN_IRQ_PREPRIO ((uint32_t)6) /* Select the preem

Definition at line 163 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_IN_MspInit().

#define AUDIO_OK ((uint8_t)0)

Definition at line 190 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_IN_Init(), BSP_AUDIO_IN_Pause(), BSP_AUDIO_IN_PDMToPCM(), BSP_AUDIO_IN_Record(), BSP_AUDIO_IN_Resume(), BSP_AUDIO_IN_SetVolume(), BSP_AUDIO_IN_Stop(), 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_OUT_IRQ_PREPRIO ((uint32_t)5) /* Select the pree

Definition at line 130 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_OUT_MspInit().

#define AUDIO_SAIx SAI2_Block_B

Definition at line 98 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_OUT_Init(), BSP_AUDIO_OUT_MspDeInit(), BSP_AUDIO_OUT_MspInit(), SAIx_DeInit(), and SAIx_Init().

#define AUDIO_SAIx_CLK_DISABLE ( ) __HAL_RCC_SAI2_CLK_

Definition at line 100 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_OUT_MspDeInit().

#define AUDIO_SAIx_CLK_ENABLE ( ) __HAL_RCC_SAI2_CLK_E

Definition at line 99 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_OUT_MspInit().

#define AUDIO_SAIx_DMAx_CHANNEL DMA_CHANNEL_3

Definition at line 121 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_OUT_MspInit().

#define AUDIO_SAIx_DMAx_CLK_DISABLE ( ) __HAL_RCC_DMA

Definition at line 119 of file stm32446e_eval_audio.h.

#define AUDIO_SAIx_DMAx_CLK_ENABLE ( ) __HAL_RCC_DMA

Definition at line **118** of file **stm32446e_eval_audio.h**.

Referenced by **BSP_AUDIO_OUT_MspInit()**.

#define AUDIO_SAIx_DMAx_IRQ DMA2_Stream6_IRQn

Definition at line **122** of file **stm32446e_eval_audio.h**.

Referenced by **BSP_AUDIO_OUT_MspDeInit()**, and **BSP_AUDIO_OUT_MspInit()**.

#define AUDIO_SAIx_DMAx_IRQHandler DMA2_Stream6_IRQHan

Definition at line **127** of file **stm32446e_eval_audio.h**.

#define AUDIO_SAIx_DMAx_MEM_DATA_SIZE DMA_MDATAALIG

Definition at line **124** of file **stm32446e_eval_audio.h**.

Referenced by **BSP_AUDIO_OUT_MspInit()**.

#define AUDIO_SAIx_DMAx_PERIPH_DATA_SIZE DMA_PDATAAL

Definition at line **123** of file **stm32446e_eval_audio.h**.

Referenced by **BSP_AUDIO_OUT_MspInit()**.

#define AUDIO_SAIx_DMAx_STREAM DMA2_Stream6

Definition at line **120** of file **stm32446e_eval_audio.h**.

Referenced by **BSP_AUDIO_OUT_MspInit()**.

#define AUDIO_SAIx_FS_PIN GPIO_PIN_9 /* PG9 */

Definition at line 112 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_OUT_MspDeInit(), and BSP_AUDIO_OUT_MspInit().

#define AUDIO_SAIx_MCK_PIN GPIO_PIN_1 /* PA1 */

Definition at line 107 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_OUT_MspDeInit(), and BSP_AUDIO_OUT_MspInit().

#define AUDIO_SAIx_MCLK_SCK_DISABLE ( ) __HAL_RCC_GPI

Definition at line 106 of file stm32446e_eval_audio.h.

#define AUDIO_SAIx_MCLK_SCK_ENABLE ( ) __HAL_RCC_GPIC

Definition at line 105 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_OUT_MspInit().

#define AUDIO_SAIx_MCLK_SCK_GPIO_PORT GPIOA

Definition at line 109 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_OUT_MspDeInit(), and BSP_AUDIO_OUT_MspInit().

#define AUDIO_SAIx_MCLK_SD_FS_AF GPIO_AF10_SAI2

Definition at line 102 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_OUT_MspInit().

#define AUDIO_SAIx_SCK_AF GPIO_AF8_SAI2

Definition at line 101 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_OUT_MspInit().

#define AUDIO_SAIx_SCK_PIN GPIO_PIN_2 /* PA2 */

Definition at line 108 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_OUT_MspDeInit(), and BSP_AUDIO_OUT_MspInit().

#define AUDIO_SAIx_SD_FS_DISABLE( ) __HAL_RCC_GPIOG_C

Definition at line 111 of file stm32446e_eval_audio.h.

#define AUDIO_SAIx_SD_FS_ENABLE( ) __HAL_RCC_GPIOG_C

Definition at line 110 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_OUT_MspInit().

#define AUDIO_SAIx_SD_FS_GPIO_PORT GPIOG

Definition at line 114 of file stm32446e_eval_audio.h.

Referenced by **BSP_AUDIO_OUT_MspDeInit()**, and **BSP_AUDIO_OUT_MspInit()**.

#define AUDIO_SAIx_SD_PIN GPIO_PIN_10 /* PG10 */

Definition at line **113** of file **stm32446e_eval_audio.h**.

Referenced by **BSP_AUDIO_OUT_MspDeInit()**, and **BSP_AUDIO_OUT_MspInit()**.

#define AUDIO_TIMEOUT ((uint8_t)2)

Definition at line **192** of file **stm32446e_eval_audio.h**.

#define AUDIO_TIMx TIM4

Definition at line **173** of file **stm32446e_eval_audio.h**.

Referenced by **TIMx_DeInit()**, and **TIMx_Init()**.

#define AUDIO_TIMx_AF GPIO_AF2_TIM4

Definition at line **181** of file **stm32446e_eval_audio.h**.

Referenced by **TIMx_IC_MspInit()**.

#define AUDIO_TIMx_CLK_DISABLE ( ) __HAL_RCC_TIM4_CLK_

Definition at line **172** of file **stm32446e_eval_audio.h**.

Referenced by **BSP_AUDIO_IN_Stop()**, and **TIMx_IC_MspDeInit()**.

#define AUDIO_TIMx_CLK_ENABLE ( ) __HAL_RCC_TIM4_CLK_E

Definition at line 171 of file stm32446e_eval_audio.h.

Referenced by TIMx_IC_MspInit().

#define AUDIO_TIMx_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIO

Definition at line 177 of file stm32446e_eval_audio.h.

#define AUDIO_TIMx_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOI

Definition at line 176 of file stm32446e_eval_audio.h.

Referenced by TIMx_IC_MspInit().

#define AUDIO_TIMx_GPIO_PORT GPIOB

Definition at line 178 of file stm32446e_eval_audio.h.

Referenced by TIMx_IC_MspInit().

#define AUDIO_TIMx_IN_CHANNEL TIM_CHANNEL_1

Definition at line 174 of file stm32446e_eval_audio.h.

Referenced by TIMx_DeInit(), and TIMx_Init().

#define AUDIO_TIMx_IN_GPIO_PIN GPIO_PIN_6

Definition at line 179 of file stm32446e_eval_audio.h.

Referenced by **TIMx_IC_MspInit()**.

#define AUDIO_TIMx_OUT_CHANNEL TIM_CHANNEL_2 /* Select

Definition at line **175** of file **stm32446e_eval_audio.h**.

Referenced by **TIMx_DeInit()**, and **TIMx_Init()**.

#define AUDIO_TIMx_OUT_GPIO_PIN GPIO_PIN_7

Definition at line **180** of file **stm32446e_eval_audio.h**.

Referenced by **TIMx_IC_MspInit()**.

#define AUDIODATA_SIZE 2 /* 16-bits audio data size */

Definition at line **187** of file **stm32446e_eval_audio.h**.

Referenced by **BSP_AUDIO_OUT_Play()**.

#define CHANNEL_DEMUX_MASK ((uint8_t)0x55)

Definition at line **204** of file **stm32446e_eval_audio.h**.

Referenced by **BSP_AUDIO_IN_PDMToPCM()**.

#define CODEC_RESET_DELAY ((uint8_t)5)

Definition at line **211** of file **stm32446e_eval_audio.h**.

#define DEFAULT_AUDIO_IN_BIT_RESOLUTION ((uint8_t)16)

Definition at line 196 of file stm32446e_eval_audio.h.

#define DEFAULT_AUDIO_IN_CHANNEL_NBR ((uint8_t)2) /* Mono

Definition at line 197 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_IN_PDMToPCM().

#define DEFAULT_AUDIO_IN_FREQ I2S_AUDIOFREQ_16K

Definition at line 195 of file stm32446e_eval_audio.h.

#define DEFAULT_AUDIO_IN_VOLUME ((uint16_t)64)

Definition at line 198 of file stm32446e_eval_audio.h.

#define DMA_MAX_SZE 0xFFFF

Definition at line 125 of file stm32446e_eval_audio.h.

#define INTERNAL_BUFF_SIZE (128*DEFAULT_AUDIO_IN_FREQ/

Definition at line 201 of file stm32446e_eval_audio.h.

Referenced by BSP_AUDIO_IN_PDMToPCM().

#define OUTPUT_DEVICE_HEADPHONE1 OUTPUT_DEVICE_HEA

Definition at line 219 of file stm32446e_eval_audio.h.

#define OUTPUT_DEVICE_HEADPHONE2 OUTPUT_DEVICE_SPE

Definition at line 220 of file stm32446e_eval_audio.h.

#define PCM_OUT_SIZE (DEFAULT_AUDIO_IN_FREQ/1000*2)

Definition at line 203 of file stm32446e_eval_audio.h.

STM32446E_EVAL BSP User Manual

Defines

STM32446E EVAL LOW LEVEL COM

STM32446E EVAL LOW LEVEL Exported Constants

Defines

#define	**COMn** ((uint8_t)1)
#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
#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_TX_AF** GPIO_AF7_USART1
#define	**EVAL_COM1_RX_PIN** GPIO_PIN_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_RX_AF** GPIO_AF7_USART1
#define	**EVAL_COM1_IRQn** USART1_IRQn
#define	**EVAL_COMx_CLK_ENABLE**(__INDEX__) do { if((__INDEX_ {**EVAL_COM1_CLK_ENABLE**();} } while(0)
#define	**EVAL_COMx_CLK_DISABLE**(__INDEX__) (((__INDEX__) = **EVAL_COM1_CLK_DISABLE**() : 0)
#define	**EVAL_COMx_TX_GPIO_CLK_ENABLE**(__INDEX__) do { if({**EVAL_COM1_TX_GPIO_CLK_ENABLE**();} } while(0)
#define	**EVAL_COMx_TX_GPIO_CLK_DISABLE**(__INDEX__) (((__I **EVAL_COM1_TX_GPIO_CLK_DISABLE**() : 0)
#define	**EVAL_COMx_RX_GPIO_CLK_ENABLE**(__INDEX__) do { if({**EVAL_COM1_RX_GPIO_CLK_ENABLE**();} } while(0)
#define	**EVAL_COMx_RX_GPIO_CLK_DISABLE**(__INDEX__) (((__I **EVAL_COM1_RX_GPIO_CLK_DISABLE**() : 0)
#define	**IO_I2C_ADDRESS** ((uint16_t)0x84) /*mfx MFX_I2C_ADDR 0 Joystick Pins definition.
#define	**IO_I2C_ADDRESS_2** ((uint16_t)0x86) /*mfx MFX_I2C_ADDR
#define	**TS_I2C_ADDRESS** ((uint16_t)0x84) /*mfx MFX_I2C_ADDR 0

```
#define TS_I2C_ADDRESS_2   ((uint16_t)0x86) /*mfx MFX_I2C_ADDI
#define CAMERA_I2C_ADDRESS   ((uint16_t)0x5A)
#define AUDIO_I2C_ADDRESS   ((uint16_t)0x34)
#define EEPROM_I2C_ADDRESS_A01   ((uint16_t)0xA0)
#define EEPROM_I2C_ADDRESS_A02   ((uint16_t)0xA6)
#define EVAL_I2Cx   FMPI2C1
#define EVAL_I2Cx_CLK_ENABLE()   __HAL_RCC_FMPI2C1_CLK_
#define EVAL_I2Cx_CLK_DISABLE()   __HAL_RCC_FMPI2C1_CLK_
#define EVAL_DMAx_CLK_ENABLE()   __HAL_RCC_DMA1_CLK_EI
#define EVAL_I2Cx_SCL_SDA_GPIO_CLK_ENABLE()   __HAL_RCC
#define EVAL_I2Cx_SCL_SDA_GPIO_CLK_DISABLE()   __HAL_RC
#define EVAL_I2Cx_FORCE_RESET()   __HAL_RCC_FMPI2C1_FOR
#define EVAL_I2Cx_RELEASE_RESET()   __HAL_RCC_FMPI2C1_R
#define EVAL_I2Cx_SCL_PIN   GPIO_PIN_12
#define EVAL_I2Cx_SCL_SDA_GPIO_PORT   GPIOD
#define EVAL_I2Cx_SCL_SDA_AF   GPIO_AF4_FMPI2C1
#define EVAL_I2Cx_SDA_PIN   GPIO_PIN_13
#define EVAL_I2Cx_EV_IRQn   FMPI2C1_EV_IRQn
#define EVAL_I2Cx_ER_IRQn   FMPI2C1_ER_IRQn
```

Define Documentation

#define AUDIO_I2C_ADDRESS ((uint16_t)0x34)

Definition at line 316 of file stm32446e_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 CAMERA_I2C_ADDRESS ((uint16_t)0x5A)

Definition at line 315 of file stm32446e_eval.h.

Referenced by BSP_CAMERA_Init().

#define COMn ((uint8_t)1)

Definition at line 236 of file stm32446e_eval.h.

#define EEPROM_I2C_ADDRESS_A01 ((uint16_t)0xA0)

Definition at line 317 of file stm32446e_eval.h.

Referenced by BSP_EEPROM_Init().

#define EEPROM_I2C_ADDRESS_A02 ((uint16_t)0xA6)

Definition at line 318 of file stm32446e_eval.h.

Referenced by **BSP_EEPROM_Init()**.

#define EVAL_COM1 USART1

Definition for COM port1, connected to USART1.

Definition at line **241** of file **stm32446e_eval.h**.

#define EVAL_COM1_CLK_DISABLE ( ) __HAL_RCC_USART1_CL

Definition at line **243** of file **stm32446e_eval.h**.

#define EVAL_COM1_CLK_ENABLE ( ) __HAL_RCC_USART1_CL

Definition at line **242** of file **stm32446e_eval.h**.

#define EVAL_COM1_IRQn USART1_IRQn

Definition at line **257** of file **stm32446e_eval.h**.

#define EVAL_COM1_RX_AF GPIO_AF7_USART1

Definition at line **255** of file **stm32446e_eval.h**.

#define EVAL_COM1_RX_GPIO_CLK_DISABLE ( ) __HAL_RCC_G

Definition at line **254** of file **stm32446e_eval.h**.

#define EVAL_COM1_RX_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line **253** of file **stm32446e_eval.h**.

#define EVAL_COM1_RX_GPIO_PORT GPIOA

Definition at line **252** of file **stm32446e_eval.h**.

#define EVAL_COM1_RX_PIN GPIO_PIN_10

Definition at line **251** of file **stm32446e_eval.h**.

#define EVAL_COM1_TX_AF GPIO_AF7_USART1

Definition at line **249** of file **stm32446e_eval.h**.

#define EVAL_COM1_TX_GPIO_CLK_DISABLE ( ) __HAL_RCC_G

Definition at line **248** of file **stm32446e_eval.h**.

#define EVAL_COM1_TX_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line **247** of file **stm32446e_eval.h**.

#define EVAL_COM1_TX_GPIO_PORT GPIOA

Definition at line **246** of file **stm32446e_eval.h**.

#define EVAL_COM1_TX_PIN GPIO_PIN_9

Definition at line **245** of file **stm32446e_eval.h**.

#define EVAL_COMx_CLK_DISABLE (__INDEX__) (((__INDEX__

Definition at line 260 of file stm32446e_eval.h.

Referenced by BSP_COM_DeInit().

#define EVAL_COMx_CLK_ENABLE (__INDEX__) do { if((__IND

Definition at line 259 of file stm32446e_eval.h.

Referenced by BSP_COM_Init().

#define EVAL_COMx_RX_GPIO_CLK_DISABLE (__INDEX__) (((

Definition at line 266 of file stm32446e_eval.h.

#define EVAL_COMx_RX_GPIO_CLK_ENABLE (__INDEX__) do

Definition at line 265 of file stm32446e_eval.h.

Referenced by BSP_COM_Init().

#define EVAL_COMx_TX_GPIO_CLK_DISABLE (__INDEX__) (((

Definition at line 263 of file stm32446e_eval.h.

#define EVAL_COMx_TX_GPIO_CLK_ENABLE (__INDEX__) do

Definition at line 262 of file stm32446e_eval.h.

Referenced by **BSP_COM_Init()**.

#define EVAL_DMAx_CLK_ENABLE ( ) __HAL_RCC_DMA1_CLK_

Definition at line **327** of file **stm32446e_eval.h**.

#define EVAL_I2Cx FMPI2C1

Definition at line **324** of file **stm32446e_eval.h**.

Referenced by **I2Cx_Init()**.

#define EVAL_I2Cx_CLK_DISABLE ( ) __HAL_RCC_FMPI2C1_CL

Definition at line **326** of file **stm32446e_eval.h**.

#define EVAL_I2Cx_CLK_ENABLE ( ) __HAL_RCC_FMPI2C1_CLK

Definition at line **325** of file **stm32446e_eval.h**.

Referenced by **I2Cx_MspInit()**.

#define EVAL_I2Cx_ER_IRQn FMPI2C1_ER_IRQn

Definition at line **342** of file **stm32446e_eval.h**.

Referenced by **I2Cx_MspInit()**.

#define EVAL_I2Cx_EV_IRQn FMPI2C1_EV_IRQn

Definition at line **341** of file **stm32446e_eval.h**.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_FORCE_RESET ( ) __HAL_RCC_FMPI2C1_FC

Definition at line 331 of file stm32446e_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_RELEASE_RESET ( ) __HAL_RCC_FMPI2C1_

Definition at line 332 of file stm32446e_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SCL_PIN GPIO_PIN_12

Definition at line 335 of file stm32446e_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SCL_SDA_AF GPIO_AF4_FMPI2C1

Definition at line 337 of file stm32446e_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SCL_SDA_GPIO_CLK_DISABLE ( ) __HAL_R

Definition at line 329 of file stm32446e_eval.h.

#define EVAL_I2Cx_SCL_SDA_GPIO_CLK_ENABLE ( ) __HAL_RC

Definition at line **328** of file **stm32446e_eval.h**.

Referenced by **I2Cx_MspInit()**.

#define EVAL_I2Cx_SCL_SDA_GPIO_PORT GPIOD

Definition at line **336** of file **stm32446e_eval.h**.

Referenced by **I2Cx_MspInit()**.

#define EVAL_I2Cx_SDA_PIN GPIO_PIN_13

Definition at line **338** of file **stm32446e_eval.h**.

Referenced by **I2Cx_MspInit()**.

#define IO_I2C_ADDRESS ((uint16_t)0x84) /*mfx MFX_I2C_ADDR

Joystick Pins definition.

Eval Pins definition connected to MFX

Definition at line **309** of file **stm32446e_eval.h**.

Referenced by **BSP_IO_ConfigIrqOutPin()**, **BSP_IO_ConfigPin()**, **BSP_IO_Init()**, **BSP_IO_ITClear()**, **BSP_IO_ITGetStatus()**, **BSP_IO_ReadPin()**, **BSP_IO_TogglePin()**, **BSP_IO_WritePin()**, and **BSP_TS_Init()**.

#define IO_I2C_ADDRESS_2 ((uint16_t)0x86) /*mfx MFX_I2C_ADD

Definition at line **310** of file **stm32446e_eval.h**.

#define TS_I2C_ADDRESS ((uint16_t)0x84) /*mfx MFX_I2C_ADDR

Definition at line 311 of file stm32446e_eval.h.

Referenced by BSP_TS_Init().

#define TS_I2C_ADDRESS_2 ((uint16_t)0x86) /*mfx MFX_I2C_ADD

Definition at line 312 of file stm32446e_eval.h.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL LOW LEVEL Private FunctionPrototypes

STM32446E EVAL LOW LEVEL

Functions

static void	**I2Cx_MspInit** (void) Initializes I2C MSP.
static void	**I2Cx_Init** (void) Initializes I2C HAL.
static void	**FMC_BANK1_Init** (void) Initializes LCD IO.
static void	**FMC_BANK1_MspInit** (void) Initializes FMC_BANK1 MSP.
void	**LCD_IO_Init** (void) Initializes LCD low level.
void	**AUDIO_IO_Init** (void) Initializes Audio low level.
void	**AUDIO_IO_DeInit** (void) DeInitializes Audio low level.
void	**CAMERA_IO_Init** (void) Initializes Camera low level.
void	**EEPROM_IO_Init** (void) Initializes peripherals used by the I2C EEPROM driver.

Function Documentation

void AUDIO_IO_DeInit (void)

DeInitializes Audio low level.

Definition at line **1143** of file **stm32446e_eval.c**.

void AUDIO_IO_Init (void)

Initializes Audio low level.

Definition at line **1135** of file **stm32446e_eval.c**.

References **I2Cx_Init()**.

void CAMERA_IO_Init (void)

Initializes Camera low level.

Definition at line **1200** of file **stm32446e_eval.c**.

References **I2Cx_Init()**.

void EEPROM_IO_Init (void)

Initializes peripherals used by the I2C EEPROM driver.

Definition at line **1251** of file **stm32446e_eval.c**.

References **I2Cx_Init()**.

Referenced by **BSP_EEPROM_Init()**.

static void FMC_BANK1_Init (void) [static]

Initializes LCD IO.

Definition at line **915** of file **stm32446e_eval.c**.

References **FMC_BANK1_MspInit()**.

Referenced by **LCD_IO_Init()**.

static void FMC_BANK1_MspInit (void) [static]

Initializes FMC_BANK1 MSP.

Definition at line **876** of file **stm32446e_eval.c**.

Referenced by **FMC_BANK1_Init()**.

static void I2Cx_Init (void) [static]

Initializes I2C HAL.

Definition at line **733** of file **stm32446e_eval.c**.

References **EVAL_I2Cx**, **hEvalI2c**, and **I2Cx_MspInit()**.

Referenced by **AUDIO_IO_Init()**, **CAMERA_IO_Init()**, **EEPROM_IO_Init()**, and **I2Cx_Error()**.

static void I2Cx_MspInit (void) [static]

Initializes I2C MSP.

Definition at line **691** of file **stm32446e_eval.c**.

References **EVAL_I2Cx_CLK_ENABLE**, **EVAL_I2Cx_ER_IRQn**, **EVAL_I2Cx_EV_IRQn**, **EVAL_I2Cx_FORCE_RESET**, **EVAL_I2Cx_RELEASE_RESET**, **EVAL_I2Cx_SCL_PIN**, **EVAL_I2Cx_SCL_SDA_AF**, **EVAL_I2Cx_SCL_SDA_GPIO_CLK_ENABLE**, **EVAL_I2Cx_SCL_SDA_GPIO_PORT**, and **EVAL_I2Cx_SDA_PIN**.

Referenced by **I2Cx_Init()**.

void LCD_IO_Init **(void)**

Initializes LCD low level.

Definition at line **1096** of file **stm32446e_eval.c**.

References **FMC_BANK1_Init()**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL LOW LEVEL Private Functions

STM32446E EVAL LOW LEVEL

Functions

uint32_t	**BSP_GetVersion** (void) This method returns the STM32446E EVAL BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LEDs.
void	**BSP_LED_DeInit** (**Led_TypeDef** Led) DeInit LEDs.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** ButtonMode) Configures button GPIO and EXTI Line.
void	**BSP_PB_DeInit** (**Button_TypeDef** Button) Push Button DeInit.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) Configures COM port.
void	**BSP_COM_DeInit** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) DeInit COM port.
static HAL_StatusTypeDef	**I2Cx_ReadMultiple** (uint8_t Addr, uint16_t Reg, uint16_t MemAddSize, uint8_t *Buffer, uint16_t Length)

	Reads multiple data.
static HAL_StatusTypeDef	**I2Cx_WriteMultiple** (uint8_t Addr, uint16_t Reg, uint16_t MemAddSize, uint8_t *Buffer, uint16_t Length) Writes a value in a register of the device through BUS in using DMA mode.
static HAL_StatusTypeDef	**I2Cx_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.
static void	**I2Cx_Error** (uint8_t Addr) Manages error callback by re-initializing I2C.
static void	**FMC_BANK1_WriteData** (uint16_t Data) Writes register value.
static void	**FMC_BANK1_WriteReg** (uint8_t Reg) Writes register address.
static uint16_t	**FMC_BANK1_ReadData** (void) Reads register value.
void	**LCD_IO_WriteData** (uint16_t RegValue) Writes data on LCD data register.
void	**LCD_IO_WriteReg** (uint8_t Reg) Writes register on LCD register.
uint16_t	**LCD_IO_ReadData** (void) Reads data from LCD data register.
void	**AUDIO_IO_Write** (uint8_t Addr, uint16_t Reg, uint16_t Value) Writes a single data.
uint16_t	**AUDIO_IO_Read** (uint8_t Addr, uint16_t Reg) Reads a single data.
void	**AUDIO_IO_Delay** (uint32_t Delay) AUDIO Codec delay.
	CAMERA_IO_Write (uint8_t Addr,

void	uint16_t Reg, uint16_t Value) Camera writes single data.
uint16_t	**CAMERA_IO_Read** (uint8_t Addr, uint16_t Reg) Camera reads single data.
void	**CAMERA_Delay** (uint32_t Delay) Camera delay.
HAL_StatusTypeDef	**EEPROM_IO_WriteData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Write data to I2C EEPROM driver in using DMA channel.
HAL_StatusTypeDef	**EEPROM_IO_ReadData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Read data from I2C EEPROM driver in using DMA channel.
HAL_StatusTypeDef	**EEPROM_IO_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.

Function Documentation

void AUDIO_IO_Delay (uint32_t Delay)

AUDIO Codec delay.

Parameters:

Delay,: Delay in ms

Definition at line **1190** of file **stm32446e_eval.c**.

uint16_t AUDIO_IO_Read (uint8_t Addr,
uint16_t Reg
)

Reads a single data.

Parameters:

Addr,: I2C address

Reg,: Reg address

Return values:

Data to be read

Definition at line **1171** of file **stm32446e_eval.c**.

References **I2Cx_ReadMultiple()**.

void AUDIO_IO_Write (uint8_t Addr,
uint16_t Reg,
uint16_t Value
)

Writes a single data.

Parameters:

Addr,: I2C address

Reg,: Reg address

Value,: Data to be written

Definition at line **1154** of file **stm32446e_eval.c**.

References **I2Cx_WriteMultiple()**.

void BSP_COM_DeInit (COM_TypeDef COM,
UART_HandleTypeDef * huart
)

DeInit COM port.

Parameters:

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

- COM1
- COM2

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

Definition at line **573** of file **stm32446e_eval.c**.

References **COM_USART**, and **EVAL_COMx_CLK_DISABLE**.

void BSP_COM_Init (COM_TypeDef COM,
UART_HandleTypeDef * huart
)

Configures COM port.

Parameters:

COM,: COM port to be configured. This parameter can be one of the following values:
- COM1
- COM2

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

Definition at line **534** of file **stm32446e_eval.c**.

References **COM_RX_AF**, **COM_RX_PIN**, **COM_RX_PORT**, **COM_TX_AF**, **COM_TX_PIN**, **COM_TX_PORT**, **COM_USART**, **EVAL_COMx_CLK_ENABLE**, **EVAL_COMx_RX_GPIO_CLK_ENABLE**, and **EVAL_COMx_TX_GPIO_CLK_ENABLE**.

uint32_t BSP_GetVersion (void)

This method returns the STM32446E EVAL BSP Driver revision.

Return values:

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

Definition at line **230** of file **stm32446e_eval.c**.

References **__STM32446E_EVAL_BSP_VERSION**.

void BSP_LED_DeInit (Led_TypeDef Led)

DeInit LEDs.

Parameters:

Led,: LED to be configured. This parameter can be one of the following values:
- LED1
- LED2
- LED3
- LED4

Note:
Led DeInit does not disable the GPIO clock nor disable the Mfx

Definition at line **296** of file **stm32446e_eval.c**.

References **BSP_IO_ConfigPin()**, **GPIO_PIN**, **LED1**, **LED3**, and **LEDx_GPIO_PORT**.

void BSP_LED_Init (Led_TypeDef Led)

Configures LEDs.

Parameters:

Led,: LED to be configured. This parameter can be one of the following values:
- LED1
- LED2
- LED3
- LED4

Definition at line **244** of file **stm32446e_eval.c**.

References **BSP_IO_ConfigPin()**, **BSP_IO_Init()**, **BSP_IO_PIN_SET**, **BSP_IO_WritePin()**, **GPIO_PIN**, **LED1**, **LED3**, **LEDx_GPIO_CLK_ENABLE**, and **LEDx_GPIO_PORT**.

void BSP_LED_Off (Led_TypeDef Led)

Turns selected LED Off.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Definition at line **369** of file **stm32446e_eval.c**.

References **BSP_IO_PIN_SET**, **BSP_IO_WritePin()**, **GPIO_PIN**, **LED1**, **LED3**, and **LEDx_GPIO_PORT**.

void BSP_LED_On (Led_TypeDef Led)

Turns selected LED On.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Definition at line **335** of file **stm32446e_eval.c**.

References **BSP_IO_PIN_RESET**, **BSP_IO_WritePin()**, **GPIO_PIN**, **LED1**, **LED3**, and **LEDx_GPIO_PORT**.

void BSP_LED_Toggle (Led_TypeDef Led)

Toggles the selected LED.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Definition at line **404** of file **stm32446e_eval.c**.

References **BSP_IO_TogglePin()**, **GPIO_PIN**, **LED1**, **LED3**, and **LEDx_GPIO_PORT**.

void BSP_PB_DeInit (Button_TypeDef Button)

Push Button DeInit.

Parameters:

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

- BUTTON_WAKEUP: Wakeup Push Button
- BUTTON_TAMPER: Tamper Push Button
- BUTTON_KEY: Key Push Button

Note:

On STM32446E-EVAL evaluation board, the three buttons (Wakeup, Tamper and key buttons) are mapped on the same push button nammed "Wakeup/Tamper" on the board serigraphy.
PB DeInit does not disable the GPIO clock

Definition at line **498** of file **stm32446e_eval.c**.

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

uint32_t BSP_PB_GetState (Button_TypeDef Button)

Returns the selected button state.

Parameters:

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

- BUTTON_WAKEUP: Wakeup Push Button
- BUTTON_TAMPER: Tamper Push Button
- BUTTON_KEY: Key Push Button

Note:

On STM32446E-EVAL evaluation board, the three buttons (Wakeup, Tamper and key buttons) are mapped on the same push button nammed "Wakeup/Tamper" on the board serigraphy.

Return values:

The Button GPIO pin value

Definition at line **520** of file **stm32446e_eval.c**.

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

void BSP_PB_Init (Button_TypeDef Button,
ButtonMode_TypeDef ButtonMode
)

Configures button GPIO and EXTI Line.

Parameters:

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

- BUTTON_WAKEUP: Wakeup Push Button

- BUTTON_TAMPER: Tamper Push Button
- BUTTON_KEY: Key Push Button

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

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

Note:

On STM32446E-EVAL evaluation board, the three buttons (Wakeup, Tamper and key buttons) are mapped on the same push button named "Wakeup/Tamper" on the board serigraphy.

Definition at line **445** of file **stm32446e_eval.c**.

References **BUTTON_IRQn**, **BUTTON_MODE_EXTI**, **BUTTON_MODE_GPIO**, **BUTTON_PIN**, **BUTTON_PORT**, **BUTTON_WAKEUP**, and **BUTTONx_GPIO_CLK_ENABLE**.

void CAMERA_Delay (uint32_t Delay)

Camera delay.

Parameters:

Delay,: Delay in ms

Definition at line **1241** of file **stm32446e_eval.c**.

uint16_t CAMERA_IO_Read (uint8_t Addr,
uint16_t Reg
)

Camera reads single data.

Parameters:

Addr,: I2C address

Reg,: Register address

Return values:

Read data

Definition at line **1226** of file **stm32446e_eval.c**.

References **I2Cx_ReadMultiple()**.

void CAMERA_IO_Write (uint8_t Addr,
uint16_t Reg,
uint16_t Value
)

Camera writes single data.

Parameters:

Addr,: I2C address

Reg,: Register address

Value,: Data to be written

Definition at line **1211** of file **stm32446e_eval.c**.

References **I2Cx_WriteMultiple()**.

HAL_StatusTypeDef EEPROM_IO_IsDeviceReady (uint16_t DevAd
uint32_t Trials
)

Checks if target device is ready for communication.

Note:

This function is used with Memory devices

Parameters:

DevAddress,: Target device address

Trials,: Number of trials

Return values:

HAL status

Definition at line **1289** of file **stm32446e_eval.c**.

References **I2Cx_IsDeviceReady()**.

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

HAL_StatusTypeDef EEPROM_IO_ReadData (uint16_t DevAddress, uint16_t MemAddress, uint8_t * pBuffer, uint32_t BufferSize)

Read data from I2C EEPROM driver in using DMA channel.

Parameters:

DevAddress,: Target device address

MemAddress,: Internal memory address

pBuffer,: Pointer to data buffer

BufferSize,: Amount of data to be read

Return values:

HAL status

Definition at line **1277** of file **stm32446e_eval.c**.

References I2Cx_ReadMultiple().

Referenced by BSP_EEPROM_ReadBuffer().

HAL_StatusTypeDef EEPROM_IO_WriteData (uint16_t DevAddress
uint16_t MemAddres
uint8_t * pBuffer,
uint32_t BufferSize
)

Write data to I2C EEPROM driver in using DMA channel.

Parameters:

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

Return values:

HAL status

Definition at line 1264 of file stm32446e_eval.c.

References I2Cx_WriteMultiple().

Referenced by BSP_EEPROM_WritePage().

static uint16_t FMC_BANK1_ReadData (void) [static]

Reads register value.

Return values:

Read value

Definition at line **980** of file **stm32446e_eval.c**.

References **FMC_BANK1**.

Referenced by **LCD_IO_ReadData()**.

static void FMC_BANK1_WriteData (uint16_t Data) [static]

Writes register value.

Parameters:

Data,: Data to be written

Definition at line **960** of file **stm32446e_eval.c**.

References **FMC_BANK1**.

Referenced by **LCD_IO_WriteData()**.

static void FMC_BANK1_WriteReg (uint8_t Reg) [static]

Writes register address.

Parameters:

Reg,: Register to be written

Definition at line **970** of file **stm32446e_eval.c**.

References **FMC_BANK1**.

Referenced by **LCD_IO_WriteReg()**.

static void I2Cx_Error (uint8_t Addr) [static]

Manages error callback by re-initializing I2C.

Parameters:

Addr,: I2C Address

Definition at line **863** of file **stm32446e_eval.c**.

References **hEvalI2c**, and **I2Cx_Init()**.

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

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 **854** of file **stm32446e_eval.c**.

References **hEvalI2c**.

Referenced by **EEPROM_IO_IsDeviceReady()**.

static HAL_StatusTypeDef I2Cx_ReadMultiple (uint8_t Addr,
uint16_t Reg,
uint16_t MemAddS
uint8_t * Buffer,

uint16_t **Length**
) [static]

Reads multiple data.

Parameters:

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

Return values:

Number of read data

Definition at line **808** of file **stm32446e_eval.c**.

References **hEvalI2c**, and **I2Cx_Error()**.

Referenced by **AUDIO_IO_Read()**, **CAMERA_IO_Read()**, and **EEPROM_IO_ReadData()**.

static HAL_StatusTypeDef I2Cx_WriteMultiple (**uint8_t** **Addr,**
uint16_t **Reg,**
uint16_t **MemAddS**
uint8_t * **Buffer,**
uint16_t **Length**
) [static]

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

Parameters:

Addr,:	Device address on BUS Bus.

Reg,:	The target register address to write
MemAddSize,:	address size
Buffer,:	The target register value to be written
Length,:	buffer size to be written

Return values:

HAL status

Definition at line **832** of file **stm32446e_eval.c**.

References **hEvalI2c**, and **I2Cx_Error()**.

Referenced by **AUDIO_IO_Write()**, **CAMERA_IO_Write()**, and **EEPROM_IO_WriteData()**.

uint16_t LCD_IO_ReadData (void)

Reads data from LCD data register.

Return values:

Read data.

Definition at line **1125** of file **stm32446e_eval.c**.

References **FMC_BANK1_ReadData()**.

void LCD_IO_WriteData (uint16_t RegValue)

Writes data on LCD data register.

Parameters:

RegValue,: Data to be written

Definition at line **1105** of file **stm32446e_eval.c**.

References **FMC_BANK1_WriteData()**.

void LCD_IO_WriteReg (uint8_t Reg)

Writes register on LCD register.

Parameters:

Reg,: Register to be written

Definition at line **1115** of file **stm32446e_eval.c**.

References **FMC_BANK1_WriteReg()**.

STM32446E_EVAL BSP User Manual

Variables

STM32446E EVAL AUDIO Exported Variables

STM32446E EVAL AUDIO

Variables

__IO uint16_t	**AudioInVolume**

Variable Documentation

__IO uint16_t AudioInVolume

Definition at line **174** of file **stm32446e_eval_audio.c**.

Referenced by **BSP_AUDIO_IN_PDMToPCM()**, and **BSP_AUDIO_IN_SetVolume()**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL
AUDIO IN Private Functions

STM32446E EVAL AUDIO

Functions

uint8_t	**BSP_AUDIO_IN_Init** (uint32_t AudioFreq, uint32_t BitRes, uint32_t ChnlNbr) Initializes wave recording.
uint8_t	**BSP_AUDIO_IN_Record** (uint16_t *pbuf, uint32_t size) Starts audio recording.
uint8_t	**BSP_AUDIO_IN_Stop** (void) Stops audio recording.
uint8_t	**BSP_AUDIO_IN_Pause** (void) Pauses the audio file stream.
uint8_t	**BSP_AUDIO_IN_Resume** (void) Resumes the audio file stream.
uint8_t	**BSP_AUDIO_IN_SetVolume** (uint8_t Volume) Controls the audio in volume level.
void	**BSP_AUDIO_IN_DeInit** (void) Deinit the audio IN peripherals.
uint8_t	**BSP_AUDIO_IN_PDMToPCM** (uint16_t *PDMBuf, uint16_t *PCMBuf) Converts audio format from PDM to PCM.
void	**HAL_I2S_RxCpltCallback** (I2S_HandleTypeDef *hi2s) Rx Transfer completed callbacks.
void	**HAL_I2S_RxHalfCpltCallback** (I2S_HandleTypeDef *hi2s) Rx Half Transfer completed callbacks.
void	**HAL_I2S_ErrorCallback** (I2S_HandleTypeDef *hi2s) I2S error callbacks.
__weak void	**BSP_AUDIO_IN_TransferComplete_CallBack** (void) User callback when record buffer is filled.
__weak void	**BSP_AUDIO_IN_HalfTransfer_CallBack** (void) Manages the DMA Half Transfer complete event.
__weak void	**BSP_AUDIO_IN_Error_Callback** (void) Audio IN Error callback function.

__weak void	**BSP_AUDIO_IN_MspInit** (I2S_HandleTypeDef *hi2s, void *Params) Initializes BSP_AUDIO_IN MSP.
__weak void	**BSP_AUDIO_IN_MspDeInit** (I2S_HandleTypeDef *hi2s, void *Params) DeInitializes BSP_AUDIO_IN MSP.
static void	**PDMDecoder_Init** (uint32_t AudioFreq, uint32_t ChnlNbr) Initializes the PDM library.
static void	**I2Sx_Init** (uint32_t AudioFreq) Initializes the Audio Codec audio interface (I2S)
static void	**TIMx_IC_MspInit** (TIM_HandleTypeDef *htim) Initializes the TIM INput Capture MSP.
static void	**TIMx_IC_MspDeInit** (TIM_HandleTypeDef *htim) Initializes the TIM INput Capture MSP.
static void	**TIMx_Init** (void) Configure TIM as a clock divider by 2.
static void	**TIMx_DeInit** (void) Configure TIM as a clock divider by 2.

Function Documentation

void BSP_AUDIO_IN_DeInit (void)

Deinit the audio IN peripherals.

Definition at line 894 of file stm32446e_eval_audio.c.

References BSP_AUDIO_IN_MspDeInit(), haudio_in_i2s, I2Sx_DeInit(), and TIMx_DeInit().

__weak void BSP_AUDIO_IN_Error_Callback (void)

Audio IN Error callback function.

Definition at line 988 of file stm32446e_eval_audio.c.

Referenced by HAL_I2S_ErrorCallback().

__weak void BSP_AUDIO_IN_HalfTransfer_CallBack (void)

Manages the DMA Half Transfer complete event.

Definition at line 978 of file stm32446e_eval_audio.c.

Referenced by HAL_I2S_RxHalfCpltCallback().

uint8_t BSP_AUDIO_IN_Init (uint32_t AudioFreq, uint32_t BitRes, uint32_t ChnlNbr)

Initializes wave recording.

Note:
This function assumes that the I2S input clock (through PLL_R in Devices RevA/Z and through dedicated PLLI2S_R in Devices RevB/Y) is already configured and ready to be used.

Parameters:

AudioFreq,:	Audio frequency to be configured for the I2S peripheral.
BitRes,:	Audio frequency to be configured for the I2S peripheral.
ChnlNbr,:	Audio frequency to be configured for the I2S peripheral.

Return values:

AUDIO_OK	if correct communication, else wrong communication

Definition at line **783** of file **stm32446e_eval_audio.c**.

References **AUDIO_I2Sx**, **AUDIO_OK**, **BSP_AUDIO_IN_MspInit()**, **haudio_in_i2s**, **I2Sx_DeInit()**, **I2Sx_Init()**, and **PDMDecoder_Init()**.

__weak void BSP_AUDIO_IN_MspDeInit (I2S_HandleTypeDef * hi2s
void * Par
)

DeInitializes BSP_AUDIO_IN MSP.

Parameters:

hi2s,:	I2S handle
Params,:	pointer on additional configuration parameters, can be NULL.

Definition at line **1067** of file **stm32446e_eval_audio.c**.

References **AUDIO_I2Sx**, **AUDIO_I2Sx_CLK_DISABLE**,

AUDIO_I2Sx_DMAx_IRQ, AUDIO_I2Sx_SCK_GPIO_PORT, AUDIO_I2Sx_SCK_PIN, AUDIO_I2Sx_SD_GPIO_PORT, and AUDIO_I2Sx_SD_PIN.

Referenced by BSP_AUDIO_IN_DeInit().

__weak void BSP_AUDIO_IN_MspInit (I2S_HandleTypeDef * hi2s, void * Param)

Initializes BSP_AUDIO_IN MSP.

Parameters:

hi2s,:	I2S handle
Params,:	pointer on additional configuration parameters, can be NULL.

Definition at line 999 of file stm32446e_eval_audio.c.

References AUDIO_I2Sx, AUDIO_I2Sx_CLK_ENABLE, AUDIO_I2Sx_DMAx_CHANNEL, AUDIO_I2Sx_DMAx_CLK_ENABLE, AUDIO_I2Sx_DMAx_IRQ, AUDIO_I2Sx_DMAx_MEM_DATA_SIZE, AUDIO_I2Sx_DMAx_PERIPH_DATA_SIZE, AUDIO_I2Sx_DMAx_STREAM, AUDIO_I2Sx_SCK_AF, AUDIO_I2Sx_SCK_GPIO_CLK_ENABLE, AUDIO_I2Sx_SCK_GPIO_PORT, AUDIO_I2Sx_SCK_PIN, AUDIO_I2Sx_SD_AF, AUDIO_I2Sx_SD_GPIO_CLK_ENABLE, AUDIO_I2Sx_SD_GPIO_PORT, AUDIO_I2Sx_SD_PIN, AUDIO_IN_IRQ_PREPRIO, and TIMx_Init().

Referenced by BSP_AUDIO_IN_Init().

uint8_t BSP_AUDIO_IN_Pause (void)

Pauses the audio file stream.

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **854** of file **stm32446e_eval_audio.c**.

References **AUDIO_OK**, and **haudio_in_i2s**.

uint8_t BSP_AUDIO_IN_PDMToPCM (uint16_t * PDMBuf,
uint16_t * PCMBuf
)

Converts audio format from PDM to PCM.

Parameters:

PDMBuf,: Pointer to data PDM buffer

PCMBuf,: Pointer to data PCM buffer

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **908** of file **stm32446e_eval_audio.c**.

References **AUDIO_OK**, **AudioInVolume**, **Channel_Demux**, **CHANNEL_DEMUX_MASK**, **DEFAULT_AUDIO_IN_CHANNEL_NBR**, **Filter**, and **INTERNAL_BUFF_SIZE**.

uint8_t BSP_AUDIO_IN_Record (uint16_t * pbuf,
uint32_t size
)

Starts audio recording.

Parameters:

pbuf,: Main buffer pointer for the recorded data storing

size,: Current size of the recorded buffer

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **817** of file **stm32446e_eval_audio.c**.

References **AUDIO_ERROR**, **AUDIO_OK**, and **haudio_in_i2s**.

uint8_t BSP_AUDIO_IN_Resume (void)

Resumes the audio file stream.

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **867** of file **stm32446e_eval_audio.c**.

References **AUDIO_OK**, and **haudio_in_i2s**.

uint8_t BSP_AUDIO_IN_SetVolume (uint8_t Volume)

Controls the audio in 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 **882** of file **stm32446e_eval_audio.c**.

References **AUDIO_OK**, and **AudioInVolume**.

uint8_t BSP_AUDIO_IN_Stop (void)

Stops audio recording.

Return values:

AUDIO_OK	if correct communication, else wrong communication

Definition at line **834** of file **stm32446e_eval_audio.c**.

References **AUDIO_ERROR**, **AUDIO_OK**, **AUDIO_TIMx_CLK_DISABLE**, and **haudio_in_i2s**.

__weak void BSP_AUDIO_IN_TransferComplete_CallBack (void)

User callback when record buffer is filled.

Definition at line **968** of file **stm32446e_eval_audio.c**.

Referenced by **HAL_I2S_RxCpltCallback()**.

void HAL_I2S_ErrorCallback (I2S_HandleTypeDef * hi2s)

I2S error callbacks.

Parameters:

hi2s,: I2S handle

Definition at line **958** of file **stm32446e_eval_audio.c**.

References **BSP_AUDIO_IN_Error_Callback()**.

void HAL_I2S_RxCpltCallback (I2S_HandleTypeDef * hi2s)

Rx Transfer completed callbacks.

Parameters:

hi2s,: I2S handle

Definition at line **937** of file **stm32446e_eval_audio.c**.

References **BSP_AUDIO_IN_TransferComplete_CallBack()**.

void HAL_I2S_RxHalfCpltCallback (I2S_HandleTypeDef * hi2s)

Rx Half Transfer completed callbacks.

Parameters:

hi2s,: I2S handle

Definition at line **947** of file **stm32446e_eval_audio.c**.

References **BSP_AUDIO_IN_HalfTransfer_CallBack()**.

static void I2Sx_Init (uint32_t AudioFreq) [static]

Initializes the Audio Codec audio interface (I2S)

Note:

This function assumes that the I2S input clock (through PLL_R in Devices RevA/Z and through dedicated PLLI2S_R in Devices

RevB/Y) is already configured and ready to be used.

Parameters:

AudioFreq,: Audio frequency to be configured for the I2S peripheral.

Definition at line **1134** of file **stm32446e_eval_audio.c**.

References **AUDIO_I2Sx**, and **haudio_in_i2s**.

Referenced by **BSP_AUDIO_IN_Init()**.

static void PDMDecoder_Init (uint32_t AudioFreq, uint32_t ChnlNbr) [static]

Initializes the PDM library.

Parameters:

AudioFreq,: Audio sampling frequency

ChnlNbr,: Number of audio channels (1: mono; 2: stereo)

Definition at line **1108** of file **stm32446e_eval_audio.c**.

References **Filter**.

Referenced by **BSP_AUDIO_IN_Init()**.

static void TIMx_DeInit (void) [static]

Configure TIM as a clock divider by 2.

I2S_SCK is externally connected to TIMx input channel

Definition at line **1296** of file **stm32446e_eval_audio.c**.

References **AUDIO_TIMx**, **AUDIO_TIMx_IN_CHANNEL**, **AUDIO_TIMx_OUT_CHANNEL**, **haudio_tim**, and **TIMx_IC_MspDeInit()**.

Referenced by **BSP_AUDIO_IN_DeInit()**.

static void TIMx_IC_MspDeInit (TIM_HandleTypeDef * htim) [stati

Initializes the TIM INput Capture MSP.

Parameters:

htim,: TIM handle

Definition at line **1211** of file **stm32446e_eval_audio.c**.

References **AUDIO_TIMx_CLK_DISABLE**.

Referenced by **TIMx_DeInit()**.

static void TIMx_IC_MspInit (TIM_HandleTypeDef * htim) [static]

Initializes the TIM INput Capture MSP.

Parameters:

htim,: TIM handle

Definition at line **1179** of file **stm32446e_eval_audio.c**.

References **AUDIO_TIMx_AF**, **AUDIO_TIMx_CLK_ENABLE**, **AUDIO_TIMx_GPIO_CLK_ENABLE**, **AUDIO_TIMx_GPIO_PORT**, **AUDIO_TIMx_IN_GPIO_PIN**, and **AUDIO_TIMx_OUT_GPIO_PIN**.

Referenced by **TIMx_Init()**.

static void TIMx_Init (void) [static]

Configure TIM as a clock divider by 2.

I2S_SCK is externally connected to TIMx input channel

Definition at line **1224** of file **stm32446e_eval_audio.c**.

References **AUDIO_TIMx**, **AUDIO_TIMx_IN_CHANNEL**, **AUDIO_TIMx_OUT_CHANNEL**, **haudio_tim**, and **TIMx_IC_MspInit()**.

Referenced by **BSP_AUDIO_IN_MspInit()**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL
AUDIO IN Exported Functions

STM32446E EVAL AUDIO

Functions

uint8_t	**BSP_AUDIO_IN_Init** (uint32_t AudioFreq, uint32_t BitRes, uint32_t ChnlNbr) Initializes wave recording.
uint8_t	**BSP_AUDIO_IN_Record** (uint16_t *pData, uint32_t Size) Starts audio recording.
uint8_t	**BSP_AUDIO_IN_Stop** (void) Stops audio recording.
uint8_t	**BSP_AUDIO_IN_Pause** (void) Pauses the audio file stream.
uint8_t	**BSP_AUDIO_IN_Resume** (void) Resumes the audio file stream.
uint8_t	**BSP_AUDIO_IN_SetVolume** (uint8_t Volume) Controls the audio in volume level.
void	**BSP_AUDIO_IN_DeInit** (void) Deinit the audio IN peripherals.
uint8_t	**BSP_AUDIO_IN_PDMToPCM** (uint16_t *PDMBuf, uint16_t *PCMBuf) Converts audio format from PDM to PCM.
void	**BSP_AUDIO_IN_TransferComplete_CallBack** (void) User callback when record buffer is filled.
void	**BSP_AUDIO_IN_HalfTransfer_CallBack** (void) Manages the DMA Half Transfer complete event.
void	**BSP_AUDIO_IN_Error_Callback** (void) Audio IN Error callback function.
void	**BSP_AUDIO_IN_MspInit** (I2S_HandleTypeDef *hi2s, void *Params) Initializes BSP_AUDIO_IN MSP.
void	**BSP_AUDIO_IN_MspDeInit** (I2S_HandleTypeDef *hi2s, void *Params) DeInitializes BSP_AUDIO_IN MSP.

Function Documentation

void BSP_AUDIO_IN_DeInit (void)

Deinit the audio IN peripherals.

Definition at line 894 of file stm32446e_eval_audio.c.

References BSP_AUDIO_IN_MspDeInit(), haudio_in_i2s, I2Sx_DeInit(), and TIMx_DeInit().

void BSP_AUDIO_IN_Error_Callback (void)

Audio IN Error callback function.

Definition at line 988 of file stm32446e_eval_audio.c.

Referenced by HAL_I2S_ErrorCallback().

void BSP_AUDIO_IN_HalfTransfer_CallBack (void)

Manages the DMA Half Transfer complete event.

Definition at line 978 of file stm32446e_eval_audio.c.

Referenced by HAL_I2S_RxHalfCpltCallback().

uint8_t BSP_AUDIO_IN_Init (uint32_t AudioFreq, uint32_t BitRes, uint32_t ChnlNbr)

Initializes wave recording.

Note:
This function assumes that the I2S input clock (through PLL_R in Devices RevA/Z and through dedicated PLLI2S_R in Devices RevB/Y) is already configured and ready to be used.

Parameters:

AudioFreq,:	Audio frequency to be configured for the I2S peripheral.
BitRes,:	Audio frequency to be configured for the I2S peripheral.
ChnlNbr,:	Audio frequency to be configured for the I2S peripheral.

Return values:

AUDIO_OK	if correct communication, else wrong communication

Definition at line **783** of file **stm32446e_eval_audio.c**.

References **AUDIO_I2Sx**, **AUDIO_OK**, **BSP_AUDIO_IN_MspInit()**, **haudio_in_i2s**, **I2Sx_DeInit()**, **I2Sx_Init()**, and **PDMDecoder_Init()**.

void BSP_AUDIO_IN_MspDeInit (I2S_HandleTypeDef * hi2s, void * Params)

DeInitializes BSP_AUDIO_IN MSP.

Parameters:

hi2s,:	I2S handle
Params,:	pointer on additional configuration parameters, can be NULL.

Definition at line **1067** of file **stm32446e_eval_audio.c**.

References **AUDIO_I2Sx**, **AUDIO_I2Sx_CLK_DISABLE**,

AUDIO_I2Sx_DMAx_IRQ, AUDIO_I2Sx_SCK_GPIO_PORT, AUDIO_I2Sx_SCK_PIN, AUDIO_I2Sx_SD_GPIO_PORT, and AUDIO_I2Sx_SD_PIN.

Referenced by BSP_AUDIO_IN_DeInit().

void BSP_AUDIO_IN_MspInit (I2S_HandleTypeDef * hi2s, void * Params)

Initializes BSP_AUDIO_IN MSP.

Parameters:

hi2s,: I2S handle

Params,: pointer on additional configuration parameters, can be NULL.

Definition at line 999 of file stm32446e_eval_audio.c.

References AUDIO_I2Sx, AUDIO_I2Sx_CLK_ENABLE, AUDIO_I2Sx_DMAx_CHANNEL, AUDIO_I2Sx_DMAx_CLK_ENABLE, AUDIO_I2Sx_DMAx_IRQ, AUDIO_I2Sx_DMAx_MEM_DATA_SIZE, AUDIO_I2Sx_DMAx_PERIPH_DATA_SIZE, AUDIO_I2Sx_DMAx_STREAM, AUDIO_I2Sx_SCK_AF, AUDIO_I2Sx_SCK_GPIO_CLK_ENABLE, AUDIO_I2Sx_SCK_GPIO_PORT, AUDIO_I2Sx_SCK_PIN, AUDIO_I2Sx_SD_AF, AUDIO_I2Sx_SD_GPIO_CLK_ENABLE, AUDIO_I2Sx_SD_GPIO_PORT, AUDIO_I2Sx_SD_PIN, AUDIO_IN_IRQ_PREPRIO, and TIMx_Init().

Referenced by BSP_AUDIO_IN_Init().

uint8_t BSP_AUDIO_IN_Pause (void)

Pauses the audio file stream.

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **854** of file **stm32446e_eval_audio.c**.

References **AUDIO_OK**, and **haudio_in_i2s**.

uint8_t BSP_AUDIO_IN_PDMToPCM (uint16_t * PDMBuf,
uint16_t * PCMBuf
)

Converts audio format from PDM to PCM.

Parameters:

PDMBuf,: Pointer to data PDM buffer

PCMBuf,: Pointer to data PCM buffer

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **908** of file **stm32446e_eval_audio.c**.

References **AUDIO_OK**, **AudioInVolume**, **Channel_Demux**, **CHANNEL_DEMUX_MASK**, **DEFAULT_AUDIO_IN_CHANNEL_NBR**, **Filter**, and **INTERNAL_BUFF_SIZE**.

uint8_t BSP_AUDIO_IN_Record (uint16_t * pbuf,
uint32_t size
)

Starts audio recording.

Parameters:

pbuf,: Main buffer pointer for the recorded data storing

size,: Current size of the recorded buffer

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **817** of file **stm32446e_eval_audio.c**.

References **AUDIO_ERROR**, **AUDIO_OK**, and **haudio_in_i2s**.

uint8_t BSP_AUDIO_IN_Resume (void)

Resumes the audio file stream.

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **867** of file **stm32446e_eval_audio.c**.

References **AUDIO_OK**, and **haudio_in_i2s**.

uint8_t BSP_AUDIO_IN_SetVolume (uint8_t Volume)

Controls the audio in 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 **882** of file **stm32446e_eval_audio.c**.

References **AUDIO_OK**, and **AudioInVolume**.

uint8_t BSP_AUDIO_IN_Stop (void)

Stops audio recording.

Return values:

AUDIO_OK	if correct communication, else wrong communication

Definition at line **834** of file **stm32446e_eval_audio.c**.

References **AUDIO_ERROR**, **AUDIO_OK**, **AUDIO_TIMx_CLK_DISABLE**, and **haudio_in_i2s**.

void BSP_AUDIO_IN_TransferComplete_CallBack (void)

User callback when record buffer is filled.

Definition at line **968** of file **stm32446e_eval_audio.c**.

Referenced by **HAL_I2S_RxCpltCallback()**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL
AUDIO OUT Private Functions

STM32446E EVAL AUDIO

Functions

uint8_t	**BSP_AUDIO_OUT_Init** (uint16_t OutputDevice, uint8_t Volume, uint32_t AudioFreq) Configures 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 SAI 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) Updates the audio frequency.
void	**BSP_AUDIO_OUT_SetAudioFrameSlot** (uint32_t AudioFrameSlot) Updates the Audio frame slot configuration.
void	**BSP_AUDIO_OUT_DeInit** (void) Deinit the audio peripherals.

void	**HAL_SAI_TxCpltCallback** (SAI_HandleTypeDef *hsai) Tx Transfer completed callbacks.
void	**HAL_SAI_TxHalfCpltCallback** (SAI_HandleTypeDef *hsai) Tx Half Transfer completed callbacks.
void	**HAL_SAI_ErrorCallback** (SAI_HandleTypeDef *hsai) SAI 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.
__weak void	**BSP_AUDIO_OUT_MspInit** (SAI_HandleTypeDef *hsai, void *Params) Initializes BSP_AUDIO_OUT MSP.
__weak void	**BSP_AUDIO_OUT_MspDeInit** (SAI_HandleTypeDef *hsai, void *Params) Deinitializes SAI MSP.
__weak void	**BSP_AUDIO_OUT_ClockConfig** (SAI_HandleTypeDef *hsai, uint32_t AudioFreq, void *Params) Clock Config.
static void	**SAIx_Init** (uint32_t AudioFreq) Initializes the Audio Codec audio interface (SAI).

Function Documentation

void BSP_AUDIO_OUT_ChangeBuffer (uint16_t * pData,
uint16_t Size
)

Sends n-Bytes on the SAI interface.

Parameters:

pData,: pointer on data address

Size,: number of data to be written

Definition at line **281** of file **stm32446e_eval_audio.c**.

References **haudio_out_sai**.

__weak void BSP_AUDIO_OUT_ClockConfig (SAI_HandleTypeDef
uint32_t
void *
)

Clock Config.

Parameters:

hsai,: might be required to set audio peripheral predivider if any.

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

Params,: pointer on additional configuration parameters, can be NULL.

Note:
This API is called by **BSP_AUDIO_OUT_Init()** and **BSP_AUDIO_OUT_SetFrequency()** Being __weak it can be overwritten by the application

Definition at line 647 of file stm32446e_eval_audio.c.

Referenced by BSP_AUDIO_OUT_Init(), and BSP_AUDIO_OUT_SetFrequency().

void BSP_AUDIO_OUT_DeInit (void)

Deinit the audio peripherals.

Definition at line 472 of file stm32446e_eval_audio.c.

References BSP_AUDIO_OUT_MspDeInit(), haudio_out_sai, and SAIx_DeInit().

__weak void BSP_AUDIO_OUT_Error_CallBack (void)

Manages the DMA FIFO error event.

Definition at line 527 of file stm32446e_eval_audio.c.

Referenced by HAL_SAI_ErrorCallback().

__weak void BSP_AUDIO_OUT_HalfTransfer_CallBack (void)

Manages the DMA Half Transfer complete event.

Definition at line 520 of file stm32446e_eval_audio.c.

Referenced by HAL_SAI_TxHalfCpltCallback().

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

Configures the audio peripherals.

Parameters:

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

Note:
The I2S PLL input clock must be done in the user application.

Return values:

AUDIO_OK	if correct communication, else wrong communication

Definition at line **209** of file **stm32446e_eval_audio.c**.

References **audio_drv**, **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, **AUDIO_SAIx**, **BSP_AUDIO_OUT_ClockConfig()**, **BSP_AUDIO_OUT_MspInit()**, **haudio_out_sai**, **SAIx_DeInit()**, and **SAIx_Init()**.

__weak void BSP_AUDIO_OUT_MspDeInit (SAI_HandleTypeDef * hsai,
void * Params
)

Deinitializes SAI MSP.

Parameters:

hsai,:	SAI handle
Params,:	pointer on additional configuration parameters, can

be NULL.

Definition at line **606** of file **stm32446e_eval_audio.c**.

References **AUDIO_SAIx**, **AUDIO_SAIx_CLK_DISABLE**, **AUDIO_SAIx_DMAx_IRQ**, **AUDIO_SAIx_FS_PIN**, **AUDIO_SAIx_MCK_PIN**, **AUDIO_SAIx_MCLK_SCK_GPIO_PORT**, **AUDIO_SAIx_SCK_PIN**, **AUDIO_SAIx_SD_FS_GPIO_PORT**, and **AUDIO_SAIx_SD_PIN**.

Referenced by **BSP_AUDIO_OUT_DeInit()**.

__weak void BSP_AUDIO_OUT_MspInit (SAI_HandleTypeDef * hsa
void * Par
)

Initializes BSP_AUDIO_OUT MSP.

Parameters:

hsai,: SAI handle

Params,: pointer on additional configuration parameters, can be NULL.

Definition at line **536** of file **stm32446e_eval_audio.c**.

References **AUDIO_OUT_IRQ_PREPRIO**, **AUDIO_SAIx**, **AUDIO_SAIx_CLK_ENABLE**, **AUDIO_SAIx_DMAx_CHANNEL**, **AUDIO_SAIx_DMAx_CLK_ENABLE**, **AUDIO_SAIx_DMAx_IRQ**, **AUDIO_SAIx_DMAx_MEM_DATA_SIZE**, **AUDIO_SAIx_DMAx_PERIPH_DATA_SIZE**, **AUDIO_SAIx_DMAx_STREAM**, **AUDIO_SAIx_FS_PIN**, **AUDIO_SAIx_MCK_PIN**, **AUDIO_SAIx_MCLK_SCK_ENABLE**, **AUDIO_SAIx_MCLK_SCK_GPIO_PORT**, **AUDIO_SAIx_MCLK_SD_FS_AF**, **AUDIO_SAIx_SCK_AF**, **AUDIO_SAIx_SCK_PIN**, **AUDIO_SAIx_SD_FS_ENABLE**, **AUDIO_SAIx_SD_FS_GPIO_PORT**, and **AUDIO_SAIx_SD_PIN**.

Referenced by **BSP_AUDIO_OUT_Init()**.

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. WARNING: 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 **294** of file **stm32446e_eval_audio.c**.

References **audio_drv**, **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, and **haudio_out_sai**.

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 **260** of file **stm32446e_eval_audio.c**.

References **audio_drv**, **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, **AUDIODATA_SIZE**, **DMA_MAX**, and **haudio_out_sai**.

uint8_t BSP_AUDIO_OUT_Resume (void)

This function Resumes the audio file stream.

WARNING: 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 **318** of file **stm32446e_eval_audio.c**.

References **audio_drv**, **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, and **haudio_out_sai**.

void BSP_AUDIO_OUT_SetAudioFrameSlot (uint32_t AudioFrame

Updates the Audio frame slot configuration.

Parameters:

AudioFrameSlot,:	specifies the audio Frame slot This parameter can be any value of **STM32446E EVAL Audio Slot TDM mode**

Note:

This API should be called after the **BSP_AUDIO_OUT_Init()** to adjust the audio frame slot.

Definition at line **456** of file **stm32446e_eval_audio.c**.

References **haudio_out_sai**.

void BSP_AUDIO_OUT_SetFrequency (uint32_t AudioFreq)

Updates the audio frequency.

Parameters:

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

Note:

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

Definition at line **433** of file **stm32446e_eval_audio.c**.

References **BSP_AUDIO_OUT_ClockConfig()**, and **haudio_out_sai**.

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 **392** of file **stm32446e_eval_audio.c**.

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

uint8_t BSP_AUDIO_OUT_SetOutputMode (uint8_t Output)

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

Parameters:

Output,: The audio output target:
OUTPUT_DEVICE_SPEAKER,
OUTPUT_DEVICE_HEADPHONE or
OUTPUT_DEVICE_BOTH

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **413** of file **stm32446e_eval_audio.c**.

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

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 **372** of file **stm32446e_eval_audio.c**.

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

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_SW: for software power off (by writing registers). Then no need to reconfigure the Codec after power on.
- 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 **344** of file **stm32446e_eval_audio.c**.

References **audio_drv**, **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, and **haudio_out_sai**.

__weak void BSP_AUDIO_OUT_TransferComplete_CallBack (void

Manages the DMA full Transfer complete event.

Definition at line **513** of file **stm32446e_eval_audio.c**.

Referenced by **HAL_SAI_TxCpltCallback()**.

void HAL_SAI_ErrorCallback (SAI_HandleTypeDef * hsai)

SAI error callbacks.

Parameters:

hsai,: SAI handle

Definition at line **505** of file **stm32446e_eval_audio.c**.

References **BSP_AUDIO_OUT_Error_CallBack()**.

void HAL_SAI_TxCpltCallback (SAI_HandleTypeDef * hsai)

Tx Transfer completed callbacks.

Parameters:

hsai,: SAI handle

Definition at line **483** of file **stm32446e_eval_audio.c**.

References **BSP_AUDIO_OUT_TransferComplete_CallBack()**.

void HAL_SAI_TxHalfCpltCallback (SAI_HandleTypeDef * hsai)

Tx Half Transfer completed callbacks.

Parameters:

hsai,: SAI handle

Definition at line **494** of file **stm32446e_eval_audio.c**.

References **BSP_AUDIO_OUT_HalfTransfer_CallBack()**.

static void SAIx_Init (uint32_t AudioFreq) [static]

Initializes the Audio Codec audio interface (SAI).

Parameters:

AudioFreq,: Audio frequency to be configured for the SAI peripheral.

Note:

The default SlotActive configuration is set to CODEC_AUDIOFRAME_SLOT_0123 and user can update this configuration using

Definition at line **697** of file **stm32446e_eval_audio.c**.

References **AUDIO_SAIx**, **CODEC_AUDIOFRAME_SLOT_0123**, and **haudio_out_sai**.

Referenced by **BSP_AUDIO_OUT_Init()**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL
AUDIO OUT Exported Functions

STM32446E EVAL AUDIO

Functions

uint8_t	**BSP_AUDIO_OUT_Init** (uint16_t OutputDevice, uint8_t Volume, uint32_t AudioFreq) Configures 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 SAI 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) Updates the audio frequency.
void	**BSP_AUDIO_OUT_SetAudioFrameSlot** (uint32_t AudioFrameSlot) Updates the Audio frame slot configuration.
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_DeInit** (void) Deinit the audio peripherals.
void	**BSP_AUDIO_OUT_TransferComplete_CallBack** (void) Manages the DMA full Transfer complete event.

void	**BSP_AUDIO_OUT_HalfTransfer_CallBack** (void) Manages the DMA Half Transfer complete event.
void	**BSP_AUDIO_OUT_Error_CallBack** (void) Manages the DMA FIFO error event.
void	**BSP_AUDIO_OUT_ClockConfig** (SAI_HandleTypeDef *hsai, uint32_t AudioFreq, void *Params) Clock Config.
void	**BSP_AUDIO_OUT_MspInit** (SAI_HandleTypeDef *hsai, void *Params) Initializes BSP_AUDIO_OUT MSP.
void	**BSP_AUDIO_OUT_MspDeInit** (SAI_HandleTypeDef *hsai, void *Params) Deinitializes SAI MSP.

Function Documentation

void BSP_AUDIO_OUT_ChangeBuffer (uint16_t * pData,
uint16_t Size
)

Sends n-Bytes on the SAI interface.

Parameters:

pData,: pointer on data address

Size,: number of data to be written

Definition at line **281** of file **stm32446e_eval_audio.c**.

References **haudio_out_sai**.

void BSP_AUDIO_OUT_ClockConfig (SAI_HandleTypeDef * hsai,
uint32_t AudioI
void * Param
)

Clock Config.

Parameters:

hsai,: might be required to set audio peripheral predivider if any.

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

Params,: pointer on additional configuration parameters, can be NULL.

Note:
This API is called by **BSP_AUDIO_OUT_Init()** and **BSP_AUDIO_OUT_SetFrequency()** Being __weak it can be overwritten by the application

Definition at line 647 of file stm32446e_eval_audio.c.

Referenced by BSP_AUDIO_OUT_Init(), and BSP_AUDIO_OUT_SetFrequency().

void BSP_AUDIO_OUT_DeInit (void)

Deinit the audio peripherals.

Definition at line 472 of file stm32446e_eval_audio.c.

References BSP_AUDIO_OUT_MspDeInit(), haudio_out_sai, and SAIx_DeInit().

void BSP_AUDIO_OUT_Error_CallBack (void)

Manages the DMA FIFO error event.

Definition at line 527 of file stm32446e_eval_audio.c.

Referenced by HAL_SAI_ErrorCallback().

void BSP_AUDIO_OUT_HalfTransfer_CallBack (void)

Manages the DMA Half Transfer complete event.

Definition at line 520 of file stm32446e_eval_audio.c.

Referenced by HAL_SAI_TxHalfCpltCallback().

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

Configures the audio peripherals.

Parameters:

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

Note:
The I2S PLL input clock must be done in the user application.

Return values:

AUDIO_OK	if correct communication, else wrong communication

Definition at line **209** of file **stm32446e_eval_audio.c**.

References **audio_drv**, **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, **AUDIO_SAIx**, **BSP_AUDIO_OUT_ClockConfig()**, **BSP_AUDIO_OUT_MspInit()**, **haudio_out_sai**, **SAIx_DeInit()**, and **SAIx_Init()**.

void BSP_AUDIO_OUT_MspDeInit (SAI_HandleTypeDef * hsai,
void * Params
)

Deinitializes SAI MSP.

Parameters:

hsai,:	SAI handle
Params,:	pointer on additional configuration parameters, can

be NULL.

Definition at line **606** of file **stm32446e_eval_audio.c**.

References **AUDIO_SAIx**, **AUDIO_SAIx_CLK_DISABLE**, **AUDIO_SAIx_DMAx_IRQ**, **AUDIO_SAIx_FS_PIN**, **AUDIO_SAIx_MCK_PIN**, **AUDIO_SAIx_MCLK_SCK_GPIO_PORT**, **AUDIO_SAIx_SCK_PIN**, **AUDIO_SAIx_SD_FS_GPIO_PORT**, and **AUDIO_SAIx_SD_PIN**.

Referenced by **BSP_AUDIO_OUT_DeInit()**.

void BSP_AUDIO_OUT_MspInit (SAI_HandleTypeDef * hsai,
void * Params
)

Initializes BSP_AUDIO_OUT MSP.

Parameters:

hsai,: SAI handle

Params,: pointer on additional configuration parameters, can be NULL.

Definition at line **536** of file **stm32446e_eval_audio.c**.

References **AUDIO_OUT_IRQ_PREPRIO**, **AUDIO_SAIx**, **AUDIO_SAIx_CLK_ENABLE**, **AUDIO_SAIx_DMAx_CHANNEL**, **AUDIO_SAIx_DMAx_CLK_ENABLE**, **AUDIO_SAIx_DMAx_IRQ**, **AUDIO_SAIx_DMAx_MEM_DATA_SIZE**, **AUDIO_SAIx_DMAx_PERIPH_DATA_SIZE**, **AUDIO_SAIx_DMAx_STREAM**, **AUDIO_SAIx_FS_PIN**, **AUDIO_SAIx_MCK_PIN**, **AUDIO_SAIx_MCLK_SCK_ENABLE**, **AUDIO_SAIx_MCLK_SCK_GPIO_PORT**, **AUDIO_SAIx_MCLK_SD_FS_AF**, **AUDIO_SAIx_SCK_AF**, **AUDIO_SAIx_SCK_PIN**, **AUDIO_SAIx_SD_FS_ENABLE**, **AUDIO_SAIx_SD_FS_GPIO_PORT**, and **AUDIO_SAIx_SD_PIN**.

Referenced by **BSP_AUDIO_OUT_Init()**.

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. WARNING: 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 **294** of file **stm32446e_eval_audio.c**.

References **audio_drv**, **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, and **haudio_out_sai**.

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 **260** of file **stm32446e_eval_audio.c**.

References **audio_drv**, **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, **AUDIODATA_SIZE**, **DMA_MAX**, and **haudio_out_sai**.

uint8_t BSP_AUDIO_OUT_Resume (void)

This function Resumes the audio file stream.

WARNING: 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 **318** of file **stm32446e_eval_audio.c**.

References **audio_drv**, **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, and **haudio_out_sai**.

void BSP_AUDIO_OUT_SetAudioFrameSlot (uint32_t AudioFrame

Updates the Audio frame slot configuration.

Parameters:

AudioFrameSlot,:	specifies the audio Frame slot This parameter can be any value of **STM32446E EVAL Audio Slot TDM mode**

Note:

This API should be called after the **BSP_AUDIO_OUT_Init()** to adjust the audio frame slot.

Definition at line **456** of file **stm32446e_eval_audio.c**.

References **haudio_out_sai**.

void BSP_AUDIO_OUT_SetFrequency (uint32_t AudioFreq)

Updates the audio frequency.

Parameters:

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

Note:

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

Definition at line **433** of file **stm32446e_eval_audio.c**.

References **BSP_AUDIO_OUT_ClockConfig()**, and **haudio_out_sai**.

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 **392** of file **stm32446e_eval_audio.c**.

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

uint8_t BSP_AUDIO_OUT_SetOutputMode (uint8_t Output)

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

Parameters:

Output,: The audio output target:
OUTPUT_DEVICE_SPEAKER,
OUTPUT_DEVICE_HEADPHONE or
OUTPUT_DEVICE_BOTH

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **413** of file **stm32446e_eval_audio.c**.

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

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 **372** of file **stm32446e_eval_audio.c**.

References audio_drv, AUDIO_ERROR, AUDIO_I2C_ADDRESS, and AUDIO_OK.

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_SW: for software power off (by writing registers). Then no need to reconfigure the Codec after power on.
- 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 344 of file stm32446e_eval_audio.c.

References audio_drv, AUDIO_ERROR, AUDIO_I2C_ADDRESS, AUDIO_OK, and haudio_out_sai.

void BSP_AUDIO_OUT_TransferComplete_CallBack (void)

Manages the DMA full Transfer complete event.

Definition at line 513 of file stm32446e_eval_audio.c.

Referenced by HAL_SAI_TxCpltCallback().

User Manual by  1.7.6.1

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL CAMERA Exported Functions

STM32446E-EVAL CAMERA

Functions

uint8_t	**BSP_CAMERA_Init** (uint32_t Resolution) Initializes the camera.
uint8_t	**BSP_CAMERA_DeInit** (void) DeInitializes the camera.
void	**BSP_CAMERA_ContinuousStart** (uint8_t *buff) Starts the camera capture in continuous mode.
void	**BSP_CAMERA_SnapshotStart** (uint8_t *buff) Starts the camera capture in snapshot mode.
void	**BSP_CAMERA_Suspend** (void) Suspend the CAMERA capture.
void	**BSP_CAMERA_Resume** (void) Resume the CAMERA capture.
uint8_t	**BSP_CAMERA_Stop** (void) Stop the CAMERA capture.
void	**BSP_CAMERA_HwReset** (void) CAMERA hardware reset.
void	**BSP_CAMERA_PwrDown** (void) CAMERA power down.
void	**BSP_CAMERA_LineEventCallback** (void) Line Event callback.
void	**BSP_CAMERA_VsyncEventCallback** (void) VSYNC Event callback.
void	**BSP_CAMERA_FrameEventCallback** (void) Frame Event callback.
void	**BSP_CAMERA_ErrorCallback** (void) Error callback.
void	**BSP_CAMERA_ContrastBrightnessConfig** (uint32_t contrast_level, uint32_t brightness_level) Configures the camera contrast and brightness.
void	**BSP_CAMERA_BlackWhiteConfig** (uint32_t Mode)

	Configures the camera white balance.
void	**BSP_CAMERA_ColorEffectConfig** (uint32_t Effect) Configures the camera color effect.
void	**BSP_CAMERA_IRQHandler** (void) Handles DCMI interrupt request.
void	**BSP_CAMERA_DMA_IRQHandler** (void) Handles DMA interrupt request.
void	**BSP_CAMERA_MspInit** (DCMI_HandleTypeDef *hdcmi, void *Params) Initializes the DCMI MSP.
void	**BSP_CAMERA_MspDeInit** (DCMI_HandleTypeDef *hdcmi, void *Params) DeInitializes the DCMI MSP.

Function Documentation

void BSP_CAMERA_BlackWhiteConfig (uint32_t Mode)

Configures the camera white balance.

Parameters:

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

- CAMERA_BLACK_WHITE_BW
- CAMERA_BLACK_WHITE_NEGATIVE
- CAMERA_BLACK_WHITE_BW_NEGATIVE
- CAMERA_BLACK_WHITE_NORMAL

Definition at line **359** of file **stm32446e_eval_camera.c**.

References **camera_drv**, and **cameraHwAddress**.

void BSP_CAMERA_ColorEffectConfig (uint32_t Effect)

Configures the camera color effect.

Parameters:

Effect,: Color effect This parameter can be one of the following values:

- CAMERA_COLOR_EFFECT_ANTIQUE
- CAMERA_COLOR_EFFECT_BLUE
- CAMERA_COLOR_EFFECT_GREEN
- CAMERA_COLOR_EFFECT_RED

Definition at line **376** of file **stm32446e_eval_camera.c**.

References **camera_drv**, and **cameraHwAddress**.

void BSP_CAMERA_ContinuousStart (uint8_t * buff)

Starts the camera capture in continuous mode.

Parameters:

buff,: pointer to the camera output buffer

Definition at line **218** of file **stm32446e_eval_camera.c**.

References **currentResolution**, **GetSize()**, and **hDcmiEval**.

void BSP_CAMERA_ContrastBrightnessConfig (uint32_t contrast_
uint32_t brightne
)

Configures the camera contrast and brightness.

Parameters:

contrast_level,: Contrast level This parameter can be one of the following values:

- CAMERA_CONTRAST_LEVEL4: for contrast +2
- CAMERA_CONTRAST_LEVEL3: for contrast +1
- CAMERA_CONTRAST_LEVEL2: for contrast 0
- CAMERA_CONTRAST_LEVEL1: for contrast -1
- CAMERA_CONTRAST_LEVEL0: for contrast -2

brightness_level,: Contrast level This parameter can be one of the following values:

- CAMERA_BRIGHTNESS_LEVEL4: for brightness +2
- CAMERA_BRIGHTNESS_LEVEL3:

for brightness +1
- CAMERA_BRIGHTNESS_LEVEL2: for brightness 0
- CAMERA_BRIGHTNESS_LEVEL1: for brightness -1
- CAMERA_BRIGHTNESS_LEVEL0: for brightness -2

Definition at line **342** of file **stm32446e_eval_camera.c**.

References **camera_drv**, and **cameraHwAddress**.

uint8_t BSP_CAMERA_DeInit (void)

DeInitializes the camera.

Return values:

Camera status

Definition at line **205** of file **stm32446e_eval_camera.c**.

References **BSP_CAMERA_MspDeInit()**, **CAMERA_OK**, and **hDcmiEval**.

void BSP_CAMERA_DMA_IRQHandler (void)

Handles DMA interrupt request.

Definition at line **395** of file **stm32446e_eval_camera.c**.

References **hDcmiEval**.

void BSP_CAMERA_ErrorCallback (void)

Error callback.

Definition at line **632** of file **stm32446e_eval_camera.c**.

Referenced by **HAL_DCMI_ErrorCallback()**.

void BSP_CAMERA_FrameEventCallback (void)

Frame Event callback.

Definition at line **613** of file **stm32446e_eval_camera.c**.

Referenced by **HAL_DCMI_FrameEventCallback()**.

void BSP_CAMERA_HwReset (void)

CAMERA hardware reset.

Definition at line **284** of file **stm32446e_eval_camera.c**.

References **BSP_IO_ConfigPin()**, **BSP_IO_PIN_RESET**, **BSP_IO_PIN_SET**, and **BSP_IO_WritePin()**.

Referenced by **BSP_CAMERA_Init()**.

uint8_t BSP_CAMERA_Init (uint32_t Resolution)

Initializes the camera.

Parameters:

Resolution,: camera resolution

Return values:

Camera status

Definition at line **138** of file **stm32446e_eval_camera.c**.

References **BSP_CAMERA_HwReset()**, **BSP_CAMERA_MspInit()**, **BSP_IO_Init()**, **BSP_IO_ReadPin()**, **camera_drv**, **CAMERA_ERROR**, **CAMERA_I2C_ADDRESS**, **CAMERA_OK**, **cameraHwAddress**, **currentResolution**, and **hDcmiEval**.

void BSP_CAMERA_IRQHandler (void)

Handles DCMI interrupt request.

Definition at line **387** of file **stm32446e_eval_camera.c**.

References **hDcmiEval**.

void BSP_CAMERA_LineEventCallback (void)

Line Event callback.

Definition at line **575** of file **stm32446e_eval_camera.c**.

Referenced by **HAL_DCMI_LineEventCallback()**.

void BSP_CAMERA_MspDeInit (DCMI_HandleTypeDef * hdcmi, void * Params)

DeInitializes the DCMI MSP.

Parameters:

hdcmi,:	HDMI handle
Params,:	pointer on additional configuration parameters, can be NULL.

Definition at line **545** of file **stm32446e_eval_camera.c**.

Referenced by **BSP_CAMERA_DeInit()**.

void BSP_CAMERA_MspInit (DCMI_HandleTypeDef * hdcmi,
void * Params
)

Initializes the DCMI MSP.

Parameters:

hdcmi,:	HDMI handle
Params,:	pointer on additional configuration parameters, can be NULL.

Definition at line **446** of file **stm32446e_eval_camera.c**.

Referenced by **BSP_CAMERA_Init()**.

void BSP_CAMERA_PwrDown (void)

CAMERA power down.

Definition at line **312** of file **stm32446e_eval_camera.c**.

References **BSP_IO_ConfigPin()**, **BSP_IO_PIN_RESET**, and **BSP_IO_WritePin()**.

Referenced by **BSP_CAMERA_Stop()**.

void BSP_CAMERA_Resume (void)

Resume the CAMERA capture.

Definition at line **249** of file **stm32446e_eval_camera.c**.

References **hDcmiEval**.

void BSP_CAMERA_SnapshotStart (uint8_t * buff)

Starts the camera capture in snapshot mode.

Parameters:

buff,: pointer to the camera output buffer

Definition at line **228** of file **stm32446e_eval_camera.c**.

References **currentResolution**, **GetSize()**, and **hDcmiEval**.

uint8_t BSP_CAMERA_Stop (void)

Stop the CAMERA capture.

Return values:

Camera status

Definition at line **261** of file **stm32446e_eval_camera.c**.

References **BSP_CAMERA_PwrDown()**, **CAMERA_ERROR**, **CAMERA_OK**, and **hDcmiEval**.

void BSP_CAMERA_Suspend (void)

Suspend the CAMERA capture.

Definition at line **237** of file **stm32446e_eval_camera.c**.

References **hDcmiEval**.

void BSP_CAMERA_VsyncEventCallback (void)

VSYNC Event callback.

Definition at line **594** of file **stm32446e_eval_camera.c**.

Referenced by **HAL_DCMI_VsyncEventCallback()**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL CAMERA Private Functions

STM32446E-EVAL CAMERA

Functions

uint8_t	**BSP_CAMERA_Init** (uint32_t Resolution) Initializes the camera.
uint8_t	**BSP_CAMERA_DeInit** (void) DeInitializes the camera.
void	**BSP_CAMERA_ContinuousStart** (uint8_t *buff) Starts the camera capture in continuous mode.
void	**BSP_CAMERA_SnapshotStart** (uint8_t *buff) Starts the camera capture in snapshot mode.
void	**BSP_CAMERA_Suspend** (void) Suspend the CAMERA capture.
void	**BSP_CAMERA_Resume** (void) Resume the CAMERA capture.
uint8_t	**BSP_CAMERA_Stop** (void) Stop the CAMERA capture.
void	**BSP_CAMERA_HwReset** (void) CAMERA hardware reset.
void	**BSP_CAMERA_PwrDown** (void) CAMERA power down.
void	**BSP_CAMERA_ContrastBrightnessConfig** (uint32_t contrast_level, uint32_t brightness_level) Configures the camera contrast and brightness.
void	**BSP_CAMERA_BlackWhiteConfig** (uint32_t Mode) Configures the camera white balance.
void	**BSP_CAMERA_ColorEffectConfig** (uint32_t Effect) Configures the camera color effect.
void	**BSP_CAMERA_IRQHandler** (void) Handles DCMI interrupt request.
void	**BSP_CAMERA_DMA_IRQHandler** (void) Handles DMA interrupt request.
	BSP_CAMERA_MspInit (DCMI_HandleTypeDef

__weak void	*hdcmi, void *Params) Initializes the DCMI MSP.
__weak void	**BSP_CAMERA_MspDeInit** (DCMI_HandleTypeDef *hdcmi, void *Params) DeInitializes the DCMI MSP.
void	**HAL_DCMI_LineEventCallback** (DCMI_HandleTypeDef *hdcmi) Line event callback.
__weak void	**BSP_CAMERA_LineEventCallback** (void) Line Event callback.
void	**HAL_DCMI_VsyncEventCallback** (DCMI_HandleTypeDef *hdcmi) VSYNC event callback.
__weak void	**BSP_CAMERA_VsyncEventCallback** (void) VSYNC Event callback.
void	**HAL_DCMI_FrameEventCallback** (DCMI_HandleTypeDef *hdcmi) Frame event callback.
__weak void	**BSP_CAMERA_FrameEventCallback** (void) Frame Event callback.
void	**HAL_DCMI_ErrorCallback** (DCMI_HandleTypeDef *hdcmi) Error callback.
__weak void	**BSP_CAMERA_ErrorCallback** (void) Error callback.
static uint32_t	**GetSize** (uint32_t Resolution) Get the capture size.

Function Documentation

void BSP_CAMERA_BlackWhiteConfig (uint32_t Mode)

Configures the camera white balance.

Parameters:

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

- CAMERA_BLACK_WHITE_BW
- CAMERA_BLACK_WHITE_NEGATIVE
- CAMERA_BLACK_WHITE_BW_NEGATIVE
- CAMERA_BLACK_WHITE_NORMAL

Definition at line **359** of file **stm32446e_eval_camera.c**.

References **camera_drv**, and **cameraHwAddress**.

void BSP_CAMERA_ColorEffectConfig (uint32_t Effect)

Configures the camera color effect.

Parameters:

Effect,: Color effect This parameter can be one of the following values:

- CAMERA_COLOR_EFFECT_ANTIQUE
- CAMERA_COLOR_EFFECT_BLUE
- CAMERA_COLOR_EFFECT_GREEN
- CAMERA_COLOR_EFFECT_RED

Definition at line **376** of file **stm32446e_eval_camera.c**.

References **camera_drv**, and **cameraHwAddress**.

void BSP_CAMERA_ContinuousStart (uint8_t * buff)

Starts the camera capture in continuous mode.

Parameters:

buff,: pointer to the camera output buffer

Definition at line **218** of file **stm32446e_eval_camera.c**.

References **currentResolution**, **GetSize()**, and **hDcmiEval**.

void BSP_CAMERA_ContrastBrightnessConfig (uint32_t contrast_
uint32_t brightne
)

Configures the camera contrast and brightness.

Parameters:

contrast_level,: Contrast level This parameter can be one of the following values:

- CAMERA_CONTRAST_LEVEL4: for contrast +2
- CAMERA_CONTRAST_LEVEL3: for contrast +1
- CAMERA_CONTRAST_LEVEL2: for contrast 0
- CAMERA_CONTRAST_LEVEL1: for contrast -1
- CAMERA_CONTRAST_LEVEL0: for contrast -2

brightness_level,: Contrast level This parameter can be one of the following values:

- CAMERA_BRIGHTNESS_LEVEL4: for brightness +2
- CAMERA_BRIGHTNESS_LEVEL3:

for brightness +1
- CAMERA_BRIGHTNESS_LEVEL2: for brightness 0
- CAMERA_BRIGHTNESS_LEVEL1: for brightness -1
- CAMERA_BRIGHTNESS_LEVEL0: for brightness -2

Definition at line **342** of file **stm32446e_eval_camera.c**.

References **camera_drv**, and **cameraHwAddress**.

uint8_t BSP_CAMERA_DeInit (void)

DeInitializes the camera.

Return values:

Camera status

Definition at line **205** of file **stm32446e_eval_camera.c**.

References **BSP_CAMERA_MspDeInit()**, **CAMERA_OK**, and **hDcmiEval**.

void BSP_CAMERA_DMA_IRQHandler (void)

Handles DMA interrupt request.

Definition at line **395** of file **stm32446e_eval_camera.c**.

References **hDcmiEval**.

__weak void BSP_CAMERA_ErrorCallback (void)

Error callback.

Definition at line **632** of file **stm32446e_eval_camera.c**.

Referenced by **HAL_DCMI_ErrorCallback()**.

__weak void BSP_CAMERA_FrameEventCallback (void)

Frame Event callback.

Definition at line **613** of file **stm32446e_eval_camera.c**.

Referenced by **HAL_DCMI_FrameEventCallback()**.

void BSP_CAMERA_HwReset (void)

CAMERA hardware reset.

Definition at line **284** of file **stm32446e_eval_camera.c**.

References **BSP_IO_ConfigPin()**, **BSP_IO_PIN_RESET**, **BSP_IO_PIN_SET**, and **BSP_IO_WritePin()**.

Referenced by **BSP_CAMERA_Init()**.

uint8_t BSP_CAMERA_Init (uint32_t Resolution)

Initializes the camera.

Parameters:

Resolution,: camera resolution

Return values:

Camera status

Definition at line **138** of file **stm32446e_eval_camera.c**.

References **BSP_CAMERA_HwReset()**, **BSP_CAMERA_MspInit()**, **BSP_IO_Init()**, **BSP_IO_ReadPin()**, **camera_drv**, **CAMERA_ERROR**, **CAMERA_I2C_ADDRESS**, **CAMERA_OK**, **cameraHwAddress**, **currentResolution**, and **hDcmiEval**.

void BSP_CAMERA_IRQHandler (void)

Handles DCMI interrupt request.

Definition at line **387** of file **stm32446e_eval_camera.c**.

References **hDcmiEval**.

__weak void BSP_CAMERA_LineEventCallback (void)

Line Event callback.

Definition at line **575** of file **stm32446e_eval_camera.c**.

Referenced by **HAL_DCMI_LineEventCallback()**.

__weak void BSP_CAMERA_MspDeInit (DCMI_HandleTypeDef * hd
void * Pa
)

DeInitializes the DCMI MSP.

Parameters:

hdcmi,:	HDMI handle
Params,:	pointer on additional configuration parameters, can be NULL.

Definition at line **545** of file **stm32446e_eval_camera.c**.

Referenced by **BSP_CAMERA_DeInit()**.

__weak void BSP_CAMERA_MspInit (DCMI_HandleTypeDef * hdcr
void * Para
)

Initializes the DCMI MSP.

Parameters:

hdcmi,:	HDMI handle
Params,:	pointer on additional configuration parameters, can be NULL.

Definition at line **446** of file **stm32446e_eval_camera.c**.

Referenced by **BSP_CAMERA_Init()**.

void BSP_CAMERA_PwrDown (void)

CAMERA power down.

Definition at line **312** of file **stm32446e_eval_camera.c**.

References **BSP_IO_ConfigPin()**, **BSP_IO_PIN_RESET**, and **BSP_IO_WritePin()**.

Referenced by **BSP_CAMERA_Stop()**.

void BSP_CAMERA_Resume (void)

Resume the CAMERA capture.

Definition at line **249** of file **stm32446e_eval_camera.c**.

References **hDcmiEval**.

void BSP_CAMERA_SnapshotStart (uint8_t * buff)

Starts the camera capture in snapshot mode.

Parameters:

buff,: pointer to the camera output buffer

Definition at line **228** of file **stm32446e_eval_camera.c**.

References **currentResolution**, **GetSize()**, and **hDcmiEval**.

uint8_t BSP_CAMERA_Stop (void)

Stop the CAMERA capture.

Return values:

Camera status

Definition at line **261** of file **stm32446e_eval_camera.c**.

References **BSP_CAMERA_PwrDown()**, **CAMERA_ERROR**, **CAMERA_OK**, and **hDcmiEval**.

void BSP_CAMERA_Suspend (void)

Suspend the CAMERA capture.

Definition at line **237** of file **stm32446e_eval_camera.c**.

References **hDcmiEval**.

__weak void BSP_CAMERA_VsyncEventCallback (void)

VSYNC Event callback.

Definition at line **594** of file **stm32446e_eval_camera.c**.

Referenced by **HAL_DCMI_VsyncEventCallback()**.

static uint32_t GetSize (uint32_t Resolution) [static]

Get the capture size.

Parameters:

Resolution,: the current resolution.

Return values:

capture size.

Definition at line **405** of file **stm32446e_eval_camera.c**.

Referenced by **BSP_CAMERA_ContinuousStart()**, and **BSP_CAMERA_SnapshotStart()**.

void HAL_DCMI_ErrorCallback (DCMI_HandleTypeDef * hdcmi)

Error callback.

Parameters:

hdcmi,: pointer to the DCMI handle

Definition at line **624** of file **stm32446e_eval_camera.c**.

References **BSP_CAMERA_ErrorCallback()**.

void HAL_DCMI_FrameEventCallback (DCMI_HandleTypeDef * hdc

Frame event callback.

Parameters:

hdcmi,: pointer to the DCMI handle

Definition at line **605** of file **stm32446e_eval_camera.c**.

References **BSP_CAMERA_FrameEventCallback()**.

void HAL_DCMI_LineEventCallback (DCMI_HandleTypeDef * hdcm

Line event callback.

Parameters:

hdcmi,: pointer to the DCMI handle

Definition at line **567** of file **stm32446e_eval_camera.c**.

References **BSP_CAMERA_LineEventCallback()**.

void HAL_DCMI_VsyncEventCallback (DCMI_HandleTypeDef * hdc

VSYNC event callback.

Parameters:

hdcmi,: pointer to the DCMI handle

Definition at line **586** of file **stm32446e_eval_camera.c**.

References **BSP_CAMERA_VsyncEventCallback()**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL LOW LEVEL Exported Functions

STM32446E EVAL LOW LEVEL

Functions

uint32_t	**BSP_GetVersion** (void) This method returns the STM32446E EVAL BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LEDs.
void	**BSP_LED_DeInit** (**Led_TypeDef** Led) DeInit LEDs.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** ButtonMode) Configures button GPIO and EXTI Line.
void	**BSP_PB_DeInit** (**Button_TypeDef** Button) Push Button DeInit.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *husart) Configures COM port.
void	**BSP_COM_DeInit** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) DeInit COM port.

Function Documentation

void BSP_COM_DeInit (COM_TypeDef COM,
UART_HandleTypeDef * huart
)

DeInit COM port.

Parameters:

COM,: COM port to be configured. This parameter can be one of the following values:
- COM1
- COM2

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

Definition at line **573** of file **stm32446e_eval.c**.

References **COM_USART**, and **EVAL_COMx_CLK_DISABLE**.

void BSP_COM_Init (COM_TypeDef COM,
UART_HandleTypeDef * huart
)

Configures COM port.

Parameters:

COM,: COM port to be configured. This parameter can be one of the following values:
- COM1
- COM2

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

USART peripheral.

Definition at line **534** of file **stm32446e_eval.c**.

References **COM_RX_AF**, **COM_RX_PIN**, **COM_RX_PORT**, **COM_TX_AF**, **COM_TX_PIN**, **COM_TX_PORT**, **COM_USART**, **EVAL_COMx_CLK_ENABLE**, **EVAL_COMx_RX_GPIO_CLK_ENABLE**, and **EVAL_COMx_TX_GPIO_CLK_ENABLE**.

uint32_t BSP_GetVersion (void)

This method returns the STM32446E EVAL BSP Driver revision.

Return values:

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

Definition at line **230** of file **stm32446e_eval.c**.

References **__STM32446E_EVAL_BSP_VERSION**.

void BSP_LED_DeInit (Led_TypeDef Led)

DeInit LEDs.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Note:

Led DeInit does not disable the GPIO clock nor disable the Mfx

Definition at line **296** of file **stm32446e_eval.c**.

References **BSP_IO_ConfigPin()**, **GPIO_PIN**, **LED1**, **LED3**, and **LEDx_GPIO_PORT**.

void BSP_LED_Init (Led_TypeDef Led)

Configures LEDs.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Definition at line **244** of file **stm32446e_eval.c**.

References **BSP_IO_ConfigPin()**, **BSP_IO_Init()**, **BSP_IO_PIN_SET**, **BSP_IO_WritePin()**, **GPIO_PIN**, **LED1**, **LED3**, **LEDx_GPIO_CLK_ENABLE**, and **LEDx_GPIO_PORT**.

void BSP_LED_Off (Led_TypeDef Led)

Turns selected LED Off.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Definition at line **369** of file **stm32446e_eval.c**.

References **BSP_IO_PIN_SET**, **BSP_IO_WritePin()**, **GPIO_PIN**, **LED1**, **LED3**, and **LEDx_GPIO_PORT**.

void BSP_LED_On (Led_TypeDef Led)

Turns selected LED On.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Definition at line **335** of file **stm32446e_eval.c**.

References **BSP_IO_PIN_RESET**, **BSP_IO_WritePin()**, **GPIO_PIN**, **LED1**, **LED3**, and **LEDx_GPIO_PORT**.

void BSP_LED_Toggle (Led_TypeDef Led)

Toggles the selected LED.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Definition at line **404** of file **stm32446e_eval.c**.

References **BSP_IO_TogglePin()**, **GPIO_PIN**, **LED1**, **LED3**, and **LEDx_GPIO_PORT**.

void BSP_PB_DeInit (Button_TypeDef Button)

Push Button DeInit.

Parameters:

Button,: Button to be configured This parameter can be one of the following values:
- BUTTON_WAKEUP: Wakeup Push Button
- BUTTON_TAMPER: Tamper Push Button
- BUTTON_KEY: Key Push Button

Note:

On STM32446E-EVAL evaluation board, the three buttons (Wakeup, Tamper and key buttons) are mapped on the same push button nammed "Wakeup/Tamper" on the board serigraphy.
PB DeInit does not disable the GPIO clock

Definition at line **498** of file **stm32446e_eval.c**.

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

uint32_t BSP_PB_GetState (Button_TypeDef Button)

Returns the selected button state.

Parameters:

Button,: Button to be checked This parameter can be one of the following values:
- BUTTON_WAKEUP: Wakeup Push Button
- BUTTON_TAMPER: Tamper Push Button
- BUTTON_KEY: Key Push Button

Note:

On STM32446E-EVAL evaluation board, the three buttons (Wakeup, Tamper and key buttons) are mapped on the same push button nammed "Wakeup/Tamper" on the board serigraphy.

Return values:

The Button GPIO pin value

Definition at line **520** of file **stm32446e_eval.c**.

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

void BSP_PB_Init (**Button_TypeDef** **Button,**
ButtonMode_TypeDef **ButtonMode**
)

Configures button GPIO and EXTI Line.

Parameters:

Button,: Button to be configured This parameter can be one of the following values:
- BUTTON_WAKEUP: Wakeup Push Button
- BUTTON_TAMPER: Tamper Push Button
- BUTTON_KEY: Key Push Button

ButtonMode,: Button mode This parameter can be one of the following values:
- BUTTON_MODE_GPIO: Button will be used as simple IO
- BUTTON_MODE_EXTI: Button will be connected to EXTI line with interrupt generation capability

Note:

On STM32446E-EVAL evaluation board, the three buttons (Wakeup, Tamper and key buttons) are mapped on the same

push button named "Wakeup/Tamper" on the board serigraphy.

Definition at line **445** of file **stm32446e_eval.c**.

References **BUTTON_IRQn**, **BUTTON_MODE_EXTI**, **BUTTON_MODE_GPIO**, **BUTTON_PIN**, **BUTTON_PORT**, **BUTTON_WAKEUP**, and **BUTTONx_GPIO_CLK_ENABLE**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL EEPROM Private Functions

STM32446E EVAL EEPROM

Functions

uint32_t	**BSP_EEPROM_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
uint8_t	**BSP_EEPROM_DeInit** (void) DeInitializes the EEPROM.
uint32_t	**BSP_EEPROM_ReadBuffer** (uint8_t *pBuffer, uint16_t ReadAddr, uint16_t *NumByteToRead) Reads a block of data from the EEPROM.
uint32_t	**BSP_EEPROM_WritePage** (uint8_t *pBuffer, uint16_t WriteAddr, uint8_t *NumByteToWrite) Writes more than one byte to the EEPROM with a single WRITE cycle.
uint32_t	**BSP_EEPROM_WriteBuffer** (uint8_t *pBuffer, uint16_t WriteAddr, uint16_t NumByteToWrite) Writes buffer of data to the I2C EEPROM.
uint32_t	**BSP_EEPROM_WaitEepromStandbyState** (void) Wait for EEPROM Standby state.
__weak void	**BSP_EEPROM_TIMEOUT_UserCallback** (void) Basic management of the timeout situation.

Function Documentation

uint8_t BSP_EEPROM_DeInit (void)

DeInitializes the EEPROM.

Return values:

EEPROM state

Definition at line **178** of file **stm32446e_eval_eeprom.c**.

References **EEPROM_OK**.

uint32_t BSP_EEPROM_Init (void)

Initializes peripherals used by the I2C EEPROM driver.

Note:

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

Return values:

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

Definition at line **155** of file **stm32446e_eval_eeprom.c**.

References **EEPROM_FAIL**, **EEPROM_I2C_ADDRESS_A01**, **EEPROM_I2C_ADDRESS_A02**, **EEPROM_IO_Init()**, **EEPROM_IO_IsDeviceReady()**, **EEPROM_MAX_TRIALS**, **EEPROM_OK**, and **EEPROMAddress**.

uint32_t BSP_EEPROM_ReadBuffer (uint8_t * pBuffer,

uint16_t ReadAddr,
uint16_t * NumByteToRead
)

Reads a block of data from the EEPROM.

Parameters:

pBuffer,:	pointer to the buffer that receives the data read from the EEPROM.
ReadAddr,:	EEPROM's internal address to start reading from.
NumByteToRead,:	pointer to the variable holding number of bytes to be read from the EEPROM.

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

Return values:

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

Definition at line **199** of file **stm32446e_eval_eeprom.c**.

References **BSP_EEPROM_TIMEOUT_UserCallback()**, **EEPROM_FAIL**, **EEPROM_IO_ReadData()**, **EEPROM_OK**, **EEPROMAddress**, and **EEPROMDataRead**.

__weak void BSP_EEPROM_TIMEOUT_UserCallback (void)

Basic management of the timeout situation.

Definition at line **453** of file **stm32446e_eval_eeprom.c**.

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

uint32_t BSP_EEPROM_WaitEepromStandbyState (void)

Wait for EEPROM Standby state.

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

Return values:

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

Definition at line **438** of file **stm32446e_eval_eeprom.c**.

References **BSP_EEPROM_TIMEOUT_UserCallback()**, **EEPROM_IO_IsDeviceReady()**, **EEPROM_MAX_TRIALS**, **EEPROM_OK**, **EEPROM_TIMEOUT**, and **EEPROMAddress**.

Referenced by **BSP_EEPROM_WritePage()**.

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

Writes buffer of data to the I2C EEPROM.

Parameters:

pBuffer,:	pointer to the buffer containing the data to be written to the EEPROM.
WriteAddr,:	EEPROM's internal address to write to.
NumByteToWrite,:	number of bytes to write to the EEPROM.

Return values:

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

Definition at line **281** of file **stm32446e_eval_eeprom.c**.

References **BSP_EEPROM_WritePage()**, **EEPROM_OK**, and **EEPROM_PAGESIZE**.

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

Writes more than one byte to the EEPROM with a single WRITE cycle.

Note:

The number of bytes (combined to write start address) must not cross the EEPROM page boundary. This function can only write into the boundaries of an EEPROM page. This function doesn't check on boundaries condition (in this driver the function **BSP_EEPROM_WriteBuffer()** which calls **BSP_EEPROM_WritePage()** is responsible of checking on Page boundaries).

Parameters:

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

Note:

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

Return values:

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

Definition at line **246** of file **stm32446e_eval_eeprom.c**.

References **BSP_EEPROM_TIMEOUT_UserCallback()**, **BSP_EEPROM_WaitEepromStandbyState()**, **EEPROM_FAIL**, **EEPROM_IO_WriteData()**, **EEPROM_OK**, **EEPROMAddress**, and **EEPROMDataWrite**.

Referenced by **BSP_EEPROM_WriteBuffer()**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL EEPROM Exported Functions

STM32446E EVAL EEPROM

Functions

uint32_t	**BSP_EEPROM_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
uint8_t	**BSP_EEPROM_DeInit** (void) DeInitializes the EEPROM.
uint32_t	**BSP_EEPROM_ReadBuffer** (uint8_t *pBuffer, uint16_t ReadAddr, uint16_t *NumByteToRead) Reads a block of data from the EEPROM.
uint32_t	**BSP_EEPROM_WritePage** (uint8_t *pBuffer, uint16_t WriteAddr, uint8_t *NumByteToWrite) Writes more than one byte to the EEPROM with a single WRITE cycle.
uint32_t	**BSP_EEPROM_WriteBuffer** (uint8_t *pBuffer, uint16_t WriteAddr, uint16_t NumByteToWrite) Writes buffer of data to the I2C EEPROM.
uint32_t	**BSP_EEPROM_WaitEepromStandbyState** (void) Wait for EEPROM Standby state.
void	**BSP_EEPROM_TIMEOUT_UserCallback** (void) Basic management of the timeout situation.
void	**EEPROM_IO_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_IO_WriteData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Write data to I2C EEPROM driver in using DMA channel.
HAL_StatusTypeDef	**EEPROM_IO_ReadData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize)

	Read data from I2C EEPROM driver in using DMA channel.
HAL_StatusTypeDef	**EEPROM_IO_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.

Function Documentation

uint8_t BSP_EEPROM_DeInit (void)

DeInitializes the EEPROM.

Return values:

EEPROM state

Definition at line **178** of file **stm32446e_eval_eeprom.c**.

References **EEPROM_OK**.

uint32_t BSP_EEPROM_Init (void)

Initializes peripherals used by the I2C EEPROM driver.

Note:

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

Return values:

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

Definition at line **155** of file **stm32446e_eval_eeprom.c**.

References **EEPROM_FAIL**, **EEPROM_I2C_ADDRESS_A01**, **EEPROM_I2C_ADDRESS_A02**, **EEPROM_IO_Init()**, **EEPROM_IO_IsDeviceReady()**, **EEPROM_MAX_TRIALS**, **EEPROM_OK**, and **EEPROMAddress**.

uint32_t BSP_EEPROM_ReadBuffer (uint8_t * pBuffer,

uint16_t ReadAddr,
uint16_t * NumByteToRead
)

Reads a block of data from the EEPROM.

Parameters:

pBuffer,:	pointer to the buffer that receives the data read from the EEPROM.
ReadAddr,:	EEPROM's internal address to start reading from.
NumByteToRead,:	pointer to the variable holding number of bytes to be read from the EEPROM.

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

Return values:

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

Definition at line **199** of file **stm32446e_eval_eeprom.c**.

References **BSP_EEPROM_TIMEOUT_UserCallback()**, **EEPROM_FAIL**, **EEPROM_IO_ReadData()**, **EEPROM_OK**, **EEPROMAddress**, and **EEPROMDataRead**.

void BSP_EEPROM_TIMEOUT_UserCallback (void)

Basic management of the timeout situation.

Definition at line **453** of file **stm32446e_eval_eeprom.c**.

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

uint32_t BSP_EEPROM_WaitEepromStandbyState (void)

Wait for EEPROM Standby state.

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

Return values:

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

Definition at line **438** of file **stm32446e_eval_eeprom.c**.

References **BSP_EEPROM_TIMEOUT_UserCallback()**, **EEPROM_IO_IsDeviceReady()**, **EEPROM_MAX_TRIALS**, **EEPROM_OK**, **EEPROM_TIMEOUT**, and **EEPROMAddress**.

Referenced by **BSP_EEPROM_WritePage()**.

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

Writes buffer of data to the I2C EEPROM.

Parameters:

pBuffer,:	pointer to the buffer containing the data to be written to the EEPROM.
WriteAddr,:	EEPROM's internal address to write to.
NumByteToWrite,:	number of bytes to write to the EEPROM.

Return values:

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

Definition at line **281** of file **stm32446e_eval_eeprom.c**.

References **BSP_EEPROM_WritePage()**, **EEPROM_OK**, and **EEPROM_PAGESIZE**.

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

Writes more than one byte to the EEPROM with a single WRITE cycle.

Note:

The number of bytes (combined to write start address) must not cross the EEPROM page boundary. This function can only write into the boundaries of an EEPROM page. This function doesn't check on boundaries condition (in this driver the function **BSP_EEPROM_WriteBuffer()** which calls **BSP_EEPROM_WritePage()** is responsible of checking on Page boundaries).

Parameters:

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

Note:

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

Return values:

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

Definition at line **246** of file **stm32446e_eval_eeprom.c**.

References **BSP_EEPROM_TIMEOUT_UserCallback()**, **BSP_EEPROM_WaitEepromStandbyState()**, **EEPROM_FAIL**, **EEPROM_IO_WriteData()**, **EEPROM_OK**, **EEPROMAddress**, and **EEPROMDataWrite**.

Referenced by **BSP_EEPROM_WriteBuffer()**.

void EEPROM_IO_Init (void)

Initializes peripherals used by the I2C EEPROM driver.

Definition at line **1251** of file **stm32446e_eval.c**.

References **I2Cx_Init()**.

Referenced by **BSP_EEPROM_Init()**.

HAL_StatusTypeDef EEPROM_IO_IsDeviceReady (uint16_t DevAd
uint32_t Trials
)

Checks if target device is ready for communication.

Note:
This function is used with Memory devices

Parameters:

DevAddress,: Target device address
Trials,: Number of trials

Return values:

HAL status

Definition at line **1289** of file **stm32446e_eval.c**.

References **I2Cx_IsDeviceReady()**.

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

HAL_StatusTypeDef EEPROM_IO_ReadData (uint16_t DevAddres
uint16_t MemAddres
uint8_t * pBuffer,
uint32_t BufferSize
)

Read data from I2C EEPROM driver in using DMA channel.

Parameters:

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

Return values:
HAL status

Definition at line **1277** of file **stm32446e_eval.c**.

References **I2Cx_ReadMultiple()**.

Referenced by **BSP_EEPROM_ReadBuffer()**.

HAL_StatusTypeDef EEPROM_IO_WriteData (uint16_t DevAddress, uint16_t MemAddress, uint8_t * pBuffer, uint32_t BufferSize)

Write data to I2C EEPROM driver in using DMA channel.

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

Return values:
HAL status

Definition at line **1264** of file **stm32446e_eval.c**.

References **I2Cx_WriteMultiple()**.

Referenced by **BSP_EEPROM_WritePage()**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL IO
Private Functions

STM32446E EVAL IO

Functions

uint8_t	**BSP_IO_Init** (void) Initializes and configures the IO functionalities and configures all necessary hardware resources (MFX, ...).
uint8_t	**BSP_IO_DeInit** (void) DeInit allows Mfx Initialization to be executed again.
uint32_t	**BSP_IO_ITGetStatus** (uint32_t IoPin) Gets the selected pins IT status.
void	**BSP_IO_ITClear** (void) Clears all the IO IT pending bits.
uint8_t	**BSP_IO_ConfigPin** (uint32_t IoPin, IO_ModeTypedef IoMode) Configures the IO pin(s) according to IO mode structure value.
uint8_t	**BSP_IO_ConfigIrqOutPin** (uint8_t IoIrqOutPinPolarity, uint8_t IoIrqOutPinType) Sets the IRQ_OUT pin polarity and type.
void	**BSP_IO_WritePin** (uint32_t IoPin, **BSP_IO_PinStateTypeDef** PinState) Sets the selected pins state.
uint32_t	**BSP_IO_ReadPin** (uint32_t IoPin) Gets the selected pins current state.
void	**BSP_IO_TogglePin** (uint32_t IoPin) Toggles the selected pins state.

Function Documentation

uint8_t BSP_IO_ConfigIrqOutPin (uint8_t IoIrqOutPinPolarity, uint8_t IoIrqOutPinType)

Sets the IRQ_OUT pin polarity and type.

Parameters:

IoIrqOutPinPolarity,: High/Low

IoIrqOutPinType,: OpenDrain/PushPull

Return values:

OK

Definition at line **251** of file **stm32446e_eval_io.c**.

References **IO_I2C_ADDRESS**, **IO_OK**, and **mfxstm32l152Identifier**.

uint8_t BSP_IO_ConfigPin (uint32_t IoPin, IO_ModeTypedef IoMode)

Configures the IO pin(s) according to IO mode structure value.

Parameters:

IoPin,: IO pin(s) to be configured. This parameter can be one of the following values:

- MFXSTM32L152_GPIO_PIN_x: where x can be from 0 to 23.

IoMode,: IO pin mode to configure This parameter can be one of the following values:

- IO_MODE_INPUT

- IO_MODE_OUTPUT
- IO_MODE_IT_RISING_EDGE
- IO_MODE_IT_FALLING_EDGE
- IO_MODE_IT_LOW_LEVEL
- IO_MODE_IT_HIGH_LEVEL
- IO_MODE_ANALOG
- IO_MODE_OFF
- IO_MODE_INPUT_PU,
- IO_MODE_INPUT_PD,
- IO_MODE_OUTPUT_OD,
- IO_MODE_OUTPUT_OD_PU,
- IO_MODE_OUTPUT_OD_PD,
- IO_MODE_OUTPUT_PP,
- IO_MODE_OUTPUT_PP_PU,
- IO_MODE_OUTPUT_PP_PD,
- IO_MODE_IT_RISING_EDGE_PU
- IO_MODE_IT_FALLING_EDGE_PU
- IO_MODE_IT_LOW_LEVEL_PU
- IO_MODE_IT_HIGH_LEVEL_PU
- IO_MODE_IT_RISING_EDGE_PD
- IO_MODE_IT_FALLING_EDGE_PD
- IO_MODE_IT_LOW_LEVEL_PD
- IO_MODE_IT_HIGH_LEVEL_PD

Return values:

IO_OK if all initializations are OK. Other value if error.

Definition at line **237** of file **stm32446e_eval_io.c**.

References **IO_I2C_ADDRESS**, **IO_OK**, and **IoDrv**.

Referenced by **BSP_CAMERA_HwReset()**, **BSP_CAMERA_PwrDown()**, **BSP_LED_DeInit()**, **BSP_LED_Init()**, **BSP_SD_DeInit()**, **BSP_SD_Init()**, **BSP_SD_IsDetected()**, and **BSP_SD_MspInit()**.

uint8_t BSP_IO_DeInit (void)

DeInit allows Mfx Initialization to be executed again.

Note:
BSP_IO_Init() has no effect if the IoDrv is already initialized
BSP_IO_DeInit() allows to erase the pointer such to allow init to be effective

Return values:
IO_OK

Definition at line 177 of file stm32446e_eval_io.c.

References IO_OK, and IoDrv.

uint8_t BSP_IO_Init (void)

Initializes and configures the IO functionalities and configures all necessary hardware resources (MFX, ...).

Note:
BSP_IO_Init() is using HAL_Delay() function to ensure that MFXSTM32L152 IO Expander is correctly reset. HAL_Delay() function provides accurate delay (in milliseconds) based on variable incremented in SysTick ISR. This implies that if BSP_IO_Init() is called from a peripheral ISR process, then the SysTick interrupt must have higher priority (numerically lower) than the peripheral interrupt. Otherwise the caller ISR process will be blocked.

Return values:
IO_OK if all initializations are OK. Other value if error.

Definition at line 142 of file stm32446e_eval_io.c.

References **IO_ERROR**, **IO_I2C_ADDRESS**, **IO_OK**, **IO_PIN_ALL**, **IoDrv**, and **mfxstm32l152Identifier**.

Referenced by **BSP_CAMERA_Init()**, **BSP_LED_Init()**, **BSP_SD_Init()**, **BSP_TS_Init()**, and **BSP_TS_ITConfig()**.

void BSP_IO_ITClear (void)

Clears all the IO IT pending bits.

Definition at line **198** of file **stm32446e_eval_io.c**.

References **IO_I2C_ADDRESS**, and **IoDrv**.

uint32_t BSP_IO_ITGetStatus (uint32_t IoPin)

Gets the selected pins IT status.

Parameters:

IoPin,: Selected pins to check the status. This parameter can be any combination of the IO pins.

Return values:

IO_OK if read status OK. Other value if error.

Definition at line **189** of file **stm32446e_eval_io.c**.

References **IO_I2C_ADDRESS**, and **IoDrv**.

uint32_t BSP_IO_ReadPin (uint32_t IoPin)

Gets the selected pins current state.

Parameters:

IoPin,: Selected pins to read. This parameter can be any combination of the IO pins.

Return values:

The current pins state

Definition at line **281** of file **stm32446e_eval_io.c**.

References **IO_I2C_ADDRESS**, and **IoDrv**.

Referenced by **BSP_CAMERA_Init()**, and **BSP_SD_IsDetected()**.

void BSP_IO_TogglePin (uint32_t IoPin)

Toggles the selected pins state.

Parameters:

IoPin,: Selected pins to toggle. This parameter can be any combination of the IO pins.

Note:

This function is only used to toggle one pin in the same time

Definition at line **292** of file **stm32446e_eval_io.c**.

References **IO_I2C_ADDRESS**, and **IoDrv**.

Referenced by **BSP_LED_Toggle()**.

void BSP_IO_WritePin (uint32_t IoPin, BSP_IO_PinStateTypeDef PinState)

Sets the selected pins state.

Parameters:

IoPin,:	Selected pins to write. This parameter can be any combination of the IO pins.
PinState,:	New pins state to write

Definition at line **269** of file **stm32446e_eval_io.c**.

References **IO_I2C_ADDRESS**, and **IoDrv**.

Referenced by **BSP_CAMERA_HwReset()**, **BSP_CAMERA_PwrDown()**, **BSP_LED_Init()**, **BSP_LED_Off()**, **BSP_LED_On()**, and **BSP_SD_MspInit()**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL IO Exported Functions

STM32446E EVAL IO

Functions

uint8_t	**BSP_IO_Init** (void) Initializes and configures the IO functionalities and configures all necessary hardware resources (MFX, ...).
uint8_t	**BSP_IO_DeInit** (void) DeInit allows Mfx Initialization to be executed again.
uint8_t	**BSP_IO_ConfigIrqOutPin** (uint8_t IoIrqOutPinPolarity, uint8_t IoIrqOutPinType) Sets the IRQ_OUT pin polarity and type.
uint32_t	**BSP_IO_ITGetStatus** (uint32_t IoPin) Gets the selected pins IT status.
void	**BSP_IO_ITClear** (void) Clears all the IO IT pending bits.
uint8_t	**BSP_IO_ConfigPin** (uint32_t IoPin, IO_ModeTypedef IoMode) Configures the IO pin(s) according to IO mode structure value.
void	**BSP_IO_WritePin** (uint32_t IoPin, **BSP_IO_PinStateTypeDef** PinState) Sets the selected pins state.
uint32_t	**BSP_IO_ReadPin** (uint32_t IoPin) Gets the selected pins current state.
void	**BSP_IO_TogglePin** (uint32_t IoPin) Toggles the selected pins state.

Function Documentation

uint8_t BSP_IO_ConfigIrqOutPin (uint8_t IoIrqOutPinPolarity, uint8_t IoIrqOutPinType)

Sets the IRQ_OUT pin polarity and type.

Parameters:

IoIrqOutPinPolarity,: High/Low

IoIrqOutPinType,: OpenDrain/PushPull

Return values:

OK

Definition at line **251** of file **stm32446e_eval_io.c**.

References **IO_I2C_ADDRESS**, **IO_OK**, and **mfxstm32l152Identifier**.

uint8_t BSP_IO_ConfigPin (uint32_t IoPin, IO_ModeTypedef IoMode)

Configures the IO pin(s) according to IO mode structure value.

Parameters:

IoPin,: IO pin(s) to be configured. This parameter can be one of the following values:

- MFXSTM32L152_GPIO_PIN_x: where x can be from 0 to 23.

IoMode,: IO pin mode to configure This parameter can be one of the following values:

- IO_MODE_INPUT

- IO_MODE_OUTPUT
- IO_MODE_IT_RISING_EDGE
- IO_MODE_IT_FALLING_EDGE
- IO_MODE_IT_LOW_LEVEL
- IO_MODE_IT_HIGH_LEVEL
- IO_MODE_ANALOG
- IO_MODE_OFF
- IO_MODE_INPUT_PU,
- IO_MODE_INPUT_PD,
- IO_MODE_OUTPUT_OD,
- IO_MODE_OUTPUT_OD_PU,
- IO_MODE_OUTPUT_OD_PD,
- IO_MODE_OUTPUT_PP,
- IO_MODE_OUTPUT_PP_PU,
- IO_MODE_OUTPUT_PP_PD,
- IO_MODE_IT_RISING_EDGE_PU
- IO_MODE_IT_FALLING_EDGE_PU
- IO_MODE_IT_LOW_LEVEL_PU
- IO_MODE_IT_HIGH_LEVEL_PU
- IO_MODE_IT_RISING_EDGE_PD
- IO_MODE_IT_FALLING_EDGE_PD
- IO_MODE_IT_LOW_LEVEL_PD
- IO_MODE_IT_HIGH_LEVEL_PD

Return values:

IO_OK if all initializations are OK. Other value if error.

Definition at line **237** of file **stm32446e_eval_io.c**.

References **IO_I2C_ADDRESS**, **IO_OK**, and **IoDrv**.

Referenced by **BSP_CAMERA_HwReset()**, **BSP_CAMERA_PwrDown()**, **BSP_LED_DeInit()**, **BSP_LED_Init()**, **BSP_SD_DeInit()**, **BSP_SD_Init()**, **BSP_SD_IsDetected()**, and **BSP_SD_MspInit()**.

uint8_t BSP_IO_DeInit (void)

DeInit allows Mfx Initialization to be executed again.

Note:
BSP_IO_Init() has no effect if the IoDrv is already initialized
BSP_IO_DeInit() allows to erase the pointer such to allow init to be effective

Return values:
IO_OK

Definition at line 177 of file stm32446e_eval_io.c.

References IO_OK, and IoDrv.

uint8_t BSP_IO_Init (void)

Initializes and configures the IO functionalities and configures all necessary hardware resources (MFX, ...).

Note:
BSP_IO_Init() is using HAL_Delay() function to ensure that MFXSTM32L152 IO Expander is correctly reset. HAL_Delay() function provides accurate delay (in milliseconds) based on variable incremented in SysTick ISR. This implies that if BSP_IO_Init() is called from a peripheral ISR process, then the SysTick interrupt must have higher priority (numerically lower) than the peripheral interrupt. Otherwise the caller ISR process will be blocked.

Return values:
IO_OK if all initializations are OK. Other value if error.

Definition at line 142 of file stm32446e_eval_io.c.

References **IO_ERROR**, **IO_I2C_ADDRESS**, **IO_OK**, **IO_PIN_ALL**, **IoDrv**, and **mfxstm32l152Identifier**.

Referenced by **BSP_CAMERA_Init()**, **BSP_LED_Init()**, **BSP_SD_Init()**, **BSP_TS_Init()**, and **BSP_TS_ITConfig()**.

void BSP_IO_ITClear (void)

Clears all the IO IT pending bits.

Definition at line **198** of file **stm32446e_eval_io.c**.

References **IO_I2C_ADDRESS**, and **IoDrv**.

uint32_t BSP_IO_ITGetStatus (uint32_t IoPin)

Gets the selected pins IT status.

Parameters:

IoPin,: Selected pins to check the status. This parameter can be any combination of the IO pins.

Return values:

IO_OK if read status OK. Other value if error.

Definition at line **189** of file **stm32446e_eval_io.c**.

References **IO_I2C_ADDRESS**, and **IoDrv**.

uint32_t BSP_IO_ReadPin (uint32_t IoPin)

Gets the selected pins current state.

Parameters:

IoPin,: Selected pins to read. This parameter can be any combination of the IO pins.

Return values:

The current pins state

Definition at line **281** of file **stm32446e_eval_io.c**.

References **IO_I2C_ADDRESS**, and **IoDrv**.

Referenced by **BSP_CAMERA_Init()**, and **BSP_SD_IsDetected()**.

void BSP_IO_TogglePin (uint32_t IoPin)

Toggles the selected pins state.

Parameters:

IoPin,: Selected pins to toggle. This parameter can be any combination of the IO pins.

Note:

This function is only used to toggle one pin in the same time

Definition at line **292** of file **stm32446e_eval_io.c**.

References **IO_I2C_ADDRESS**, and **IoDrv**.

Referenced by **BSP_LED_Toggle()**.

void BSP_IO_WritePin (uint32_t IoPin, BSP_IO_PinStateTypeDef PinState)

Sets the selected pins state.

Parameters:

IoPin,:	Selected pins to write. This parameter can be any combination of the IO pins.
PinState,:	New pins state to write

Definition at line **269** of file **stm32446e_eval_io.c**.

References **IO_I2C_ADDRESS**, and **IoDrv**.

Referenced by **BSP_CAMERA_HwReset()**, **BSP_CAMERA_PwrDown()**, **BSP_LED_Init()**, **BSP_LED_Off()**, **BSP_LED_On()**, and **BSP_SD_MspInit()**.

STM32446E_EVAL BSP User Manual

Enumerations

STM32446E EVAL IO Exported Types

STM32446E EVAL IO

Enumerations

enum	**BSP_IO_PinStateTypeDef** { **BSP_IO_PIN_RESET** = 0, **BSP_IO_PIN_SET** = 1 }
enum	**IO_StatusTypeDef** { **IO_OK** = 0, **IO_ERROR** = 1, **IO_TIMEOUT** = 2 }

Enumeration Type Documentation

enum BSP_IO_PinStateTypeDef

Enumerator:

BSP_IO_PIN_RESET

BSP_IO_PIN_SET

Definition at line **68** of file **stm32446e_eval_io.h**.

enum IO_StatusTypeDef

Enumerator:

IO_OK

IO_ERROR

IO_TIMEOUT

Definition at line **74** of file **stm32446e_eval_io.h**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL LCD
Private Functions

STM32446E EVAL LCD

Functions

uint8_t	**BSP_LCD_Init** (void) Initializes the LCD.
uint8_t	**BSP_LCD_DeInit** (void) DeInitializes 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 *fonts) 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 *Text, **Line_ModeTypdef** Mode) Displays characters on the LCD.

void	**BSP_LCD_DisplayStringAtLine** (uint16_t Line, uint8_t *ptr) Displays a character on the LCD.
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) Draws a pixel on LCD.
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** Points, 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 (16 bpp).
void	**BSP_LCD_DrawRGBImage** (uint16_t Xpos, uint16_t Ypos, uint16_t Xsize, uint16_t Ysize, uint8_t *pdata)

	Draws RGB Image (16 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_FillPolygon** (**pPoint** Points, uint16_t PointCount) Draws 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.
static void	**DrawChar** (uint16_t Xpos, uint16_t Ypos, const uint8_t *c) Draws a character on LCD.
static void	**SetDisplayWindow** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Sets display window.
static void	**FillTriangle** (uint16_t x1, uint16_t x2, uint16_t x3, uint16_t y1, uint16_t y2, uint16_t y3) Fills a triangle (between 3 points).

Function Documentation

void BSP_LCD_Clear (uint16_t Color)

Clears the hole LCD.

Parameters:

Color,: Color of the background

Definition at line **247** of file **stm32446e_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

Definition at line **272** of file **stm32446e_eval_lcd.c**.

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

uint8_t BSP_LCD_DeInit (void)

DeInitializes the LCD.

Return values:

LCD state

Definition at line **165** of file **stm32446e_eval_lcd.c**.

References **LCD_OK**.

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

Definition at line **292** of file **stm32446e_eval_lcd.c**.

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

Referenced by **BSP_LCD_DisplayStringAt()**.

void BSP_LCD_DisplayOff (void)

Disables the display.

Definition at line **879** of file **stm32446e_eval_lcd.c**.

References **LcdDrv**.

void BSP_LCD_DisplayOn (void)

Enables the display.

Definition at line **871** of file **stm32446e_eval_lcd.c**.

References **LcdDrv**.

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

Displays characters on the LCD.

Parameters:

- **Xpos,:** X position (in pixel)
- **Ypos,:** Y position (in pixel)
- **Text,:** 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

Definition at line **309** of file **stm32446e_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 * ptr
)

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

ptr,: Pointer to string to display on LCD

Definition at line **373** of file **stm32446e_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 (16 bpp).

Parameters:

Xpos,: Bmp X position in the LCD

Ypos,: Bmp Y position in the LCD

pbmp,: Pointer to Bmp picture address.

Definition at line **662** of file **stm32446e_eval_lcd.c**.

References **BSP_LCD_GetXSize()**, **BSP_LCD_GetYSize()**, **LcdDrv**, and **SetDisplayWindow()**.

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

Definition at line **554** of file **stm32446e_eval_lcd.c**.

References **BSP_LCD_DrawPixel()**, **BSP_LCD_SetFont()**, **LCD_DEFAULT_FONT**, 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

Definition at line **630** of file **stm32446e_eval_lcd.c**.

References **BSP_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

Definition at line **416** of file **stm32446e_eval_lcd.c**.

References **BSP_LCD_DrawPixel()**, **LcdDrv**, and **LCD_DrawPropTypeDef::TextColor**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_DrawRect()**, **BSP_LCD_FillCircle()**, **BSP_LCD_FillEllipse()**, 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

Definition at line **463** of file **stm32446e_eval_lcd.c**.

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

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

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

Draws a pixel on LCD.

Parameters:

- **Xpos,:** X position
- **Ypos,:** Y position
- **RGB_Code,:** Pixel color in RGB mode (5-6-5)

Definition at line **402** of file **stm32446e_eval_lcd.c**.

References **LcdDrv**.

Referenced by **BSP_LCD_DrawCircle()**, **BSP_LCD_DrawEllipse()**, **BSP_LCD_DrawHLine()**, **BSP_LCD_DrawLine()**, **BSP_LCD_DrawVLine()**, and **DrawChar()**.

void BSP_LCD_DrawPolygon (pPoint Points,
uint16_t PointCount
)

Draws an poly-line (between many points).

Parameters:

Points,: Pointer to the points array

PointCount,: Number of points

Definition at line **603** of file **stm32446e_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

Definition at line **537** of file **stm32446e_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 *  pdata
                                  )
```

Draws RGB Image (16 bpp).

Parameters:

- **Xpos,:** X position in the LCD
- **Ypos,:** Y position in the LCD
- **Xsize,:** X size in the LCD
- **Ysize,:** Y size in the LCD
- **pdata,:** Pointer to the RGB Image address.

Definition at line **693** of file **stm32446e_eval_lcd.c**.

References **BSP_LCD_GetXSize()**, **BSP_LCD_GetYSize()**, **LcdDrv**, and **SetDisplayWindow()**.

```
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

Definition at line **439** of file **stm32446e_eval_lcd.c**.

References **BSP_LCD_DrawPixel()**, **LcdDrv**, and **LCD_DrawPropTypeDef::TextColor**.

Referenced by **BSP_LCD_DrawRect()**.

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

Definition at line **728** of file **stm32446e_eval_lcd.c**.

References **BSP_LCD_DrawCircle()**, **BSP_LCD_DrawHLine()**, **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

Definition at line **842** of file **stm32446e_eval_lcd.c**.

References **BSP_LCD_DrawHLine()**.

void BSP_LCD_FillPolygon (pPoint Points,
uint16_t PointCount
)

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

Parameters:
Points,: Pointer to the points array
PointCount,: Number of points

Definition at line **775** of file **stm32446e_eval_lcd.c**.

References **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

Definition at line **712** of file **stm32446e_eval_lcd.c**.

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

Referenced by **BSP_LCD_ClearStringLine()**.

uint16_t BSP_LCD_GetBackColor (void)

Gets the LCD background color.

Return values:

Used background color

Definition at line **202** of file **stm32446e_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**.

sFONT* BSP_LCD_GetFont (void)

Gets the LCD text font.

Return values:

Used font

Definition at line **238** of file **stm32446e_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 **193** of file **stm32446e_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 **175** of file **stm32446e_eval_lcd.c**.

References **LcdDrv**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_ClearStringLine()**, **BSP_LCD_DisplayStringAt()**, **BSP_LCD_DrawBitmap()**, and **BSP_LCD_DrawRGBImage()**.

uint32_t BSP_LCD_GetYSize (void)

Gets the LCD Y size.

Return values:

Used LCD Y size

Definition at line **184** of file **stm32446e_eval_lcd.c**.

References **LcdDrv**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_DrawBitmap()**, and **BSP_LCD_DrawRGBImage()**.

uint8_t BSP_LCD_Init (void)

Initializes the LCD.

Return values:

LCD state

Definition at line **136** of file **stm32446e_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**, **BSP_LCD_SetFont()**, **LCD_DEFAULT_FONT**, **LCD_ERROR**, **LCD_OK**, **LcdDrv**, **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 **384** of file **stm32446e_eval_lcd.c**.

References **LcdDrv**.

void BSP_LCD_SetBackColor (uint16_t Color)

Sets the LCD background color.

Parameters:

Color,: Background color code RGB(5-6-5)

Definition at line **220** of file **stm32446e_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**.

void BSP_LCD_SetFont (sFONT * fonts)

Sets the LCD text font.

Parameters:

fonts,: Font to be used

Definition at line **229** of file **stm32446e_eval_lcd.c**.

References **LCD_DrawPropTypeDef::pFont**.

Referenced by **BSP_LCD_DrawCircle()**, and **BSP_LCD_Init()**.

void BSP_LCD_SetTextColor (uint16_t Color)

Sets the LCD text color.

Parameters:

Color,: Text color code RGB(5-6-5)

Definition at line **211** of file **stm32446e_eval_lcd.c**.

References **LCD_DrawPropTypeDef::TextColor**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_ClearStringLine()**, **BSP_LCD_FillCircle()**, and **BSP_LCD_FillRect()**.

static void DrawChar (uint16_t Xpos,

```
                    uint16_t        Ypos,
                    const uint8_t * c
                  )                 [static]
```

Draws a character on LCD.

Parameters:

- **Xpos,:** Line where to display the character shape
- **Ypos,:** Start column address
- **c,:** Pointer to the character data

Definition at line **894** of file **stm32446e_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**, **BSP_LCD_DrawPixel()**, **LCD_DrawPropTypeDef::pFont**, and **LCD_DrawPropTypeDef::TextColor**.

Referenced by **BSP_LCD_DisplayChar()**.

```
static void FillTriangle ( uint16_t x1,
                           uint16_t x2,
                           uint16_t x3,
                           uint16_t y1,
                           uint16_t y2,
                           uint16_t y3
                         )          [static]
```

Fills a triangle (between 3 points).

Parameters:

- **x1,:** **Point** 1 X position
- **y1,:** **Point** 1 Y position
- **x2,:** **Point** 2 X position
- **y2,:** **Point** 2 Y position

x3,: **Point** 3 X position
y3,: **Point** 3 Y position

Definition at line **966** of file **stm32446e_eval_lcd.c**.

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

Referenced by **BSP_LCD_FillPolygon()**.

static void 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

Definition at line **949** of file **stm32446e_eval_lcd.c**.

References **LcdDrv**.

Referenced by **BSP_LCD_DrawBitmap()**, and **BSP_LCD_DrawRGBImage()**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL LCD Exported Functions

STM32446E EVAL LCD

Functions

uint8_t	**BSP_LCD_Init** (void) Initializes the LCD.
uint8_t	**BSP_LCD_DeInit** (void) DeInitializes 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 *fonts) 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 *ptr) Displays a character on the LCD.
void	**BSP_LCD_DisplayStringAt** (uint16_t Xpos, uint16_t Ypos, uint8_t *Text, **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) Draws a pixel on LCD.
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** Points, 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 (16 bpp).
void	**BSP_LCD_DrawRGBImage** (uint16_t Xpos, uint16_t Ypos, uint16_t Xsize, uint16_t Ysize, uint8_t *pbmp) Draws RGB Image (16 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_FillPolygon** (**pPoint** Points, uint16_t PointCount) Draws 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_DisplayOff** (void) Disables the display.
void	**BSP_LCD_DisplayOn** (void) Enables the display.

Function Documentation

void BSP_LCD_Clear (uint16_t Color)

Clears the hole LCD.

Parameters:

Color,: Color of the background

Definition at line **247** of file **stm32446e_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

Definition at line **272** of file **stm32446e_eval_lcd.c**.

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

uint8_t BSP_LCD_DeInit (void)

DeInitializes the LCD.

Return values:

LCD state

Definition at line **165** of file **stm32446e_eval_lcd.c**.

References **LCD_OK**.

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

Definition at line **292** of file **stm32446e_eval_lcd.c**.

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

Referenced by **BSP_LCD_DisplayStringAt()**.

void BSP_LCD_DisplayOff (void)

Disables the display.

Definition at line **879** of file **stm32446e_eval_lcd.c**.

References **LcdDrv**.

void BSP_LCD_DisplayOn (void)

Enables the display.

Definition at line **871** of file **stm32446e_eval_lcd.c**.

References **LcdDrv**.

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

Displays characters on the LCD.

Parameters:

- **Xpos,:** X position (in pixel)
- **Ypos,:** Y position (in pixel)
- **Text,:** 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

Definition at line **309** of file **stm32446e_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 * ptr
)

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

ptr,: Pointer to string to display on LCD

Definition at line **373** of file **stm32446e_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 (16 bpp).

Parameters:

Xpos,: Bmp X position in the LCD

Ypos,: Bmp Y position in the LCD

pbmp,: Pointer to Bmp picture address.

Definition at line **662** of file **stm32446e_eval_lcd.c**.

References **BSP_LCD_GetXSize()**, **BSP_LCD_GetYSize()**, **LcdDrv**, and **SetDisplayWindow()**.

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

Definition at line **554** of file **stm32446e_eval_lcd.c**.

References **BSP_LCD_DrawPixel()**, **BSP_LCD_SetFont()**, **LCD_DEFAULT_FONT**, 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

Definition at line **630** of file **stm32446e_eval_lcd.c**.

References **BSP_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

Definition at line **416** of file **stm32446e_eval_lcd.c**.

References **BSP_LCD_DrawPixel()**, **LcdDrv**, and **LCD_DrawPropTypeDef::TextColor**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_DrawRect()**, **BSP_LCD_FillCircle()**, **BSP_LCD_FillEllipse()**, 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

Definition at line **463** of file **stm32446e_eval_lcd.c**.

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

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

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

Draws a pixel on LCD.

Parameters:

- **Xpos,:** X position
- **Ypos,:** Y position
- **RGB_Code,:** Pixel color in RGB mode (5-6-5)

Definition at line **402** of file **stm32446e_eval_lcd.c**.

References **LcdDrv**.

Referenced by **BSP_LCD_DrawCircle()**, **BSP_LCD_DrawEllipse()**, **BSP_LCD_DrawHLine()**, **BSP_LCD_DrawLine()**, **BSP_LCD_DrawVLine()**, and **DrawChar()**.

void BSP_LCD_DrawPolygon (pPoint Points,
uint16_t PointCount
)

Draws an poly-line (between many points).

Parameters:

Points,: Pointer to the points array

PointCount,: Number of points

Definition at line **603** of file **stm32446e_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

Definition at line **537** of file **stm32446e_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 *  pdata
                                  )
```

Draws RGB Image (16 bpp).

Parameters:

- **Xpos,:** X position in the LCD
- **Ypos,:** Y position in the LCD
- **Xsize,:** X size in the LCD
- **Ysize,:** Y size in the LCD
- **pdata,:** Pointer to the RGB Image address.

Definition at line **693** of file **stm32446e_eval_lcd.c**.

References **BSP_LCD_GetXSize()**, **BSP_LCD_GetYSize()**, **LcdDrv**, and **SetDisplayWindow()**.

```
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

Definition at line **439** of file **stm32446e_eval_lcd.c**.

References **BSP_LCD_DrawPixel()**, **LcdDrv**, and **LCD_DrawPropTypeDef::TextColor**.

Referenced by **BSP_LCD_DrawRect()**.

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

Definition at line **728** of file **stm32446e_eval_lcd.c**.

References **BSP_LCD_DrawCircle()**, **BSP_LCD_DrawHLine()**, **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

Definition at line **842** of file **stm32446e_eval_lcd.c**.

References **BSP_LCD_DrawHLine()**.

void BSP_LCD_FillPolygon (pPoint Points,
uint16_t PointCount
)

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

Parameters:
Points,: Pointer to the points array
PointCount,: Number of points

Definition at line **775** of file **stm32446e_eval_lcd.c**.

References **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

Definition at line **712** of file **stm32446e_eval_lcd.c**.

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

Referenced by **BSP_LCD_ClearStringLine()**.

uint16_t BSP_LCD_GetBackColor (void)

Gets the LCD background color.

Return values:

Used background color

Definition at line **202** of file **stm32446e_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**.

sFONT* BSP_LCD_GetFont (void)

Gets the LCD text font.

Return values:

Used font

Definition at line **238** of file **stm32446e_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 **193** of file **stm32446e_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 **175** of file **stm32446e_eval_lcd.c**.

References **LcdDrv**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_ClearStringLine()**, **BSP_LCD_DisplayStringAt()**, **BSP_LCD_DrawBitmap()**, and **BSP_LCD_DrawRGBImage()**.

uint32_t BSP_LCD_GetYSize (void)

Gets the LCD Y size.

Return values:

Used LCD Y size

Definition at line **184** of file **stm32446e_eval_lcd.c**.

References **LcdDrv**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_DrawBitmap()**, and **BSP_LCD_DrawRGBImage()**.

uint8_t BSP_LCD_Init (void)

Initializes the LCD.

Return values:

LCD state

Definition at line **136** of file **stm32446e_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**, **BSP_LCD_SetFont()**, **LCD_DEFAULT_FONT**, **LCD_ERROR**, **LCD_OK**, **LcdDrv**, **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 **384** of file **stm32446e_eval_lcd.c**.

References **LcdDrv**.

void BSP_LCD_SetBackColor (__IO uint16_t Color)

void BSP_LCD_SetFont (sFONT * fonts)

Sets the LCD text font.

Parameters:

fonts,: Font to be used

Definition at line **229** of file **stm32446e_eval_lcd.c**.

References **LCD_DrawPropTypeDef::pFont**.

Referenced by **BSP_LCD_DrawCircle()**, and **BSP_LCD_Init()**.

void BSP_LCD_SetTextColor (__IO uint16_t Color)

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL QSPI Exported Functions

STM32446E EVAL QSPI

Functions

uint8_t	**BSP_QSPI_Init** (void) Initializes the QSPI interface.
uint8_t	**BSP_QSPI_DeInit** (void) De-Initializes the QSPI interface.
uint8_t	**BSP_QSPI_Read** (uint8_t *pData, uint32_t ReadAddr, uint32_t Size) Reads an amount of data from the QSPI memory.
uint8_t	**BSP_QSPI_Write** (uint8_t *pData, uint32_t WriteAddr, uint32_t Size) Writes an amount of data to the QSPI memory.
uint8_t	**BSP_QSPI_Erase_Block** (uint32_t BlockAddress) Erases the specified block of the QSPI memory.
uint8_t	**BSP_QSPI_Erase_Chip** (void) Erases the entire QSPI memory.
uint8_t	**BSP_QSPI_GetStatus** (void) Reads current status of the QSPI memory.
uint8_t	**BSP_QSPI_GetInfo** (**QSPI_Info** *pInfo) Return the configuration of the QSPI memory.
uint8_t	**BSP_QSPI_MemoryMappedMode** (void) Configure the QSPI in memory-mapped mode.
void	**BSP_QSPI_MspInit** (QSPI_HandleTypeDef *hqspi, void *Params) QSPI MSP Initialization This function configures the hardware resources used in this example:
void	**BSP_QSPI_MspDeInit** (QSPI_HandleTypeDef *hqspi, void *Params) QSPI MSP De-Initialization This function frees the hardware resources used in this example:

Function Documentation

uint8_t BSP_QSPI_DeInit (void)

De-Initializes the QSPI interface.

Return values:

QSPI memory status

Definition at line **172** of file **stm32446e_eval_qspi.c**.

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

uint8_t BSP_QSPI_Erase_Block (uint32_t BlockAddress)

Erases the specified block of the QSPI memory.

Parameters:

BlockAddress,: Block address to erase

Return values:

QSPI memory status

Definition at line **315** of file **stm32446e_eval_qspi.c**.

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

uint8_t BSP_QSPI_Erase_Chip (void)

Erases the entire QSPI memory.

Return values:

QSPI memory status

Definition at line **357** of file **stm32446e_eval_qspi.c**.

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

uint8_t BSP_QSPI_GetInfo (QSPI_Info * pInfo)

Return the configuration of the QSPI memory.

Parameters:

pInfo,: pointer on the configuration structure

Return values:

QSPI memory status

Definition at line **450** of file **stm32446e_eval_qspi.c**.

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

uint8_t BSP_QSPI_GetStatus (void)

Reads current status of the QSPI memory.

Return values:

QSPI memory status

Definition at line **397** of file **stm32446e_eval_qspi.c**.

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

uint8_t BSP_QSPI_Init (void)

Initializes the QSPI interface.

Return values:

QSPI memory status

Definition at line **119** of file **stm32446e_eval_qspi.c**.

References **BSP_QSPI_MspInit()**, **QSPI_DummyCyclesCfg()**, **QSPI_EnterFourBytesAddress()**, **QSPI_ERROR**, **QSPI_NOT_SUPPORTED**, **QSPI_OK**, **QSPI_ResetMemory()**, and **QSPIHandle**.

uint8_t BSP_QSPI_MemoryMappedMode (void)

Configure the QSPI in memory-mapped mode.

Return values:

QSPI memory status

Definition at line **466** of file **stm32446e_eval_qspi.c**.

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

void BSP_QSPI_MspDeInit (QSPI_HandleTypeDef * hqspi,
void * Params
)

QSPI MSP De-Initialization This function frees the hardware resources used in this example:

- Disable the Peripheral's clock
- Revert GPIO and NVIC configuration to their default state

Return values:

None

Definition at line **570** of file **stm32446e_eval_qspi.c**.

References **QSPI_CLK_DISABLE**, **QSPI_D0_GPIO_PORT**, **QSPI_D0_PIN**, **QSPI_D1_GPIO_PORT**, **QSPI_D1_PIN**, **QSPI_D2_GPIO_PORT**, **QSPI_D2_PIN**, **QSPI_D3_GPIO_PORT**, **QSPI_D3_PIN**, **QSPI_FORCE_RESET**, and **QSPI_RELEASE_RESET**.

Referenced by **BSP_QSPI_DeInit()**.

void BSP_QSPI_MspInit (QSPI_HandleTypeDef * hqspi, void * Params)

QSPI MSP Initialization This function configures the hardware resources used in this example:

- Peripheral's clock enable
- Peripheral's GPIO Configuration
- NVIC configuration for QSPI interrupt

Return values:

None

Definition at line **511** of file **stm32446e_eval_qspi.c**.

References **QSPI_CLK_ENABLE**, **QSPI_CLK_GPIO_CLK_ENABLE**, **QSPI_CLK_GPIO_PORT**, **QSPI_CLK_PIN**, **QSPI_CS_GPIO_CLK_ENABLE**, **QSPI_CS_GPIO_PORT**, **QSPI_CS_PIN**, **QSPI_D0_GPIO_PORT**, **QSPI_D0_PIN**, **QSPI_D1_GPIO_PORT**, **QSPI_D1_PIN**, **QSPI_D2_GPIO_PORT**, **QSPI_D2_PIN**, **QSPI_D3_GPIO_PORT**, **QSPI_D3_PIN**, **QSPI_Dx_GPIO_CLK_ENABLE**, **QSPI_FORCE_RESET**, and **QSPI_RELEASE_RESET**.

Referenced by **BSP_QSPI_Init()**.

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

Reads an amount of data from the QSPI memory.

Parameters:

pData,:	Pointer to data to be read
ReadAddr,:	Read start address
Size,:	Size of data to read

Return values:

QSPI memory status

Definition at line **195** of file **stm32446e_eval_qspi.c**.

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

uint8_t BSP_QSPI_Write (uint8_t * pData,
uint32_t WriteAddr,
uint32_t Size
)

Writes an amount of data to the QSPI memory.

Parameters:

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

Return values:

QSPI memory status

Definition at line **235** of file **stm32446e_eval_qspi.c**.

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

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL QSPI
Private Functions

STM32446E EVAL QSPI

Functions

static uint8_t	**QSPI_ResetMemory** (QSPI_HandleTypeDef *hqspi) This function reset the QSPI memory.
static uint8_t	**QSPI_EnterFourBytesAddress** (QSPI_HandleTypeDef *hqspi) This function set the QSPI memory in 4-byte address mode.
static uint8_t	**QSPI_DummyCyclesCfg** (QSPI_HandleTypeDef *hqspi) This function configure the dummy cycles on memory side.
static uint8_t	**QSPI_WriteEnable** (QSPI_HandleTypeDef *hqspi) This function send a Write Enable and wait it is effective.
static uint8_t	**QSPI_AutoPollingMemReady** (QSPI_HandleTypeDef *hqspi, uint32_t Timeout) This function read the SR of the memory and wait the EOP.
__weak void	**BSP_QSPI_MspInit** (QSPI_HandleTypeDef *hqspi, void *Params) QSPI MSP Initialization This function configures the hardware resources used in this example:
__weak void	**BSP_QSPI_MspDeInit** (QSPI_HandleTypeDef *hqspi, void *Params) QSPI MSP De-Initialization This function frees the hardware resources used in this example:

Function Documentation

__weak void BSP_QSPI_MspDeInit (QSPI_HandleTypeDef * hqspi,
void * Param
)

QSPI MSP De-Initialization This function frees the hardware resources used in this example:

- Disable the Peripheral's clock
- Revert GPIO and NVIC configuration to their default state

Return values:

None

Definition at line **570** of file **stm32446e_eval_qspi.c**.

References **QSPI_CLK_DISABLE**, **QSPI_D0_GPIO_PORT**, **QSPI_D0_PIN**, **QSPI_D1_GPIO_PORT**, **QSPI_D1_PIN**, **QSPI_D2_GPIO_PORT**, **QSPI_D2_PIN**, **QSPI_D3_GPIO_PORT**, **QSPI_D3_PIN**, **QSPI_FORCE_RESET**, and **QSPI_RELEASE_RESET**.

Referenced by **BSP_QSPI_DeInit()**.

__weak void BSP_QSPI_MspInit (QSPI_HandleTypeDef * hqspi,
void * Params
)

QSPI MSP Initialization This function configures the hardware resources used in this example:

- Peripheral's clock enable
- Peripheral's GPIO Configuration
- NVIC configuration for QSPI interrupt

Return values:

None

Definition at line **511** of file **stm32446e_eval_qspi.c**.

References **QSPI_CLK_ENABLE**, **QSPI_CLK_GPIO_CLK_ENABLE**, **QSPI_CLK_GPIO_PORT**, **QSPI_CLK_PIN**, **QSPI_CS_GPIO_CLK_ENABLE**, **QSPI_CS_GPIO_PORT**, **QSPI_CS_PIN**, **QSPI_D0_GPIO_PORT**, **QSPI_D0_PIN**, **QSPI_D1_GPIO_PORT**, **QSPI_D1_PIN**, **QSPI_D2_GPIO_PORT**, **QSPI_D2_PIN**, **QSPI_D3_GPIO_PORT**, **QSPI_D3_PIN**, **QSPI_Dx_GPIO_CLK_ENABLE**, **QSPI_FORCE_RESET**, and **QSPI_RELEASE_RESET**.

Referenced by **BSP_QSPI_Init()**.

static uint8_t QSPI_AutoPollingMemReady (QSPI_HandleTypeDef *
uint32_t
)

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

Parameters:

hqspi,: QSPI handle

Timeout,: timeout value

Return values:

None

Definition at line **784** of file **stm32446e_eval_qspi.c**.

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

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

static uint8_t QSPI_DummyCyclesCfg (QSPI_HandleTypeDef * hqs

This function configure the dummy cycles on memory side.

Parameters:

hqspi,: QSPI handle

Return values:

None

Definition at line **679** of file **stm32446e_eval_qspi.c**.

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

Referenced by **BSP_QSPI_Init()**.

static uint8_t QSPI_EnterFourBytesAddress (QSPI_HandleTypeDef

This function set the QSPI memory in 4-byte address mode.

Parameters:

hqspi,: QSPI handle

Return values:

None

Definition at line **638** of file **stm32446e_eval_qspi.c**.

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

Referenced by **BSP_QSPI_Init()**.

static uint8_t QSPI_ResetMemory (QSPI_HandleTypeDef * hqspi)

This function reset the QSPI memory.

Parameters:

hqspi,: QSPI handle

Return values:

None

Definition at line **596** of file **stm32446e_eval_qspi.c**.

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

Referenced by **BSP_QSPI_Init()**.

static uint8_t QSPI_WriteEnable (QSPI_HandleTypeDef * hqspi) [s

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

Parameters:

hqspi,: QSPI handle

Return values:

None

Definition at line **738** of file **stm32446e_eval_qspi.c**.

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

Referenced by **BSP_QSPI_Erase_Block()**, **BSP_QSPI_Erase_Chip()**, **BSP_QSPI_Write()**, **QSPI_DummyCyclesCfg()**, and **QSPI_EnterFourBytesAddress()**.

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP
User Manual by  1.7.6.1

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL SD Exported Functions

STM32446E EVAL SD

Functions

uint8_t	**BSP_SD_Init** (void) Initializes the SD card device.
uint8_t	**BSP_SD_DeInit** (void) DeInitializes the SD card device.
uint8_t	**BSP_SD_ITConfig** (void) Configures Interrupt mode for SD detection pin.
void	**BSP_SD_DetectIT** (void)
void	**BSP_SD_DetectCallback** (void)
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.
void	**BSP_SD_MspInit** (SD_HandleTypeDef *hsd, void *Params) Initializes the SD MSP.
void	**BSP_SD_MspDeInit** (SD_HandleTypeDef *hsd, void *Params)

DeInitializes the SD MSP.

Function Documentation

uint8_t BSP_SD_DeInit (void)

DeInitializes the SD card device.

Return values:
SD status

Definition at line **199** of file **stm32446e_eval_sd.c**.

References **BSP_IO_ConfigPin()**, **BSP_SD_MspDeInit()**, **MSD_ERROR**, **MSD_OK**, **uSdHandle**, and **UseExtiModeDetection**.

void BSP_SD_DetectCallback (void)

void BSP_SD_DetectIT (void)

void BSP_SD_DMA_Rx_IRQHandler (void)

Handles SD DMA Rx transfer interrupt request.

Definition at line **556** of file **stm32446e_eval_sd.c**.

References **uSdHandle**.

void BSP_SD_DMA_Tx_IRQHandler (void)

Handles SD DMA Tx transfer interrupt request.

Definition at line **548** of file **stm32446e_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 **381** of file **stm32446e_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

Definition at line **578** of file **stm32446e_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 **569** of file **stm32446e_eval_sd.c**.

References **uSdHandle**.

uint8_t BSP_SD_Init (void)

Initializes the SD card device.

Return values:

SD status

Definition at line **145** of file **stm32446e_eval_sd.c**.

References **BSP_IO_ConfigPin()**, **BSP_IO_Init()**, **BSP_SD_IsDetected()**, **BSP_SD_MspInit()**, **MSD_ERROR**, **MSD_ERROR_SD_NOT_PRESENT**, **MSD_OK**, **SD_PRESENT**, **uSdCardInfo**, and **uSdHandle**.

void BSP_SD_IRQHandler (void)

Handles SD card interrupt request.

Definition at line **540** of file **stm32446e_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 **240** of file **stm32446e_eval_sd.c**.

References **BSP_IO_ConfigPin()**, **BSP_IO_ReadPin()**, **SD_NOT_PRESENT**, **SD_PRESENT**, and **UseExtiModeDetection**.

Referenced by **BSP_SD_Init()**, and **BSP_SD_ITConfig()**.

uint8_t BSP_SD_ITConfig (void)

Configures Interrupt mode for SD detection pin.

Return values:

Returns 0

Definition at line **226** of file **stm32446e_eval_sd.c**.

References **BSP_SD_IsDetected()**, and **UseExtiModeDetection**.

void BSP_SD_MspDeInit (SD_HandleTypeDef * hsd,
void * Params
)

DeInitializes the SD MSP.

Parameters:

hsd,: SD handle

Params,: pointer on additional configuration parameters, can be NULL.

Definition at line **507** of file **stm32446e_eval_sd.c**.

References **SD_DMAx_Rx_IRQn**, **SD_DMAx_Rx_STREAM**,

SD_DMAx_Tx_IRQn, and SD_DMAx_Tx_STREAM.

Referenced by BSP_SD_DeInit().

void BSP_SD_MspInit (SD_HandleTypeDef * hsd,
void * Params
)

Initializes the SD MSP.

Parameters:

hsd,: SD handle

Params,: pointer on additional configuration parameters, can be NULL.

Definition at line 398 of file stm32446e_eval_sd.c.

References __DMAx_TxRx_CLK_ENABLE, BSP_IO_ConfigPin(), BSP_IO_PIN_RESET, BSP_IO_WritePin(), SD_DMAx_Rx_CHANNEL, SD_DMAx_Rx_IRQn, SD_DMAx_Rx_STREAM, SD_DMAx_Tx_CHANNEL, SD_DMAx_Tx_IRQn, and SD_DMAx_Tx_STREAM.

Referenced by BSP_SD_Init().

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 **275** of file **stm32446e_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 **315** of file **stm32446e_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, **SD_DATATIMEOUT**, 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 **295** of file **stm32446e_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 **349** of file **stm32446e_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, **SD_DATATIMEOUT**, and **uSdHandle**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL SD Private Functions

STM32446E EVAL SD

Functions

uint8_t	**BSP_SD_Init** (void) Initializes the SD card device.
uint8_t	**BSP_SD_DeInit** (void) DeInitializes 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.
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.
__weak void	**BSP_SD_MspInit** (SD_HandleTypeDef *hsd, void *Params) Initializes the SD MSP.
__weak void	**BSP_SD_MspDeInit** (SD_HandleTypeDef *hsd, void *Params) DeInitializes the SD MSP.
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

uint8_t BSP_SD_DeInit (void)

DeInitializes the SD card device.

Return values:
SD status

Definition at line **199** of file **stm32446e_eval_sd.c**.

References **BSP_IO_ConfigPin()**, **BSP_SD_MspDeInit()**, **MSD_ERROR**, **MSD_OK**, **uSdHandle**, and **UseExtiModeDetection**.

void BSP_SD_DMA_Rx_IRQHandler (void)

Handles SD DMA Rx transfer interrupt request.

Definition at line **556** of file **stm32446e_eval_sd.c**.

References **uSdHandle**.

void BSP_SD_DMA_Tx_IRQHandler (void)

Handles SD DMA Tx transfer interrupt request.

Definition at line **548** of file **stm32446e_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 **381** of file **stm32446e_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

Definition at line **578** of file **stm32446e_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 **569** of file **stm32446e_eval_sd.c**.

References **uSdHandle**.

uint8_t BSP_SD_Init (void)

Initializes the SD card device.

Return values:

SD status

Definition at line **145** of file **stm32446e_eval_sd.c**.

References **BSP_IO_ConfigPin()**, **BSP_IO_Init()**, **BSP_SD_IsDetected()**, **BSP_SD_MspInit()**, **MSD_ERROR**, **MSD_ERROR_SD_NOT_PRESENT**, **MSD_OK**, **SD_PRESENT**, **uSdCardInfo**, and **uSdHandle**.

void BSP_SD_IRQHandler (void)

Handles SD card interrupt request.

Definition at line **540** of file **stm32446e_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 **240** of file **stm32446e_eval_sd.c**.

References **BSP_IO_ConfigPin()**, **BSP_IO_ReadPin()**, **SD_NOT_PRESENT**, **SD_PRESENT**, and **UseExtiModeDetection**.

Referenced by **BSP_SD_Init()**, and **BSP_SD_ITConfig()**.

uint8_t BSP_SD_ITConfig (void)

Configures Interrupt mode for SD detection pin.

Return values:

Returns 0

Definition at line **226** of file **stm32446e_eval_sd.c**.

References **BSP_SD_IsDetected()**, and **UseExtiModeDetection**.

__weak void BSP_SD_MspDeInit (SD_HandleTypeDef * hsd, void * Params)

DeInitializes the SD MSP.

Parameters:

hsd,: SD handle

Params,: pointer on additional configuration parameters, can be NULL.

Definition at line **507** of file **stm32446e_eval_sd.c**.

References **SD_DMAx_Rx_IRQn**, **SD_DMAx_Rx_STREAM**, **SD_DMAx_Tx_IRQn**, and **SD_DMAx_Tx_STREAM**.

Referenced by **BSP_SD_DeInit()**.

__weak void BSP_SD_MspInit (SD_HandleTypeDef * hsd,
void * Params
)

Initializes the SD MSP.

Parameters:

hsd,:	SD handle
Params,:	pointer on additional configuration parameters, can be NULL.

Definition at line **398** of file **stm32446e_eval_sd.c**.

References **__DMAx_TxRx_CLK_ENABLE**, **BSP_IO_ConfigPin()**, **BSP_IO_PIN_RESET**, **BSP_IO_WritePin()**, **SD_DMAx_Rx_CHANNEL**, **SD_DMAx_Rx_IRQn**, **SD_DMAx_Rx_STREAM**, **SD_DMAx_Tx_CHANNEL**, **SD_DMAx_Tx_IRQn**, and **SD_DMAx_Tx_STREAM**.

Referenced by **BSP_SD_Init()**.

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 **275** of file **stm32446e_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 **315** of file **stm32446e_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, **SD_DATATIMEOUT**, 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 **295** of file **stm32446e_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 **349** of file **stm32446e_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, **SD_DATATIMEOUT**, and **uSdHandle**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL SDRAM Exported Functions

STM32446E EVAL SDRAM

Functions

uint8_t	**BSP_SDRAM_Init** (void) Initializes the SDRAM device.
uint8_t	**BSP_SDRAM_DeInit** (void) DeInitializes the SDRAM device.
void	**BSP_SDRAM_Initialization_sequence** (uint32_t RefreshCount) Programs the SDRAM device.
uint8_t	**BSP_SDRAM_ReadData** (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize) Reads an mount of data from the SDRAM memory in polling mode.
uint8_t	**BSP_SDRAM_ReadData_DMA** (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize) Reads an mount of data from the SDRAM memory in DMA mode.
uint8_t	**BSP_SDRAM_WriteData** (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize) Writes an mount of data to the SDRAM memory in polling mode.
uint8_t	**BSP_SDRAM_WriteData_DMA** (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize) Writes an mount of data to the SDRAM memory in DMA mode.
uint8_t	**BSP_SDRAM_Sendcmd** (FMC_SDRAM_CommandTypeDef *SdramCmd) Sends command to the SDRAM bank.
void	**BSP_SDRAM_DMA_IRQHandler** (void) Handles SDRAM DMA transfer interrupt request.
void	**BSP_SDRAM_MspInit** (SDRAM_HandleTypeDef *hsdram, void *Params) Initializes SDRAM MSP.

void **BSP_SDRAM_MspDeInit** (SDRAM_HandleTypeDef *hsdram, void *Params)

DeInitializes SDRAM MSP.

Function Documentation

uint8_t BSP_SDRAM_DeInit (void)

DeInitializes the SDRAM device.

Return values:
SDRAM status

Definition at line **190** of file **stm32446e_eval_sdram.c**.

References **BSP_SDRAM_MspDeInit()**, **SDRAM_ERROR**, **SDRAM_OK**, and **sdramHandle**.

void BSP_SDRAM_DMA_IRQHandler (void)

Handles SDRAM DMA transfer interrupt request.

Definition at line **366** of file **stm32446e_eval_sdram.c**.

References **sdramHandle**.

uint8_t BSP_SDRAM_Init (void)

Initializes the SDRAM device.

Return values:
SDRAM status

Definition at line **141** of file **stm32446e_eval_sdram.c**.

References **BSP_SDRAM_Initialization_sequence()**, **BSP_SDRAM_MspInit()**, **REFRESH_COUNT**, **SDCLOCK_PERIOD**, **SDRAM_ERROR**, **SDRAM_MEMORY_WIDTH**, **SDRAM_OK**,

sdramHandle, and **Timing**.

void BSP_SDRAM_Initialization_sequence (uint32_t RefreshCount

Programs the SDRAM device.

Parameters:

RefreshCount,: SDRAM refresh counter value

Definition at line **215** of file **stm32446e_eval_sdram.c**.

References **Command**, **SDRAM_MODEREG_BURST_LENGTH_1**, **SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL**, **SDRAM_MODEREG_CAS_LATENCY_2**, **SDRAM_MODEREG_OPERATING_MODE_STANDARD**, **SDRAM_MODEREG_WRITEBURST_MODE_SINGLE**, **SDRAM_TIMEOUT**, and **sdramHandle**.

Referenced by **BSP_SDRAM_Init()**.

void BSP_SDRAM_MspDeInit (SDRAM_HandleTypeDef * hsdram, void * Params)

DeInitializes SDRAM MSP.

Parameters:

hsdram,: SDRAM handle

Params,: pointer on additional configuration parameters, can be NULL.

Definition at line **471** of file **stm32446e_eval_sdram.c**.

References **SDRAM_DMAx_IRQn**, and **SDRAM_DMAx_STREAM**.

Referenced by **BSP_SDRAM_DeInit()**.

void BSP_SDRAM_MspInit (SDRAM_HandleTypeDef * hsdram,
void * Params
)

Initializes SDRAM MSP.

Parameters:

hsdram,: SDRAM handle

Params,: pointer on additional configuration parameters, can be NULL.

Definition at line **376** of file **stm32446e_eval_sdram.c**.

References **__DMAx_CLK_ENABLE**, **SDRAM_DMAx_CHANNEL**, **SDRAM_DMAx_IRQn**, and **SDRAM_DMAx_STREAM**.

Referenced by **BSP_SDRAM_Init()**.

uint8_t BSP_SDRAM_ReadData (uint32_t uwStartAddress,
uint32_t * pData,
uint32_t uwDataSize
)

Reads an mount of data from the SDRAM memory 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:

SDRAM status

Definition at line 277 of file stm32446e_eval_sdram.c.

References SDRAM_ERROR, SDRAM_OK, and sdramHandle.

uint8_t BSP_SDRAM_ReadData_DMA (uint32_t uwStartAddress,
uint32_t * pData,
uint32_t uwDataSize
)

Reads an mount of data from the SDRAM memory 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:

SDRAM status

Definition at line 296 of file stm32446e_eval_sdram.c.

References SDRAM_ERROR, SDRAM_OK, and sdramHandle.

uint8_t BSP_SDRAM_Sendcmd (FMC_SDRAM_CommandTypeDef

Sends command to the SDRAM bank.

Parameters:

SdramCmd,:	Pointer to SDRAM command structure

Return values:

HAL status

Definition at line 351 of file stm32446e_eval_sdram.c.

References **SDRAM_ERROR**, **SDRAM_OK**, **SDRAM_TIMEOUT**, and **sdramHandle**.

uint8_t BSP_SDRAM_WriteData (uint32_t uwStartAddress, uint32_t * pData, uint32_t uwDataSize)

Writes an mount of data to the SDRAM memory 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:

SDRAM status

Definition at line **315** of file **stm32446e_eval_sdram.c**.

References **SDRAM_ERROR**, **SDRAM_OK**, and **sdramHandle**.

uint8_t BSP_SDRAM_WriteData_DMA (uint32_t uwStartAddress, uint32_t * pData, uint32_t uwDataSize)

Writes an mount of data to the SDRAM memory 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:

SDRAM status

Definition at line **334** of file **stm32446e_eval_sdram.c**.

References **SDRAM_ERROR**, **SDRAM_OK**, and **sdramHandle**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL
SDRAM Private Functions

STM32446E EVAL SDRAM

Functions

uint8_t	**BSP_SDRAM_Init** (void) Initializes the SDRAM device.
uint8_t	**BSP_SDRAM_DeInit** (void) DeInitializes the SDRAM device.
void	**BSP_SDRAM_Initialization_sequence** (uint32_t RefreshCount) Programs the SDRAM device.
uint8_t	**BSP_SDRAM_ReadData** (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize) Reads an mount of data from the SDRAM memory in polling mode.
uint8_t	**BSP_SDRAM_ReadData_DMA** (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize) Reads an mount of data from the SDRAM memory in DMA mode.
uint8_t	**BSP_SDRAM_WriteData** (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize) Writes an mount of data to the SDRAM memory in polling mode.
uint8_t	**BSP_SDRAM_WriteData_DMA** (uint32_t uwStartAddress, uint32_t *pData, uint32_t uwDataSize) Writes an mount of data to the SDRAM memory in DMA mode.
uint8_t	**BSP_SDRAM_Sendcmd** (FMC_SDRAM_CommandTypeDef *SdramCmd) Sends command to the SDRAM bank.
void	**BSP_SDRAM_DMA_IRQHandler** (void) Handles SDRAM DMA transfer interrupt request.
__weak void	**BSP_SDRAM_MspInit** (SDRAM_HandleTypeDef *hsdram, void *Params) Initializes SDRAM MSP.

__weak void	**BSP_SDRAM_MspDeInit** (SDRAM_HandleTypeDef *hsdram, void *Params) DeInitializes SDRAM MSP.

Function Documentation

uint8_t BSP_SDRAM_DeInit (void)

DeInitializes the SDRAM device.

Return values:
SDRAM status

Definition at line **190** of file **stm32446e_eval_sdram.c**.

References **BSP_SDRAM_MspDeInit()**, **SDRAM_ERROR**, **SDRAM_OK**, and **sdramHandle**.

void BSP_SDRAM_DMA_IRQHandler (void)

Handles SDRAM DMA transfer interrupt request.

Definition at line **366** of file **stm32446e_eval_sdram.c**.

References **sdramHandle**.

uint8_t BSP_SDRAM_Init (void)

Initializes the SDRAM device.

Return values:
SDRAM status

Definition at line **141** of file **stm32446e_eval_sdram.c**.

References **BSP_SDRAM_Initialization_sequence()**, **BSP_SDRAM_MspInit()**, **REFRESH_COUNT**, **SDCLOCK_PERIOD**, **SDRAM_ERROR**, **SDRAM_MEMORY_WIDTH**, **SDRAM_OK**,

sdramHandle, and **Timing**.

void BSP_SDRAM_Initialization_sequence (uint32_t RefreshCount

Programs the SDRAM device.

Parameters:

RefreshCount,: SDRAM refresh counter value

Definition at line **215** of file **stm32446e_eval_sdram.c**.

References **Command**, **SDRAM_MODEREG_BURST_LENGTH_1**, **SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL**, **SDRAM_MODEREG_CAS_LATENCY_2**, **SDRAM_MODEREG_OPERATING_MODE_STANDARD**, **SDRAM_MODEREG_WRITEBURST_MODE_SINGLE**, **SDRAM_TIMEOUT**, and **sdramHandle**.

Referenced by **BSP_SDRAM_Init()**.

__weak void BSP_SDRAM_MspDeInit (SDRAM_HandleTypeDef * h
void * P
)

DeInitializes SDRAM MSP.

Parameters:

hsdram,: SDRAM handle

Params,: pointer on additional configuration parameters, can be NULL.

Definition at line **471** of file **stm32446e_eval_sdram.c**.

References **SDRAM_DMAx_IRQn**, and **SDRAM_DMAx_STREAM**.

Referenced by **BSP_SDRAM_DeInit()**.

__weak void BSP_SDRAM_MspInit (SDRAM_HandleTypeDef * hsd
void * Para
)

Initializes SDRAM MSP.

Parameters:

hsdram,: SDRAM handle

Params,: pointer on additional configuration parameters, can be NULL.

Definition at line **376** of file **stm32446e_eval_sdram.c**.

References **__DMAx_CLK_ENABLE**, **SDRAM_DMAx_CHANNEL**, **SDRAM_DMAx_IRQn**, and **SDRAM_DMAx_STREAM**.

Referenced by **BSP_SDRAM_Init()**.

uint8_t BSP_SDRAM_ReadData (uint32_t uwStartAddress,
uint32_t * pData,
uint32_t uwDataSize
)

Reads an mount of data from the SDRAM memory 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:

SDRAM status

Definition at line 277 of file stm32446e_eval_sdram.c.

References SDRAM_ERROR, SDRAM_OK, and sdramHandle.

uint8_t BSP_SDRAM_ReadData_DMA (uint32_t uwStartAddress,
uint32_t * pData,
uint32_t uwDataSize
)

Reads an mount of data from the SDRAM memory 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:

SDRAM status

Definition at line 296 of file stm32446e_eval_sdram.c.

References SDRAM_ERROR, SDRAM_OK, and sdramHandle.

uint8_t BSP_SDRAM_Sendcmd (FMC_SDRAM_CommandTypeDef

Sends command to the SDRAM bank.

Parameters:

SdramCmd,:	Pointer to SDRAM command structure

Return values:

HAL status

Definition at line 351 of file stm32446e_eval_sdram.c.

References **SDRAM_ERROR**, **SDRAM_OK**, **SDRAM_TIMEOUT**, and **sdramHandle**.

uint8_t BSP_SDRAM_WriteData (uint32_t uwStartAddress, uint32_t * pData, uint32_t uwDataSize)

Writes an mount of data to the SDRAM memory 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:

SDRAM status

Definition at line **315** of file **stm32446e_eval_sdram.c**.

References **SDRAM_ERROR**, **SDRAM_OK**, and **sdramHandle**.

uint8_t BSP_SDRAM_WriteData_DMA (uint32_t uwStartAddress, uint32_t * pData, uint32_t uwDataSize)

Writes an mount of data to the SDRAM memory 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:

SDRAM status

Definition at line **334** of file **stm32446e_eval_sdram.c**.

References **SDRAM_ERROR**, **SDRAM_OK**, and **sdramHandle**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL TS Exported Functions

STM32446E EVAL TS

Functions

uint8_t	**BSP_TS_Init** (uint16_t xSize, uint16_t ySize) Initializes and configures the touch screen functionalities and configures all necessary hardware resources (GPIOs, clocks..).
uint8_t	**BSP_TS_DeInit** (void) DeInitializes the TouchScreen.
uint8_t	**BSP_TS_GetState** (**TS_StateTypeDef** *TS_State) Returns status and positions of the touch screen.
uint8_t	**BSP_TS_ITEnable** (void) Enables the touch screen interrupts.
uint8_t	**BSP_TS_ITDisable** (void) Disables the touch screen interrupts.
uint8_t	**BSP_TS_ITConfig** (void) Configures and enables the touch screen interrupts.
uint8_t	**BSP_TS_ITGetStatus** (void) Gets the touch screen interrupt status.
void	**BSP_TS_ITClear** (void) Clears all touch screen interrupts.
void	**BSP_TS_FIFOClear** (void) Clears touch screen FIFO containing 128 x,y values.

Function Documentation

uint8_t BSP_TS_DeInit (void)

DeInitializes the TouchScreen.

Return values:

TS state

Definition at line **174** of file **stm32446e_eval_ts.c**.

References **TS_OK**.

void BSP_TS_FIFOClear (void)

Clears touch screen FIFO containing 128 x,y values.

Definition at line **239** of file **stm32446e_eval_ts.c**.

uint8_t BSP_TS_GetState (TS_StateTypeDef * TS_State)

Returns status and positions of the touch screen.

Parameters:

TS_State,: Pointer to touch screen current state structure

Return values:

TS_OK if all initializations are OK. Other value if error.

Definition at line **253** of file **stm32446e_eval_ts.c**.

References **AddressI2C**, **TS_StateTypeDef::TouchDetected**, **ts_driver**, **TS_OK**, **TS_SWAP_X**, **TS_SWAP_XY**, **TS_SWAP_Y**, **tsBundaryX**, **tsBundaryY**, **tsOrientation**, **TS_StateTypeDef::x**, and

TS_StateTypeDef::y.

uint8_t BSP_TS_Init (uint16_t xSize,
uint16_t ySize
)

Initializes and configures the touch screen functionalities and configures all necessary hardware resources (GPIOs, clocks..).

Parameters:

xSize,: Maximum X size of the TS area on LCD

ySize,: Maximum Y size of the TS area on LCD

Return values:

TS_OK if all initializations are OK. Other value if error.

Definition at line **142** of file **stm32446e_eval_ts.c**.

References **AddressI2C**, **BSP_IO_Init()**, **IO_I2C_ADDRESS**, **ts_driver**, **TS_I2C_ADDRESS**, **TS_OK**, **TS_SWAP_NONE**, **tsBundaryX**, **tsBundaryY**, and **tsOrientation**.

void BSP_TS_ITClear (void)

Clears all touch screen interrupts.

Definition at line **220** of file **stm32446e_eval_ts.c**.

References **AddressI2C**, and **ts_driver**.

uint8_t BSP_TS_ITConfig (void)

Configures and enables the touch screen interrupts.

Return values:

TS_OK if all initializations are OK. Other value if error.

Definition at line **206** of file **stm32446e_eval_ts.c**.

References **AddressI2C**, **BSP_IO_Init()**, **ts_driver**, and **TS_OK**.

uint8_t BSP_TS_ITDisable (void)

Disables the touch screen interrupts.

Return values:

TS_OK if all initializations are OK. Other value if error.

Definition at line **195** of file **stm32446e_eval_ts.c**.

References **AddressI2C**, **ts_driver**, and **TS_OK**.

uint8_t BSP_TS_ITEnable (void)

Enables the touch screen interrupts.

Return values:

TS_OK if all initializations are OK. Other value if error.

Definition at line **184** of file **stm32446e_eval_ts.c**.

References **AddressI2C**, **ts_driver**, and **TS_OK**.

uint8_t BSP_TS_ITGetStatus (void)

Gets the touch screen interrupt status.

Return values:

TS_OK if all initializations are OK. Other value if error.

Definition at line **230** of file **stm32446e_eval_ts.c**.

References **AddressI2C**, and **ts_driver**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL TS
Private Functions

STM32446E EVAL TS

Functions

uint8_t	**BSP_TS_Init** (uint16_t xSize, uint16_t ySize) Initializes and configures the touch screen functionalities and configures all necessary hardware resources (GPIOs, clocks..).
uint8_t	**BSP_TS_DeInit** (void) DeInitializes the TouchScreen.
uint8_t	**BSP_TS_ITEnable** (void) Enables the touch screen interrupts.
uint8_t	**BSP_TS_ITDisable** (void) Disables the touch screen interrupts.
uint8_t	**BSP_TS_ITConfig** (void) Configures and enables the touch screen interrupts.
void	**BSP_TS_ITClear** (void) Clears all touch screen interrupts.
uint8_t	**BSP_TS_ITGetStatus** (void) Gets the touch screen interrupt status.
void	**BSP_TS_FIFOClear** (void) Clears touch screen FIFO containing 128 x,y values.
uint8_t	**BSP_TS_GetState** (**TS_StateTypeDef** *TS_State) Returns status and positions of the touch screen.

Function Documentation

uint8_t BSP_TS_DeInit (void)

DeInitializes the TouchScreen.

Return values:

TS state

Definition at line **174** of file **stm32446e_eval_ts.c**.

References **TS_OK**.

void BSP_TS_FIFOClear (void)

Clears touch screen FIFO containing 128 x,y values.

Definition at line **239** of file **stm32446e_eval_ts.c**.

uint8_t BSP_TS_GetState (TS_StateTypeDef * TS_State)

Returns status and positions of the touch screen.

Parameters:

TS_State,: Pointer to touch screen current state structure

Return values:

TS_OK if all initializations are OK. Other value if error.

Definition at line **253** of file **stm32446e_eval_ts.c**.

References **AddressI2C**, **TS_StateTypeDef::TouchDetected**, **ts_driver**, **TS_OK**, **TS_SWAP_X**, **TS_SWAP_XY**, **TS_SWAP_Y**, **tsBundaryX**, **tsBundaryY**, **tsOrientation**, **TS_StateTypeDef::x**, and

TS_StateTypeDef::y.

uint8_t BSP_TS_Init (uint16_t xSize,
uint16_t ySize
)

Initializes and configures the touch screen functionalities and configures all necessary hardware resources (GPIOs, clocks..).

Parameters:

xSize,: Maximum X size of the TS area on LCD

ySize,: Maximum Y size of the TS area on LCD

Return values:

TS_OK if all initializations are OK. Other value if error.

Definition at line **142** of file **stm32446e_eval_ts.c**.

References **AddressI2C**, **BSP_IO_Init()**, **IO_I2C_ADDRESS**, **ts_driver**, **TS_I2C_ADDRESS**, **TS_OK**, **TS_SWAP_NONE**, **tsBundaryX**, **tsBundaryY**, and **tsOrientation**.

void BSP_TS_ITClear (void)

Clears all touch screen interrupts.

Definition at line **220** of file **stm32446e_eval_ts.c**.

References **AddressI2C**, and **ts_driver**.

uint8_t BSP_TS_ITConfig (void)

Configures and enables the touch screen interrupts.

Return values:

TS_OK if all initializations are OK. Other value if error.

Definition at line **206** of file **stm32446e_eval_ts.c**.

References **AddressI2C**, **BSP_IO_Init()**, **ts_driver**, and **TS_OK**.

uint8_t BSP_TS_ITDisable (void)

Disables the touch screen interrupts.

Return values:

TS_OK if all initializations are OK. Other value if error.

Definition at line **195** of file **stm32446e_eval_ts.c**.

References **AddressI2C**, **ts_driver**, and **TS_OK**.

uint8_t BSP_TS_ITEnable (void)

Enables the touch screen interrupts.

Return values:

TS_OK if all initializations are OK. Other value if error.

Definition at line **184** of file **stm32446e_eval_ts.c**.

References **AddressI2C**, **ts_driver**, and **TS_OK**.

uint8_t BSP_TS_ITGetStatus (void)

Gets the touch screen interrupt status.

Return values:

TS_OK if all initializations are OK. Other value if error.

Definition at line **230** of file **stm32446e_eval_ts.c**.

References **AddressI2C**, and **ts_driver**.

STM32446E_EVAL BSP User Manual

STM32446E EVAL LOW LEVEL Private Variables

STM32446E EVAL LOW LEVEL

Variables

const uint32_t	**GPIO_PIN** [**LEDn**]
GPIO_TypeDef *	**BUTTON_PORT** [**BUTTONn**]
const uint16_t	**BUTTON_PIN** [**BUTTONn**]
const uint16_t	**BUTTON_IRQn** [**BUTTONn**]
USART_TypeDef *	**COM_USART** [**COMn**] = {**EVAL_COM1**}
GPIO_TypeDef *	**COM_TX_PORT** [**COMn**] = {**EVAL_COM1_TX_GPIO_PORT**}
GPIO_TypeDef *	**COM_RX_PORT** [**COMn**] = {**EVAL_COM1_RX_GPIO_PORT**}
const uint16_t	**COM_TX_PIN** [**COMn**] = {**EVAL_COM1_TX_PIN**}
const uint16_t	**COM_RX_PIN** [**COMn**] = {**EVAL_COM1_RX_PIN**}
const uint16_t	**COM_TX_AF** [**COMn**] = {**EVAL_COM1_TX_AF**}
const uint16_t	**COM_RX_AF** [**COMn**] = {**EVAL_COM1_RX_AF**}
static FMPI2C_HandleTypeDef	**hEvalI2c**

Variable Documentation

const uint16_t BUTTON_IRQn[BUTTONn]

Initial value:

```
 {WAKEUP_BUTTON_EXTI_IRQn,
                                                TAMPER_BUT
TON_EXTI_IRQn,
                                                KEY_BUTTON
_EXTI_IRQn}
```

Definition at line **132** of file **stm32446e_eval.c**.

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

const uint16_t BUTTON_PIN[BUTTONn]

Initial value:

```
 {WAKEUP_BUTTON_PIN,
                                              TAMPER_BUTT
ON_PIN,
                                              KEY_BUTTON_
PIN}
```

Definition at line **128** of file **stm32446e_eval.c**.

Referenced by **BSP_PB_DeInit()**, **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}
```

Definition at line **124** of file **stm32446e_eval.c**.

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

const uint16_t COM_RX_AF[COMn] = {EVAL_COM1_RX_AF}

Definition at line **148** of file **stm32446e_eval.c**.

Referenced by **BSP_COM_Init()**.

const uint16_t COM_RX_PIN[COMn] = {EVAL_COM1_RX_PIN}

Definition at line **144** of file **stm32446e_eval.c**.

Referenced by **BSP_COM_Init()**.

GPIO_TypeDef* COM_RX_PORT[COMn] = {EVAL_COM1_RX_GPIO_

Definition at line **140** of file **stm32446e_eval.c**.

Referenced by **BSP_COM_Init()**.

const uint16_t COM_TX_AF[COMn] = {EVAL_COM1_TX_AF}

Definition at line **146** of file **stm32446e_eval.c**.

Referenced by **BSP_COM_Init()**.

const uint16_t COM_TX_PIN[COMn] = {EVAL_COM1_TX_PIN}

Definition at line **142** of file **stm32446e_eval.c**.

Referenced by **BSP_COM_Init()**.

GPIO_TypeDef* COM_TX_PORT[COMn] = {EVAL_COM1_TX_GPIO_

Definition at line **138** of file **stm32446e_eval.c**.

Referenced by **BSP_COM_Init()**.

USART_TypeDef* COM_USART[COMn] = {EVAL_COM1}

Definition at line **136** of file **stm32446e_eval.c**.

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

const uint32_t GPIO_PIN[LEDn]

Initial value:

```
{LED1_PIN,
                                        LED3_PIN}
```

Definition at line **119** of file **stm32446e_eval.c**.

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

FMPI2C_HandleTypeDef hEvalI2c `[static]`

Definition at line **150** of file **stm32446e_eval.c**.

Referenced by **I2Cx_Error()**, **I2Cx_Init()**, **I2Cx_IsDeviceReady()**, **I2Cx_ReadMultiple()**, and **I2Cx_WriteMultiple()**.

STM32446E_EVAL BSP User Manual

Enumerations

STM32446E EVAL LOW LEVEL Exported Types

STM32446E EVAL LOW LEVEL

Enumerations

enum	**Led_TypeDef** { **LED1** = 0, **LED_GREEN** = LED1, **LED3** = 1, **LED_RED** = LED3 }
enum	**Button_TypeDef** { **BUTTON_WAKEUP** = 0, **BUTTON_TAMPER** = 1, **BUTTON_KEY** = 2 }
enum	**ButtonMode_TypeDef** { **BUTTON_MODE_GPIO** = 0, **BUTTON_MODE_EXTI** = 1 }
enum	**COM_TypeDef** { **COM1** = 0, **COM2** = 1 }

Enumeration Type Documentation

enum Button_TypeDef

Enumerator:

BUTTON_WAKEUP

BUTTON_TAMPER

BUTTON_KEY

Definition at line **95** of file **stm32446e_eval.h**.

enum ButtonMode_TypeDef

Enumerator:

BUTTON_MODE_GPIO

BUTTON_MODE_EXTI

Definition at line **102** of file **stm32446e_eval.h**.

enum COM_TypeDef

Enumerator:

COM1

COM2

Definition at line **126** of file **stm32446e_eval.h**.

enum Led_TypeDef

Enumerator:

LED1

LED_GREEN
LED3
LED_RED

Definition at line **75** of file **stm32446e_eval.h**.

STM32446E_EVAL BSP User Manual

Defines

STM32446E EVAL LOW LEVEL BUTTON

STM32446E EVAL LOW LEVEL Exported Constants

Defines

#define	**BUTTONn** ((uint8_t)3)
#define	**WAKEUP_BUTTON_PIN** GPIO_PIN_13 Wakeup push-button.
#define	**WAKEUP_BUTTON_GPIO_PORT** GPIOC
#define	**WAKEUP_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_G
#define	**WAKEUP_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_(
#define	**WAKEUP_BUTTON_EXTI_IRQn** EXTI15_10_IRQn
#define	**TAMPER_BUTTON_PIN** GPIO_PIN_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_G
#define	**TAMPER_BUTTON_EXTI_IRQn** EXTI15_10_IRQn
#define	**KEY_BUTTON_PIN** GPIO_PIN_13 Key push-button.
#define	**KEY_BUTTON_GPIO_PORT** GPIOC
#define	**KEY_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOC
#define	**KEY_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_GPIO
#define	**KEY_BUTTON_EXTI_IRQn** EXTI15_10_IRQn
#define	**MFX_IRQOUT_PIN** GPIO_PIN_0 MFX Irq Out Pin.
#define	**MFX_IRQOUT_GPIO_PORT** GPIOA
#define	**MFX_IRQOUT_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOA
#define	**MFX_IRQOUT_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOA
#define	**MFX_IRQOUT_EXTI_IRQn** EXTI0_IRQn
#define	**BUTTONx_GPIO_CLK_ENABLE**(__INDEX__)
#define	**BUTTONx_GPIO_CLK_DISABLE**(__INDEX__)

Define Documentation

#define BUTTONn ((uint8_t)3)

Definition at line **185** of file **stm32446e_eval.h**.

#define BUTTONx_GPIO_CLK_DISABLE (__INDEX__)

Value:

```
(((__INDEX__) == 0) ? WAKEUP_BUTTON_GPIO_CLK_DISA
BLE() :\
                                                (
(__INDEX__) == 1) ? TAMPER_BUTTON_GPIO_CLK_DISABLE
() : KEY_BUTTON_GPIO_CLK_DISABLE())
```

Definition at line **227** of file **stm32446e_eval.h**.

#define BUTTONx_GPIO_CLK_ENABLE (__INDEX__)

Value:

```
do { if((__INDEX__) == 0) {WAKEUP_BUTTON_GPIO_CLK
_ENABLE();} else\

  if((__INDEX__) == 1) {TAMPER_BUTTON_GPIO_CLK_EN
ABLE();} else\

                       {KEY_BUTTON_GPIO_CLK_ENABLE
();   }} while(0)
```

Definition at line **223** of file **stm32446e_eval.h**.

Referenced by **BSP_PB_Init()**.

#define KEY_BUTTON_EXTI_IRQn EXTI15_10_IRQn

Definition at line 212 of file stm32446e_eval.h.

#define KEY_BUTTON_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPI

Definition at line 211 of file stm32446e_eval.h.

#define KEY_BUTTON_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPI

Definition at line 210 of file stm32446e_eval.h.

#define KEY_BUTTON_GPIO_PORT GPIOC

Definition at line 209 of file stm32446e_eval.h.

#define KEY_BUTTON_PIN GPIO_PIN_13

Key push-button.

Definition at line 208 of file stm32446e_eval.h.

#define MFX_IRQOUT_EXTI_IRQn EXTI0_IRQn

Definition at line 221 of file stm32446e_eval.h.

#define MFX_IRQOUT_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPI

Definition at line 220 of file stm32446e_eval.h.

#define MFX_IRQOUT_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIC

Definition at line 219 of file stm32446e_eval.h.

#define MFX_IRQOUT_GPIO_PORT GPIOA

Definition at line 218 of file stm32446e_eval.h.

#define MFX_IRQOUT_PIN GPIO_PIN_0

MFX Irq Out Pin.

Definition at line 217 of file stm32446e_eval.h.

#define TAMPER_BUTTON_EXTI_IRQn EXTI15_10_IRQn

Definition at line 203 of file stm32446e_eval.h.

#define TAMPER_BUTTON_GPIO_CLK_DISABLE ( ) __HAL_RCC_

Definition at line 202 of file stm32446e_eval.h.

#define TAMPER_BUTTON_GPIO_CLK_ENABLE ( ) __HAL_RCC_

Definition at line 201 of file stm32446e_eval.h.

#define TAMPER_BUTTON_GPIO_PORT GPIOC

Definition at line 200 of file stm32446e_eval.h.

#define TAMPER_BUTTON_PIN GPIO_PIN_13

Tamper push-button.

Definition at line 199 of file stm32446e_eval.h.

#define WAKEUP_BUTTON_EXTI_IRQn EXTI15_10_IRQn

Definition at line 194 of file stm32446e_eval.h.

#define WAKEUP_BUTTON_GPIO_CLK_DISABLE ( ) __HAL_RCC

Definition at line 193 of file stm32446e_eval.h.

#define WAKEUP_BUTTON_GPIO_CLK_ENABLE ( ) __HAL_RCC_

Definition at line 192 of file stm32446e_eval.h.

#define WAKEUP_BUTTON_GPIO_PORT GPIOC

Definition at line 191 of file stm32446e_eval.h.

#define WAKEUP_BUTTON_PIN GPIO_PIN_13

Wakeup push-button.

Definition at line 190 of file stm32446e_eval.h.

STM32446E_EVAL BSP User Manual

STM32446E EVAL CAMERA Private Variables

STM32446E-EVAL CAMERA

Variables

CAMERA_DrvTypeDef *	**camera_drv**
static DCMI_HandleTypeDef	**hDcmiEval**
uint32_t	**currentResolution**
static uint32_t	**cameraHwAddress**

Variable Documentation

CAMERA_DrvTypeDef* camera_drv

Definition at line 112 of file stm32446e_eval_camera.c.

Referenced by BSP_CAMERA_BlackWhiteConfig(), BSP_CAMERA_ColorEffectConfig(), BSP_CAMERA_ContrastBrightnessConfig(), and BSP_CAMERA_Init().

uint32_t cameraHwAddress [static]

Definition at line 116 of file stm32446e_eval_camera.c.

Referenced by BSP_CAMERA_BlackWhiteConfig(), BSP_CAMERA_ColorEffectConfig(), BSP_CAMERA_ContrastBrightnessConfig(), and BSP_CAMERA_Init().

uint32_t currentResolution

Definition at line 115 of file stm32446e_eval_camera.c.

Referenced by BSP_CAMERA_ContinuousStart(), BSP_CAMERA_Init(), and BSP_CAMERA_SnapshotStart().

DCMI_HandleTypeDef hDcmiEval [static]

Definition at line 114 of file stm32446e_eval_camera.c.

Referenced by BSP_CAMERA_ContinuousStart(), BSP_CAMERA_DeInit(), BSP_CAMERA_DMA_IRQHandler(),

BSP_CAMERA_Init(), **BSP_CAMERA_IRQHandler()**, **BSP_CAMERA_Resume()**, **BSP_CAMERA_SnapshotStart()**, **BSP_CAMERA_Stop()**, and **BSP_CAMERA_Suspend()**.

STM32446E_EVAL BSP User Manual

Defines | Enumerations

STM32446E EVAL CAMERA Exported Types

STM32446E-EVAL CAMERA

Defines

#define	**RESOLUTION_R160x120**	CAMERA_R160x120 /* QQVGA Resolution */
#define	**RESOLUTION_R320x240**	CAMERA_R320x240 /* QVGA Resolution */
#define	**RESOLUTION_R480x272**	CAMERA_R480x272 /* 480x272 Resolution */
#define	**RESOLUTION_R640x480**	CAMERA_R640x480 /* VGA Resolution */

Enumerations

enum	**Camera_StatusTypeDef** { **CAMERA_OK** = 0x00, **CAMERA_ERROR** = 0x01, **CAMERA_TIMEOUT** = 0x02 } Camera State structures definition. More...

Define Documentation

#define RESOLUTION_R160x120 CAMERA_R160x120 /* QQVGA R

Definition at line 80 of file stm32446e_eval_camera.h.

#define RESOLUTION_R320x240 CAMERA_R320x240 /* QVGA Re

Definition at line 81 of file stm32446e_eval_camera.h.

#define RESOLUTION_R480x272 CAMERA_R480x272 /* 480x272 F

Definition at line 82 of file stm32446e_eval_camera.h.

#define RESOLUTION_R640x480 CAMERA_R640x480 /* VGA Resc

Definition at line 83 of file stm32446e_eval_camera.h.

Enumeration Type Documentation

enum Camera_StatusTypeDef

Camera State structures definition.

Enumerator:

CAMERA_OK

CAMERA_ERROR

CAMERA_TIMEOUT

Definition at line **73** of file **stm32446e_eval_camera.h**.

STM32446E_EVAL BSP User Manual

Defines

STM32446E EVAL
Audio Slot TDM mode

STM32446E EVAL AUDIO Exported Constants

In W8994 codec the Audio frame contains 4 slots : TDM Mode TDM format : +------------------|------------------|-------------------|------------------+ | CODEC_SLOT0 Left | CODEC_SLOT1 Left | CODEC_SLOT0 Right | CODEC_SLOT1 Right | +--+. More...

Defines

#define	**CODEC_AUDIOFRAME_SLOT_0123** SAI_SLOTACTIVE_0 | SAI_SLOTACTIVE_1 | SAI_SLOTACTIVE_2 | SAI_SLOTACTIVE_3
#define	**CODEC_AUDIOFRAME_SLOT_02** SAI_SLOTACTIVE_0 | SAI_SLOTACTIVE_2
#define	**CODEC_AUDIOFRAME_SLOT_13** SAI_SLOTACTIVE_1 | SAI_SLOTACTIVE_3

Detailed Description

In W8994 codec the Audio frame contains 4 slots : TDM Mode TDM format : +-----------------|-----------------|-----------------|-----------------+ | CODEC_SLOT0 Left | CODEC_SLOT1 Left | CODEC_SLOT0 Right | CODEC_SLOT1 Right | +---+.

Define Documentation

#define CODEC_AUDIOFRAME_SLOT_0123 SAI_SLOTACTIVE_0 |

Definition at line **88** of file **stm32446e_eval_audio.h**.

Referenced by **SAIx_Init()**.

#define CODEC_AUDIOFRAME_SLOT_02 SAI_SLOTACTIVE_0 | S

Definition at line **90** of file **stm32446e_eval_audio.h**.

#define CODEC_AUDIOFRAME_SLOT_13 SAI_SLOTACTIVE_1 | S

Definition at line **92** of file **stm32446e_eval_audio.h**.

STM32446E_EVAL BSP User Manual

STM32446E EVAL SDRAM Private Variables

STM32446E EVAL SDRAM

Variables

static SDRAM_HandleTypeDef	**sdramHandle**
static FMC_SDRAM_TimingTypeDef	**Timing**
static FMC_SDRAM_CommandTypeDef	**Command**

Variable Documentation

FMC_SDRAM_CommandTypeDef Command [static]

Definition at line 121 of file stm32446e_eval_sdram.c.

Referenced by BSP_SDRAM_Initialization_sequence().

SDRAM_HandleTypeDef sdramHandle [static]

Definition at line 119 of file stm32446e_eval_sdram.c.

Referenced by BSP_SDRAM_DeInit(), BSP_SDRAM_DMA_IRQHandler(), BSP_SDRAM_Init(), BSP_SDRAM_Initialization_sequence(), BSP_SDRAM_ReadData(), BSP_SDRAM_ReadData_DMA(), BSP_SDRAM_Sendcmd(), BSP_SDRAM_WriteData(), and BSP_SDRAM_WriteData_DMA().

FMC_SDRAM_TimingTypeDef Timing [static]

Definition at line 120 of file stm32446e_eval_sdram.c.

Referenced by BSP_SDRAM_Init().

STM32446E_EVAL BSP User Manual

Defines

STM32446E EVAL AUDIO Exported Macros

STM32446E 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_MAX_

Definition at line **237** of file **stm32446e_eval_audio.h**.

Referenced by **BSP_AUDIO_OUT_Play()**.

STM32446E_EVAL BSP User Manual

STM32446E EVAL LCD
Private Variables

STM32446E EVAL LCD

Variables

LCD_DrawPropTypeDef	**DrawProp**
static LCD_DrvTypeDef *	**LcdDrv**

Variable Documentation

LCD_DrawPropTypeDef DrawProp

Definition at line **112** of file **stm32446e_eval_lcd.c**.

LCD_DrvTypeDef* LcdDrv [static]

Definition at line **113** of file **stm32446e_eval_lcd.c**.

Referenced by **BSP_LCD_DisplayOff()**, **BSP_LCD_DisplayOn()**, **BSP_LCD_DrawBitmap()**, **BSP_LCD_DrawHLine()**, **BSP_LCD_DrawPixel()**, **BSP_LCD_DrawRGBImage()**, **BSP_LCD_DrawVLine()**, **BSP_LCD_GetXSize()**, **BSP_LCD_GetYSize()**, **BSP_LCD_Init()**, **BSP_LCD_ReadPixel()**, and **SetDisplayWindow()**.

STM32446E_EVAL BSP User Manual

Defines

STM32446E EVAL
EEPROM Exported Constants

STM32446E EVAL EEPROM

Defines

#define	**EEPROM_PAGESIZE** ((uint8_t)4)
#define	**EEPROM_MAX_SIZE** ((uint16_t)0x2000) /* 64Kbit */
#define	**EEPROM_MAX_TRIALS** ((uint32_t)3000)
#define	**EEPROM_OK** ((uint32_t)0)
#define	**EEPROM_FAIL** ((uint32_t)1)
#define	**EEPROM_TIMEOUT** ((uint32_t)2)

Define Documentation

#define EEPROM_FAIL ((uint32_t)1)

Definition at line 82 of file stm32446e_eval_eeprom.h.

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

#define EEPROM_MAX_SIZE ((uint16_t)0x2000) /* 64Kbit */

Definition at line 75 of file stm32446e_eval_eeprom.h.

#define EEPROM_MAX_TRIALS ((uint32_t)3000)

Definition at line 79 of file stm32446e_eval_eeprom.h.

Referenced by BSP_EEPROM_Init(), and BSP_EEPROM_WaitEepromStandbyState().

#define EEPROM_OK ((uint32_t)0)

Definition at line 81 of file stm32446e_eval_eeprom.h.

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

#define EEPROM_PAGESIZE ((uint8_t)4)

Definition at line 74 of file stm32446e_eval_eeprom.h.

Referenced by **BSP_EEPROM_WriteBuffer()**.

#define EEPROM_TIMEOUT ((uint32_t)2)

Definition at line **83** of file **stm32446e_eval_eeprom.h**.

Referenced by **BSP_EEPROM_WaitEepromStandbyState()**.

STM32446E_EVAL BSP User Manual

STM32446E EVAL EEPROM Private Variables

STM32446E EVAL EEPROM

Variables

__IO uint16_t	**EEPROMAddress** = 0
__IO uint16_t	**EEPROMDataRead**
__IO uint8_t	**EEPROMDataWrite**

Variable Documentation

__IO uint16_t EEPROMAddress = 0

Definition at line **128** of file **stm32446e_eval_eeprom.c**.

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

__IO uint16_t EEPROMDataRead

Definition at line **129** of file **stm32446e_eval_eeprom.c**.

Referenced by **BSP_EEPROM_ReadBuffer()**.

__IO uint8_t EEPROMDataWrite

Definition at line **130** of file **stm32446e_eval_eeprom.c**.

Referenced by **BSP_EEPROM_WritePage()**.

STM32446E_EVAL BSP User Manual

Functions

STM32446E EVAL
AUDIO Private Function Prototypes

STM32446E EVAL AUDIO

Functions

static void	**SAIx_DeInit** (void) Deinitializes the Audio Codec audio interface (SAI).
static void	**I2Sx_DeInit** (void) Deinitializes the Audio Codec audio interface (I2S).

Function Documentation

static void I2Sx_DeInit (void) [static]

Deinitializes the Audio Codec audio interface (I2S).

Definition at line **1162** of file **stm32446e_eval_audio.c**.

References **AUDIO_I2Sx**, and **haudio_in_i2s**.

Referenced by **BSP_AUDIO_IN_DeInit()**, and **BSP_AUDIO_IN_Init()**.

static void SAIx_DeInit (void) [static]

Deinitializes the Audio Codec audio interface (SAI).

Definition at line **753** of file **stm32446e_eval_audio.c**.

References **AUDIO_SAIx**, and **haudio_out_sai**.

Referenced by **BSP_AUDIO_OUT_DeInit()**, and **BSP_AUDIO_OUT_Init()**.

STM32446E_EVAL BSP User Manual

Defines

STM32446E EVAL IO Exported Constants

STM32446E EVAL IO

Defines

#define	**IO_PIN_0** ((uint32_t)0x0001)
#define	**IO_PIN_1** ((uint32_t)0x0002)
#define	**IO_PIN_2** ((uint32_t)0x0004)
#define	**IO_PIN_3** ((uint32_t)0x0008)
#define	**IO_PIN_4** ((uint32_t)0x0010)
#define	**IO_PIN_5** ((uint32_t)0x0020)
#define	**IO_PIN_6** ((uint32_t)0x0040)
#define	**IO_PIN_7** ((uint32_t)0x0080)
#define	**IO_PIN_8** ((uint32_t)0x0100)
#define	**IO_PIN_9** ((uint32_t)0x0200)
#define	**IO_PIN_10** ((uint32_t)0x0400)
#define	**IO_PIN_11** ((uint32_t)0x0800)
#define	**IO_PIN_12** ((uint32_t)0x1000)
#define	**IO_PIN_13** ((uint32_t)0x2000)
#define	**IO_PIN_14** ((uint32_t)0x4000)
#define	**IO_PIN_15** ((uint32_t)0x8000)
#define	**IO_PIN_16** ((uint32_t)0x010000)
#define	**IO_PIN_17** ((uint32_t)0x020000)
#define	**IO_PIN_18** ((uint32_t)0x040000)
#define	**IO_PIN_19** ((uint32_t)0x080000)
#define	**IO_PIN_20** ((uint32_t)0x100000)
#define	**IO_PIN_21** ((uint32_t)0x200000)
#define	**IO_PIN_22** ((uint32_t)0x400000)
#define	**IO_PIN_23** ((uint32_t)0x800000)
#define	**IO_PIN_ALL** ((uint32_t)0xFFFFFF)

Define Documentation

#define IO_PIN_0 ((uint32_t)0x0001)

Definition at line 88 of file stm32446e_eval_io.h.

#define IO_PIN_1 ((uint32_t)0x0002)

Definition at line 89 of file stm32446e_eval_io.h.

#define IO_PIN_10 ((uint32_t)0x0400)

Definition at line 98 of file stm32446e_eval_io.h.

#define IO_PIN_11 ((uint32_t)0x0800)

Definition at line 99 of file stm32446e_eval_io.h.

#define IO_PIN_12 ((uint32_t)0x1000)

Definition at line 100 of file stm32446e_eval_io.h.

#define IO_PIN_13 ((uint32_t)0x2000)

Definition at line 101 of file stm32446e_eval_io.h.

#define IO_PIN_14 ((uint32_t)0x4000)

Definition at line 102 of file stm32446e_eval_io.h.

#define IO_PIN_15 ((uint32_t)0x8000)

Definition at line 103 of file stm32446e_eval_io.h.

#define IO_PIN_16 ((uint32_t)0x010000)

Definition at line 104 of file stm32446e_eval_io.h.

#define IO_PIN_17 ((uint32_t)0x020000)

Definition at line 105 of file stm32446e_eval_io.h.

#define IO_PIN_18 ((uint32_t)0x040000)

Definition at line 106 of file stm32446e_eval_io.h.

#define IO_PIN_19 ((uint32_t)0x080000)

Definition at line 107 of file stm32446e_eval_io.h.

#define IO_PIN_2 ((uint32_t)0x0004)

Definition at line 90 of file stm32446e_eval_io.h.

#define IO_PIN_20 ((uint32_t)0x100000)

Definition at line 108 of file stm32446e_eval_io.h.

#define IO_PIN_21 ((uint32_t)0x200000)

Definition at line **109** of file **stm32446e_eval_io.h**.

#define IO_PIN_22 ((uint32_t)0x400000)

Definition at line **110** of file **stm32446e_eval_io.h**.

#define IO_PIN_23 ((uint32_t)0x800000)

Definition at line **111** of file **stm32446e_eval_io.h**.

#define IO_PIN_3 ((uint32_t)0x0008)

Definition at line **91** of file **stm32446e_eval_io.h**.

#define IO_PIN_4 ((uint32_t)0x0010)

Definition at line **92** of file **stm32446e_eval_io.h**.

#define IO_PIN_5 ((uint32_t)0x0020)

Definition at line **93** of file **stm32446e_eval_io.h**.

#define IO_PIN_6 ((uint32_t)0x0040)

Definition at line **94** of file **stm32446e_eval_io.h**.

#define IO_PIN_7 ((uint32_t)0x0080)

Definition at line **95** of file **stm32446e_eval_io.h**.

#define IO_PIN_8 ((uint32_t)0x0100)

Definition at line **96** of file **stm32446e_eval_io.h**.

#define IO_PIN_9 ((uint32_t)0x0200)

Definition at line **97** of file **stm32446e_eval_io.h**.

#define IO_PIN_ALL ((uint32_t)0xFFFFFF)

Definition at line **112** of file **stm32446e_eval_io.h**.

Referenced by **BSP_IO_Init()**.

STM32446E_EVAL BSP User Manual

STM32446E EVAL IO
Private Variables

STM32446E EVAL IO

Variables

static IO_DrvTypeDef *	**IoDrv** = NULL
static uint8_t	**mfxstm32l152Identifier**

Variable Documentation

IO_DrvTypeDef* IoDrv = NULL [static]

Definition at line 113 of file stm32446e_eval_io.c.

Referenced by BSP_IO_ConfigPin(), BSP_IO_DeInit(), BSP_IO_Init(), BSP_IO_ITClear(), BSP_IO_ITGetStatus(), BSP_IO_ReadPin(), BSP_IO_TogglePin(), and BSP_IO_WritePin().

uint8_t mfxstm32l152Identifier [static]

Definition at line 114 of file stm32446e_eval_io.c.

Referenced by BSP_IO_ConfigIrqOutPin(), and BSP_IO_Init().

STM32446E_EVAL BSP User Manual

Defines

STM32446E EVAL LOW LEVEL LED

STM32446E EVAL LOW LEVEL Exported Constants

Define for STM32446E_EVAL board. More...

Defines

#define **LEDn** ((uint8_t)2)
#define **LEDx_GPIO_PORT** GPIOB
#define **LEDx_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_CLK_E
#define **LEDx_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_CLK_D
#define **LED1_PIN** GPIO_PIN_11
#define **LED3_PIN** GPIO_PIN_4

Detailed Description

Define for STM32446E_EVAL board.

Define Documentation

#define LED1_PIN GPIO_PIN_11

Definition at line 163 of file stm32446e_eval.h.

#define LED3_PIN GPIO_PIN_4

Definition at line 164 of file stm32446e_eval.h.

#define LEDn ((uint8_t)2)

Definition at line 157 of file stm32446e_eval.h.

#define LEDx_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOB_CLK

Definition at line 162 of file stm32446e_eval.h.

#define LEDx_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOB_CLK_

Definition at line 161 of file stm32446e_eval.h.

Referenced by BSP_LED_Init().

#define LEDx_GPIO_PORT GPIOB

Definition at line 160 of file stm32446e_eval.h.

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

Generated on Fri Jan 15 2016 10:06:22 for STM32446E_EVAL BSP
User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

Defines

STM32446E EVAL LCD
Private Defines

STM32446E EVAL LCD

Defines

#define	**POLY_X**(Z) ((int32_t)((Points + Z)->X))
#define	**POLY_Y**(Z) ((int32_t)((Points + Z)->Y))

Define Documentation

#define POLY_X (Z) ((int32_t)((Points + Z)->X))

Definition at line **95** of file **stm32446e_eval_lcd.c**.

Referenced by **BSP_LCD_FillPolygon()**.

#define POLY_Y (Z) ((int32_t)((Points + Z)->Y))

Definition at line **96** of file **stm32446e_eval_lcd.c**.

Referenced by **BSP_LCD_FillPolygon()**.

STM32446E_EVAL BSP User Manual

Defines

STM32446E EVAL QSPI Exported Constants

STM32446E EVAL QSPI

Defines

#define **QSPI_OK** ((uint8_t)0x00)
#define **QSPI_ERROR** ((uint8_t)0x01)
#define **QSPI_BUSY** ((uint8_t)0x02)
#define **QSPI_NOT_SUPPORTED** ((uint8_t)0x04)
#define **QSPI_SUSPENDED** ((uint8_t)0x08)
#define **QSPI_CLK_ENABLE**() __HAL_RCC_QSPI_CLK_ENABLE()
#define **QSPI_CLK_DISABLE**() __HAL_RCC_QSPI_CLK_DISABLE(
#define **QSPI_CS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOG_CL
#define **QSPI_CS_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOG_CL
#define **QSPI_CLK_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOD_C
#define **QSPI_CLK_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOD_C
#define **QSPI_Dx_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOF_CLK
#define **QSPI_Dx_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOF_CL
#define **QSPI_FORCE_RESET**() __HAL_RCC_QSPI_FORCE_RESE
#define **QSPI_RELEASE_RESET**() __HAL_RCC_QSPI_RELEASE_
#define **QSPI_CS_PIN** GPIO_PIN_6
#define **QSPI_CS_GPIO_PORT** GPIOG
#define **QSPI_CLK_PIN** GPIO_PIN_3
#define **QSPI_CLK_GPIO_PORT** GPIOD
#define **QSPI_D0_PIN** GPIO_PIN_8
#define **QSPI_D0_GPIO_PORT** GPIOF
#define **QSPI_D1_PIN** GPIO_PIN_9
#define **QSPI_D1_GPIO_PORT** GPIOF
#define **QSPI_D2_PIN** GPIO_PIN_7
#define **QSPI_D2_GPIO_PORT** GPIOF
#define **QSPI_D3_PIN** GPIO_PIN_6
#define **QSPI_D3_GPIO_PORT** GPIOF

Define Documentation

#define QSPI_BUSY ((uint8_t)0x02)

Definition at line 71 of file stm32446e_eval_qspi.h.

Referenced by BSP_QSPI_GetStatus().

#define QSPI_CLK_DISABLE ( ) __HAL_RCC_QSPI_CLK_DISABL

Definition at line 78 of file stm32446e_eval_qspi.h.

Referenced by BSP_QSPI_MspDeInit().

#define QSPI_CLK_ENABLE ( ) __HAL_RCC_QSPI_CLK_ENABLE

Definition at line 77 of file stm32446e_eval_qspi.h.

Referenced by BSP_QSPI_MspInit().

#define QSPI_CLK_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOD_

Definition at line 83 of file stm32446e_eval_qspi.h.

#define QSPI_CLK_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOD_

Definition at line 82 of file stm32446e_eval_qspi.h.

Referenced by BSP_QSPI_MspInit().

#define QSPI_CLK_GPIO_PORT GPIOD

Definition at line **99** of file **stm32446e_eval_qspi.h**.

Referenced by **BSP_QSPI_MspInit()**.

#define QSPI_CLK_PIN GPIO_PIN_3

Definition at line **98** of file **stm32446e_eval_qspi.h**.

Referenced by **BSP_QSPI_MspInit()**.

#define QSPI_CS_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOG_(

Definition at line **80** of file **stm32446e_eval_qspi.h**.

#define QSPI_CS_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOG_C

Definition at line **79** of file **stm32446e_eval_qspi.h**.

Referenced by **BSP_QSPI_MspInit()**.

#define QSPI_CS_GPIO_PORT GPIOG

Definition at line **96** of file **stm32446e_eval_qspi.h**.

Referenced by **BSP_QSPI_MspInit()**.

#define QSPI_CS_PIN GPIO_PIN_6

Definition at line **95** of file **stm32446e_eval_qspi.h**.

Referenced by **BSP_QSPI_MspInit()**.

#define QSPI_D0_GPIO_PORT GPIOF

Definition at line 105 of file stm32446e_eval_qspi.h.

Referenced by BSP_QSPI_MspDeInit(), and BSP_QSPI_MspInit().

#define QSPI_D0_PIN GPIO_PIN_8

Definition at line 104 of file stm32446e_eval_qspi.h.

Referenced by BSP_QSPI_MspDeInit(), and BSP_QSPI_MspInit().

#define QSPI_D1_GPIO_PORT GPIOF

Definition at line 107 of file stm32446e_eval_qspi.h.

Referenced by BSP_QSPI_MspDeInit(), and BSP_QSPI_MspInit().

#define QSPI_D1_PIN GPIO_PIN_9

Definition at line 106 of file stm32446e_eval_qspi.h.

Referenced by BSP_QSPI_MspDeInit(), and BSP_QSPI_MspInit().

#define QSPI_D2_GPIO_PORT GPIOF

Definition at line 109 of file stm32446e_eval_qspi.h.

Referenced by BSP_QSPI_MspDeInit(), and BSP_QSPI_MspInit().

#define QSPI_D2_PIN GPIO_PIN_7

Definition at line 108 of file stm32446e_eval_qspi.h.

Referenced by BSP_QSPI_MspDeInit(), and BSP_QSPI_MspInit().

#define QSPI_D3_GPIO_PORT GPIOF

Definition at line 111 of file stm32446e_eval_qspi.h.

Referenced by BSP_QSPI_MspDeInit(), and BSP_QSPI_MspInit().

#define QSPI_D3_PIN GPIO_PIN_6

Definition at line 110 of file stm32446e_eval_qspi.h.

Referenced by BSP_QSPI_MspDeInit(), and BSP_QSPI_MspInit().

#define QSPI_Dx_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOF_C

Definition at line 89 of file stm32446e_eval_qspi.h.

#define QSPI_Dx_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOF_CL

Definition at line 88 of file stm32446e_eval_qspi.h.

Referenced by BSP_QSPI_MspInit().

#define QSPI_ERROR ((uint8_t)0x01)

Definition at line 70 of file stm32446e_eval_qspi.h.

Referenced by BSP_QSPI_DeInit(), BSP_QSPI_Erase_Block(), BSP_QSPI_Erase_Chip(), BSP_QSPI_GetStatus(),

BSP_QSPI_Init(), **BSP_QSPI_MemoryMappedMode()**, **BSP_QSPI_Read()**, **BSP_QSPI_Write()**, **QSPI_AutoPollingMemReady()**, **QSPI_DummyCyclesCfg()**, **QSPI_EnterFourBytesAddress()**, **QSPI_ResetMemory()**, and **QSPI_WriteEnable()**.

#define QSPI_FORCE_RESET ( ) __HAL_RCC_QSPI_FORCE_RES

Definition at line **91** of file **stm32446e_eval_qspi.h**.

Referenced by **BSP_QSPI_MspDeInit()**, and **BSP_QSPI_MspInit()**.

#define QSPI_NOT_SUPPORTED ((uint8_t)0x04)

Definition at line **72** of file **stm32446e_eval_qspi.h**.

Referenced by **BSP_QSPI_Init()**.

#define QSPI_OK ((uint8_t)0x00)

Definition at line **69** of file **stm32446e_eval_qspi.h**.

Referenced by **BSP_QSPI_DeInit()**, **BSP_QSPI_Erase_Block()**, **BSP_QSPI_Erase_Chip()**, **BSP_QSPI_GetInfo()**, **BSP_QSPI_GetStatus()**, **BSP_QSPI_Init()**, **BSP_QSPI_MemoryMappedMode()**, **BSP_QSPI_Read()**, **BSP_QSPI_Write()**, **QSPI_AutoPollingMemReady()**, **QSPI_DummyCyclesCfg()**, **QSPI_EnterFourBytesAddress()**, **QSPI_ResetMemory()**, and **QSPI_WriteEnable()**.

#define QSPI_RELEASE_RESET ( ) __HAL_RCC_QSPI_RELEASE

Definition at line **92** of file **stm32446e_eval_qspi.h**.

Referenced by **BSP_QSPI_MspDeInit()**, and **BSP_QSPI_MspInit()**.

#define QSPI_SUSPENDED ((uint8_t)0x08)

Definition at line **73** of file **stm32446e_eval_qspi.h**.

Referenced by **BSP_QSPI_GetStatus()**.

STM32446E_EVAL BSP User Manual

STM32446E EVAL QSPI
Private Variables

STM32446E EVAL QSPI

Variables

QSPI_HandleTypeDef	**QSPIHandle**

Variable Documentation

QSPI_HandleTypeDef QSPIHandle

Definition at line **88** of file **stm32446e_eval_qspi.c**.

Referenced by **BSP_QSPI_DeInit()**, **BSP_QSPI_Erase_Block()**, **BSP_QSPI_Erase_Chip()**, **BSP_QSPI_GetStatus()**, **BSP_QSPI_Init()**, **BSP_QSPI_MemoryMappedMode()**, **BSP_QSPI_Read()**, **BSP_QSPI_Write()**, **QSPI_AutoPollingMemReady()**, **QSPI_DummyCyclesCfg()**, **QSPI_EnterFourBytesAddress()**, **QSPI_ResetMemory()**, and **QSPI_WriteEnable()**.

STM32446E_EVAL BSP User Manual

Defines

STM32446E EVAL SD Exported Types

STM32446E EVAL SD

Defines

#define **SD_CardInfo** HAL_SD_CardInfoTypedef
SD Card information structure.

Define Documentation

#define SD_CardInfo HAL_SD_CardInfoTypedef

SD Card information structure.

Definition at line **70** of file **stm32446e_eval_sd.h**.

STM32446E_EVAL BSP User Manual

Defines

STM32446E EVAL SDRAM Exported Types

STM32446E EVAL SDRAM

Defines

#define	**SDRAM_OK** ((uint8_t)0x00) SDRAM status structure definition.
#define	**SDRAM_ERROR** ((uint8_t)0x01)

Define Documentation

#define SDRAM_ERROR ((uint8_t)0x01)

Definition at line 70 of file stm32446e_eval_sdram.h.

Referenced by BSP_SDRAM_DeInit(), BSP_SDRAM_Init(), BSP_SDRAM_ReadData(), BSP_SDRAM_ReadData_DMA(), BSP_SDRAM_Sendcmd(), BSP_SDRAM_WriteData(), and BSP_SDRAM_WriteData_DMA().

#define SDRAM_OK ((uint8_t)0x00)

SDRAM status structure definition.

Definition at line 69 of file stm32446e_eval_sdram.h.

Referenced by BSP_SDRAM_DeInit(), BSP_SDRAM_Init(), BSP_SDRAM_ReadData(), BSP_SDRAM_ReadData_DMA(), BSP_SDRAM_Sendcmd(), BSP_SDRAM_WriteData(), and BSP_SDRAM_WriteData_DMA().

STM32446E_EVAL BSP User Manual

Defines | Enumerations

STM32446E EVAL TS Exported Constants

STM32446E EVAL TS

Defines

#define **TS_SWAP_NONE** ((uint8_t)0x00)
#define **TS_SWAP_X** ((uint8_t)0x01)
#define **TS_SWAP_Y** ((uint8_t)0x02)
#define **TS_SWAP_XY** ((uint8_t)0x04)

Enumerations

enum	**TS_StatusTypeDef** { **TS_OK** = 0x00, **TS_ERROR** = 0x01, **TS_TIMEOUT** = 0x02 }

Define Documentation

#define TS_SWAP_NONE ((uint8_t)0x00)

Definition at line **81** of file **stm32446e_eval_ts.h**.

Referenced by **BSP_TS_Init()**.

#define TS_SWAP_X ((uint8_t)0x01)

Definition at line **82** of file **stm32446e_eval_ts.h**.

Referenced by **BSP_TS_GetState()**.

#define TS_SWAP_XY ((uint8_t)0x04)

Definition at line **84** of file **stm32446e_eval_ts.h**.

Referenced by **BSP_TS_GetState()**.

#define TS_SWAP_Y ((uint8_t)0x02)

Definition at line **83** of file **stm32446e_eval_ts.h**.

Referenced by **BSP_TS_GetState()**.

Enumeration Type Documentation

enum TS_StatusTypeDef

Enumerator:

TS_OK

TS_ERROR

TS_TIMEOUT

Definition at line **86** of file **stm32446e_eval_ts.h**.

STM32446E_EVAL BSP User Manual

STM32446E EVAL SD
Private Variables

STM32446E EVAL SD

Variables

static SD_HandleTypeDef	**uSdHandle**
static **SD_CardInfo**	**uSdCardInfo**
static uint8_t	**UseExtiModeDetection** = 0

Variable Documentation

SD_CardInfo uSdCardInfo [static]

Definition at line 123 of file stm32446e_eval_sd.c.

Referenced by BSP_SD_Init().

SD_HandleTypeDef uSdHandle [static]

Definition at line 122 of file stm32446e_eval_sd.c.

Referenced by BSP_SD_DeInit(), 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().

uint8_t UseExtiModeDetection = 0 [static]

Definition at line 124 of file stm32446e_eval_sd.c.

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

STM32446E_EVAL BSP User Manual

Drivers Directory Reference

Directories

directory	**BSP**

STM32446E_EVAL BSP User Manual

BSP Directory Reference

Directories

directory	**STM32446E_EVAL**

STM32446E_EVAL BSP User Manual

STM32446E_EVAL Directory Reference

Files

file **stm32446e_eval.c** [code]

This file provides a set of firmware functions to manage LEDs, push-buttons and COM ports available on STM32446E-EVAL evaluation board(MB1045) RevB from STMicroelectronics.

file **stm32446e_eval.h** [code]

This file contains definitions for STM32446E_EVAL's LEDs, push-buttons and COM ports hardware resources.

file **stm32446e_eval_audio.c** [code]

This file provides the Audio driver for the STM32446E-EVAL evaluation board.

file **stm32446e_eval_audio.h** [code]

This file contains the common defines and functions prototypes for the **stm32446e_eval_audio.c** driver.

file **stm32446e_eval_camera.c** [code]

This file includes the driver for Camera modules mounted on STM32446E-EVAL evaluation board.

file **stm32446e_eval_camera.h** [code]

This file contains the common defines and functions prototypes for the **stm32446e_eval_camera.c** driver.

file **stm32446e_eval_eeprom.c** [code]

This file provides a set of functions needed to manage an I2C M24LR64 EEPROM memory. To be able to use this driver, the switch EE_M24LR64 must be defined in your toolchain compiler preprocessor.

file **stm32446e_eval_eeprom.h** [code]

This file contains all the functions prototypes for the **stm32446e_eval_eeprom.c** firmware driver.

file **stm32446e_eval_io.c** [code]

This file provides a set of functions needed to manage the IO pins on STM32446E-EVAL evaluation board.

file **stm32446e_eval_io.h** [code]

This file contains the common defines and functions prototypes for the **stm32446e_eval_io.c** driver.

file **stm32446e_eval_lcd.c** [code]

This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM32446E-EVAL evaluation board.

file **stm32446e_eval_lcd.h** [code]

This file contains the common defines and functions prototypes for the **stm32446e_eval_lcd.c** driver.

file **stm32446e_eval_qspi.c** [code]

This file includes a standard driver for the N25Q256A QSPI memory mounted on STM32446E-EVAL board.

file **stm32446e_eval_qspi.h** [code]

This file contains the common defines and functions prototypes for the **stm32446e_eval_qspi.c** driver.

file **stm32446e_eval_sd.c** [code]

This file includes the uSD card driver mounted on STM32446E-EVAL evaluation board.

file **stm32446e_eval_sd.h** [code]

This file contains the common defines and functions prototypes for the **stm32446e_eval_sd.c** driver.

file **stm32446e_eval_sdram.c** [code]

This file includes the SDRAM driver for the MT48LC4M32B2B5-7 memory device mounted on STM32446E-EVAL evaluation board.

file **stm32446e_eval_sdram.h** [code]

This file contains the common defines and functions prototypes for the **stm32446e_eval_sdram.c** driver.

file **stm32446e_eval_ts.c** [code]

This file provides a set of functions needed to manage the Touch Screen on STM32446E-EVAL evaluation board.

file **stm32446e_eval_ts.h** [code]

This file contains the common defines and functions prototypes for the **stm32446e_eval_ts.c** driver.

STM32446E_EVAL BSP User Manual

stm32446e_eval.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval.h
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file contains definitions for STM32446E_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 /* IMPORTANT: in order to compile with RevA
following flag shall be defined  */
00040 /* in the preprocessor options:  USE_STM3244
6E_EVAL_REVA !!!!!!!!! */
00041
00042 /* Define to prevent recursive inclusion ---
-----------------------------------*/
00043 #ifndef __STM32446E_EVAL_H
00044 #define __STM32446E_EVAL_H
00045
00046 #ifdef __cplusplus
00047  extern "C" {
00048 #endif
00049
00050 /* USE_STM32446E_EVAL_REVA must USE USE_IOEX
PANDER */
00051 #if defined(USE_STM32446E_EVAL_REVA)
00052 #ifndef USE_IOEXPANDER
00053 #define USE_IOEXPANDER
00054 #endif // USE_IOEXPANDER
00055 #endif // USE_STM32446E_EVAL_REVA
00056
00057 /* Includes ----------------------------------
-----------------------------------*/
00058 #include "stm32f4xx_hal.h"
00059
00060 /** @addtogroup BSP
00061   * @{
00062   */
00063
00064 /** @addtogroup STM32446E_EVAL
00065   * @{
```

```
00066   */
00067
00068 /** @defgroup STM32446E_EVAL_LOW_LEVEL STM32
446E EVAL LOW LEVEL
00069   * @{
00070   */
00071
00072 /** @defgroup STM32446E_EVAL_LOW_LEVEL_Expor
ted_Types STM32446E EVAL LOW LEVEL Exported Types
00073   * @{
00074   */
00075 typedef enum
00076 {
00077 #if defined(USE_IOEXPANDER)
00078 LED1 = 0,
00079 LED_GREEN = LED1,
00080 LED2 = 1,
00081 LED_ORANGE = LED2,
00082 LED3 = 2,
00083 LED_RED = LED3,
00084 LED4 = 3,
00085 LED_BLUE = LED4
00086 #else
00087 LED1 = 0,
00088 LED_GREEN = LED1,
00089 LED3 = 1,
00090 LED_RED = LED3,
00091 #endif /* USE_IOEXPANDER */
00092 }Led_TypeDef;
00093
00094
00095 typedef enum
00096 {
00097   BUTTON_WAKEUP = 0,
00098   BUTTON_TAMPER = 1,
00099   BUTTON_KEY = 2
00100 }Button_TypeDef;
```

```
00101
00102 typedef enum
00103 {
00104   BUTTON_MODE_GPIO = 0,
00105   BUTTON_MODE_EXTI = 1
00106 }ButtonMode_TypeDef;
00107
00108 #if defined(USE_IOEXPANDER)
00109 typedef enum
00110 {
00111   JOY_MODE_GPIO = 0,
00112   JOY_MODE_EXTI = 1
00113 }JOYMode_TypeDef;
00114
00115 typedef enum
00116 {
00117   JOY_NONE  = 0,
00118   JOY_SEL   = 1,
00119   JOY_DOWN  = 2,
00120   JOY_LEFT  = 3,
00121   JOY_RIGHT = 4,
00122   JOY_UP    = 5
00123 }JOYState_TypeDef;
00124 #endif /* USE_IOEXPANDER */
00125
00126 typedef enum
00127 {
00128   COM1 = 0,
00129   COM2 = 1
00130 }COM_TypeDef;
00131 /**
00132   * @}
00133   */
00134
00135 /** @defgroup STM32446E_EVAL_LOW_LEVEL_Exported_Constants STM32446E EVAL LOW LEVEL Exported Constants
```

```
00136   * @{
00137   */
00138
00139 /**
00140   * @brief  Define for STM32446E_EVAL board
00141   */
00142 #if !defined (USE_STM32446E_EVAL)
00143  #define USE_STM32446E_EVAL
00144 #endif
00145
00146 /** @addtogroup STM32446E_EVAL_LOW_LEVEL_LED
 STM32446E EVAL LOW LEVEL LED
00147   * @{
00148   */
00149
00150 #if !defined(USE_STM32446E_EVAL_REVA)
00151
00152 #if defined(USE_IOEXPANDER)
00153 #define LEDn                                    ((u
int8_t)4)
00154 #define LED2_PIN                                IO_
PIN_21
00155 #define LED4_PIN                                IO_
PIN_23
00156 #else
00157 #define LEDn                                    ((u
int8_t)2)
00158 #endif /* USE_IOEXPANDER */
00159
00160 #define LEDx_GPIO_PORT                          GPI
OB
00161 #define LEDx_GPIO_CLK_ENABLE()                  __H
AL_RCC_GPIOB_CLK_ENABLE()
00162 #define LEDx_GPIO_CLK_DISABLE()                 __H
AL_RCC_GPIOB_CLK_DISABLE()
00163 #define LED1_PIN                                GPI
O_PIN_11
```

```
00164 #define LED3_PIN                                GPI
O_PIN_4
00165
00166 #else
00167
00168 #define LEDn                                    ((u
int8_t)4)
00169 #define LED1_PIN                                IO_
PIN_20
00170 #define LED3_PIN                                IO_
PIN_22
00171 #define LED2_PIN                                IO_
PIN_21
00172 #define LED4_PIN                                IO_
PIN_23
00173
00174 #endif // !USE_STM32446E_EVAL_REVA
00175
00176
00177 /**
00178   * @}
00179   */
00180
00181 /** @addtogroup STM32446E_EVAL_LOW_LEVEL_BUT
TON STM32446E EVAL LOW LEVEL BUTTON
00182   * @{
00183   */
00184 /* Joystick pins are connected to IO Expande
r (accessible through FMPI2C interface) */
00185 #define BUTTONn
((uint8_t)3)
00186
00187 /**
00188   * @brief Wakeup push-button
00189   */
00190 #define WAKEUP_BUTTON_PIN
GPIO_PIN_13
```

```
00191 #define WAKEUP_BUTTON_GPIO_PORT
GPIOC
00192 #define WAKEUP_BUTTON_GPIO_CLK_ENABLE()
__HAL_RCC_GPIOC_CLK_ENABLE()
00193 #define WAKEUP_BUTTON_GPIO_CLK_DISABLE()
__HAL_RCC_GPIOC_CLK_DISABLE()
00194 #define WAKEUP_BUTTON_EXTI_IRQn
EXTI15_10_IRQn
00195
00196 /**
00197   * @brief Tamper push-button
00198   */
00199 #define TAMPER_BUTTON_PIN
 GPIO_PIN_13
00200 #define TAMPER_BUTTON_GPIO_PORT
 GPIOC
00201 #define TAMPER_BUTTON_GPIO_CLK_ENABLE()
 __HAL_RCC_GPIOC_CLK_ENABLE()
00202 #define TAMPER_BUTTON_GPIO_CLK_DISABLE()
 __HAL_RCC_GPIOC_CLK_DISABLE()
00203 #define TAMPER_BUTTON_EXTI_IRQn
 EXTI15_10_IRQn
00204
00205 /**
00206   * @brief Key push-button
00207   */
00208 #define KEY_BUTTON_PIN
 GPIO_PIN_13
00209 #define KEY_BUTTON_GPIO_PORT
 GPIOC
00210 #define KEY_BUTTON_GPIO_CLK_ENABLE()
 __HAL_RCC_GPIOC_CLK_ENABLE()
00211 #define KEY_BUTTON_GPIO_CLK_DISABLE()
 __HAL_RCC_GPIOC_CLK_DISABLE()
00212 #define KEY_BUTTON_EXTI_IRQn
 EXTI15_10_IRQn
00213
```

```
00214 /**
00215   * @brief MFX Irq Out Pin
00216   */
00217 #define MFX_IRQOUT_PIN                   GP
IO_PIN_0
00218 #define MFX_IRQOUT_GPIO_PORT             GP
IOA
00219 #define MFX_IRQOUT_GPIO_CLK_ENABLE()     __
HAL_RCC_GPIOA_CLK_ENABLE()
00220 #define MFX_IRQOUT_GPIO_CLK_DISABLE()    __
HAL_RCC_GPIOA_CLK_DISABLE()
00221 #define MFX_IRQOUT_EXTI_IRQn             EX
TI0_IRQn
00222
00223 #define BUTTONx_GPIO_CLK_ENABLE(__INDEX__)
  do { if((__INDEX__) == 0) {WAKEUP_BUTTON_GPIO_CL
K_ENABLE();} else\
00224
       if((__INDEX__) == 1) {TAMPER_BUTTON_GPIO_CL
K_ENABLE();} else\
00225
                            {KEY_BUTTON_GPIO_CLK_E
NABLE();   }} while(0)


00226
00227 #define BUTTONx_GPIO_CLK_DISABLE(__INDEX__)
   (((__INDEX__) == 0) ? WAKEUP_BUTTON_GPIO_CLK_DI
SABLE() :\
00228
    ((__INDEX__) == 1) ? TAMPER_BUTTON_GPIO_CLK_DI
SABLE() : KEY_BUTTON_GPIO_CLK_DISABLE())
00229 /**
00230   * @}
00231   */
00232
00233 /** @addtogroup STM32446E_EVAL_LOW_LEVEL_COM
```

```
 STM32446E EVAL LOW LEVEL COM
00234   * @{
00235   */
00236 #define COMn                                ((u
int8_t)1)
00237
00238 /**
00239  * @brief Definition for COM port1, connecte
d to USART1
00240  */
00241 #define EVAL_COM1                             U
SART1
00242 #define EVAL_COM1_CLK_ENABLE()                _
_HAL_RCC_USART1_CLK_ENABLE()
00243 #define EVAL_COM1_CLK_DISABLE()               _
_HAL_RCC_USART1_CLK_DISABLE()
00244
00245 #define EVAL_COM1_TX_PIN                      G
PIO_PIN_9
00246 #define EVAL_COM1_TX_GPIO_PORT                G
PIOA
00247 #define EVAL_COM1_TX_GPIO_CLK_ENABLE()     _
_HAL_RCC_GPIOA_CLK_ENABLE()
00248 #define EVAL_COM1_TX_GPIO_CLK_DISABLE()    _
_HAL_RCC_GPIOA_CLK_DISABLE()
00249 #define EVAL_COM1_TX_AF                       G
PIO_AF7_USART1
00250
00251 #define EVAL_COM1_RX_PIN                      G
PIO_PIN_10
00252 #define EVAL_COM1_RX_GPIO_PORT                G
PIOA
00253 #define EVAL_COM1_RX_GPIO_CLK_ENABLE()     _
_HAL_RCC_GPIOA_CLK_ENABLE()
00254 #define EVAL_COM1_RX_GPIO_CLK_DISABLE()    _
_HAL_RCC_GPIOA_CLK_DISABLE()
00255 #define EVAL_COM1_RX_AF                       G
```

```
PIO_AF7_USART1
00256
00257 #define EVAL_COM1_IRQn                    U
SART1_IRQn
00258
00259 #define EVAL_COMx_CLK_ENABLE(__INDEX__)
       do { if((__INDEX__) == COM1) {EVAL_COM1_CLK
_ENABLE();} } while(0)
00260 #define EVAL_COMx_CLK_DISABLE(__INDEX__)
       (((__INDEX__) == 0) ? EVAL_COM1_CLK_DISABLE
() : 0)
00261
00262 #define EVAL_COMx_TX_GPIO_CLK_ENABLE(__INDEX
__)    do { if((__INDEX__) == COM1) {EVAL_COM1_TX_
GPIO_CLK_ENABLE();} } while(0)
00263 #define EVAL_COMx_TX_GPIO_CLK_DISABLE(__INDE
X__)   (((__INDEX__) == 0) ? EVAL_COM1_TX_GPIO_CLK
_DISABLE() : 0)
00264
00265 #define EVAL_COMx_RX_GPIO_CLK_ENABLE(__INDEX
__)    do { if((__INDEX__) == COM1) {EVAL_COM1_RX_
GPIO_CLK_ENABLE();} } while(0)
00266 #define EVAL_COMx_RX_GPIO_CLK_DISABLE(__INDE
X__)   (((__INDEX__) == 0) ? EVAL_COM1_RX_GPIO_CLK
_DISABLE() : 0)
00267
00268 /**
00269   * @brief Joystick Pins definition
00270   */
00271 #if defined(USE_IOEXPANDER)
00272
00273 #define JOY_SEL_PIN                       IO_PI
N_5
00274 #define JOY_UP_PIN                        IO_PI
N_4
00275 #define JOY_DOWN_PIN                      IO_PI
N_3
```

```
00276 #define JOY_LEFT_PIN                    IO_PI
N_2
00277 #define JOY_RIGHT_PIN                   IO_PI
N_1
00278 #define JOY_NONE_PIN                    JOY_A
LL_PINS
00279 #define JOY_ALL_PINS                    (IO_P
IN_5 | IO_PIN_1 | IO_PIN_2 | IO_PIN_3 | IO_PIN_4)
00280
00281 #endif /* USE_IOEXPANDER */
00282
00283 /**
00284   * @brief Eval Pins definition connected to
 MFX
00285   */
00286
00287 #if defined(USE_IOEXPANDER)
00288
00289 #define XSDN_PIN                        IO_PI
N_10
00290 #define RSTI_PIN                        IO_PI
N_11
00291 #define CAM_PLUG_PIN                    IO_PI
N_12
00292 #define AUDIO_INT_PIN                   IO_PI
N_5
00293 #define OTG_FS1_OVER_CURRENT_PIN        IO_PI
N_6
00294 #define OTG_FS1_POWER_SWITCH_PIN        IO_PI
N_7
00295 #define OTG_FS2_OVER_CURRENT_PIN        IO_PI
N_8
00296 #define SD_DETECT_PIN                   IO_PI
N_15
00297
00298 #endif /* USE_IOEXPANDER */
00299
```

```
00300
00301 /* Exported constant IO ----------------------
-----------------------------------*/
00302
00303 /*  The MFX_I2C_ADDR input pin selects the M
FX I2C device address
00304         MFX_I2C_ADDR input pin     MFX I2C d
evice address
00305             0                          b: 1
000 010x    (0x84)
00306             1                          b: 1
000 011x    (0x86)
00307    This input is sampled during the MFX firm
ware startup.  */
00308
00309 #define IO_I2C_ADDRESS                  ((u
int16_t)0x84)  /*mfx MFX_I2C_ADDR 0*/
00310 #define IO_I2C_ADDRESS_2                ((u
int16_t)0x86)  /*mfx MFX_I2C_ADDR 1*/
00311 #define TS_I2C_ADDRESS                  ((u
int16_t)0x84)  /*mfx MFX_I2C_ADDR 0*/
00312 #define TS_I2C_ADDRESS_2                ((u
int16_t)0x86)  /*mfx MFX_I2C_ADDR 1*/
00313
00314
00315 #define CAMERA_I2C_ADDRESS              ((u
int16_t)0x5A)
00316 #define AUDIO_I2C_ADDRESS               ((u
int16_t)0x34)
00317 #define EEPROM_I2C_ADDRESS_A01          ((u
int16_t)0xA0)
00318 #define EEPROM_I2C_ADDRESS_A02          ((u
int16_t)0xA6)
00319
00320
00321 /* User can use this section to tailor I2Cx/
I2Cx instance used and associated
```

```
00322    resources */
00323 /* Definition for I2Cx clock resources */
00324 #define EVAL_I2Cx
  FMPI2C1
00325 #define EVAL_I2Cx_CLK_ENABLE()
  __HAL_RCC_FMPI2C1_CLK_ENABLE()
00326 #define EVAL_I2Cx_CLK_DISABLE()
  __HAL_RCC_FMPI2C1_CLK_DISABLE()
00327 #define EVAL_DMAx_CLK_ENABLE()
  __HAL_RCC_DMA1_CLK_ENABLE()
00328 #define EVAL_I2Cx_SCL_SDA_GPIO_CLK_ENABLE()
  __HAL_RCC_GPIOD_CLK_ENABLE()
00329 #define EVAL_I2Cx_SCL_SDA_GPIO_CLK_DISABLE()
  __HAL_RCC_GPIOD_CLK_DISABLE()
00330
00331 #define EVAL_I2Cx_FORCE_RESET()
  __HAL_RCC_FMPI2C1_FORCE_RESET()
00332 #define EVAL_I2Cx_RELEASE_RESET()
  __HAL_RCC_FMPI2C1_RELEASE_RESET()
00333
00334 /* Definition for I2Cx Pins */
00335 #define EVAL_I2Cx_SCL_PIN
  GPIO_PIN_12
00336 #define EVAL_I2Cx_SCL_SDA_GPIO_PORT
  GPIOD
00337 #define EVAL_I2Cx_SCL_SDA_AF
  GPIO_AF4_FMPI2C1
00338 #define EVAL_I2Cx_SDA_PIN
  GPIO_PIN_13
00339
00340 /* I2C interrupt requests */
00341 #define EVAL_I2Cx_EV_IRQn
  FMPI2C1_EV_IRQn
00342 #define EVAL_I2Cx_ER_IRQn
  FMPI2C1_ER_IRQn
00343
00344 #ifndef EVAL_I2Cx_TIMING
```

```
00345 #define EVAL_I2Cx_TIMING
  ((uint32_t)0x00805252)
00346 #endif /* EVAL_I2Cx_TIMING */
00347
00348
00349 /**
00350   * @}
00351   */
00352
00353 /**
00354   * @}
00355   */
00356
00357 /** @defgroup STM32446E_EVAL_LOW_LEVEL_Expor
ted_Macros STM32446E EVAL LOW LEVEL Exported Macros

00358   * @{
00359   */
00360 /**
00361   * @}
00362   */
00363
00364 /** @defgroup STM32446E_EVAL_LOW_LEVEL_Expor
ted_Functions STM32446E EVAL LOW LEVEL Exported Fu
nctions
00365   * @{
00366   */
00367 uint32_t         BSP_GetVersion(void);
00368 void             BSP_LED_Init(Led_TypeDef Le
d);
00369 void             BSP_LED_DeInit(Led_TypeDef
Led);
00370 void             BSP_LED_On(Led_TypeDef Led)
;
00371 void             BSP_LED_Off(Led_TypeDef Led
);
00372 void             BSP_LED_Toggle(Led_TypeDef
```

```
Led);
00373 void              BSP_PB_Init(Button_TypeDef
Button, ButtonMode_TypeDef ButtonMode);
00374 void              BSP_PB_DeInit(Button_TypeDef
 Button);
00375 uint32_t          BSP_PB_GetState(Button_Type
Def Button);
00376 void              BSP_COM_Init(COM_TypeDef CO
M, UART_HandleTypeDef *husart);
00377 void              BSP_COM_DeInit(COM_TypeDef
COM, UART_HandleTypeDef *huart);
00378 #if defined(USE_IOEXPANDER)
00379 uint8_t           BSP_JOY_Init(JOYMode_TypeDe
f JoyMode);
00380 void              BSP_JOY_DeInit(void);
00381 JOYState_TypeDef BSP_JOY_GetState(void);
00382 void              BSP_MFX_reg_access_for_debu
g(void);
00383 #endif /* USE_IOEXPANDER */
00384
00385 /**
00386   * @}
00387   */
00388
00389 /**
00390   * @}
00391   */
00392
00393 /**
00394   * @}
00395   */
00396
00397 /**
00398   * @}
00399   */
00400
00401 #ifdef __cplusplus
```

```
00402 }
00403 #endif
00404
00405 #endif /* __STM32446E_EVAL_H */
00406
00407 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM32446E_EVAL BSP User Manual

stm32446e_eval_audio.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval_audio.h
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm32446e_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 __STM32446E_EVAL_AUDIO_H
00041 #define __STM32446E_EVAL_AUDIO_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------
-----------------------------------*/
00048 /* Include audio component Driver */
00049 #include "../Components/wm8994/wm8994.h"
00050 #include "stm32446e_eval.h"
00051 #include "../../../Middlewares/ST/STM32_Audi
o/Addons/PDM/pdm_filter.h"
00052
00053 /** @addtogroup BSP
00054   * @{
00055   */
00056
00057 /** @addtogroup STM32446E_EVAL
00058   * @{
00059   */
00060
00061 /** @addtogroup STM32446E_EVAL_AUDIO
00062   * @{
00063   */
00064
00065 /** @defgroup STM32446E_EVAL_AUDIO_Exported_
Types STM32446E EVAL AUDIO Exported Types
00066   * @{
```

```
00067   */
00068 /**
00069   * @}
00070   */
00071
00072 /** @defgroup STM32446E_EVAL_AUDIO_Exported_
Constants  STM32446E EVAL AUDIO Exported Constants
00073   * @{
00074   */
00075
00076 /*-----------------------------------------
-----------------------------------
00077                                USER SAI defines p
arameters
00078  -----------------------------------------
-----------------------------------*/
00079 /** @defgroup CODEC_AudioFrame_SLOT_TDMMode
 STM32446E EVAL Audio Slot TDM mode
00080   * @brief In W8994 codec the Audio frame co
ntains 4 slots : TDM Mode
00081   * TDM format :
00082   * +------------------|------------------|-
------------------|-------------------+
00083   * | CODEC_SLOT0 Left | CODEC_SLOT1 Left |
CODEC_SLOT0 Right  | CODEC_SLOT1 Right |
00084   * +-------------------------------------
-------------------------------------+
00085   * @{
00086   */
00087 /* To have 2 separate audio stream in Both h
eadphone and speaker the 4 slot must be activated
*/
00088 #define CODEC_AUDIOFRAME_SLOT_0123
         SAI_SLOTACTIVE_0 | SAI_SLOTACTIVE_1 | SAI
_SLOTACTIVE_2 | SAI_SLOTACTIVE_3
00089 /* To have an audio stream in headphone only
 SAI Slot 0 and Slot 2 must be activated */
```

```
00090 #define CODEC_AUDIOFRAME_SLOT_02
         SAI_SLOTACTIVE_0 | SAI_SLOTACTIVE_2
00091 /* To have an audio stream in speaker only S
AI Slot 1 and Slot 3 must be activated */
00092 #define CODEC_AUDIOFRAME_SLOT_13
         SAI_SLOTACTIVE_1 | SAI_SLOTACTIVE_3
00093 /**
00094   * @}
00095   */
00096
00097 /* SAI peripheral configuration defines */
00098 #define AUDIO_SAIx
 SAI2_Block_B
00099 #define AUDIO_SAIx_CLK_ENABLE()
 __HAL_RCC_SAI2_CLK_ENABLE()
00100 #define AUDIO_SAIx_CLK_DISABLE()
 __HAL_RCC_SAI2_CLK_DISABLE()
00101 #define AUDIO_SAIx_SCK_AF
 GPIO_AF8_SAI2
00102 #define AUDIO_SAIx_MCLK_SD_FS_AF
 GPIO_AF10_SAI2
00103
00104
00105 #define AUDIO_SAIx_MCLK_SCK_ENABLE()
 __HAL_RCC_GPIOA_CLK_ENABLE()
00106 #define AUDIO_SAIx_MCLK_SCK_DISABLE()
 __HAL_RCC_GPIOA_CLK_DISABLE()
00107 #define AUDIO_SAIx_MCK_PIN
 GPIO_PIN_1 /* PA1 */
00108 #define AUDIO_SAIx_SCK_PIN
 GPIO_PIN_2 /* PA2 */
00109 #define AUDIO_SAIx_MCLK_SCK_GPIO_PORT
 GPIOA
00110 #define AUDIO_SAIx_SD_FS_ENABLE()
 __HAL_RCC_GPIOG_CLK_ENABLE()
00111 #define AUDIO_SAIx_SD_FS_DISABLE()
 __HAL_RCC_GPIOG_CLK_DISABLE()
```

```
00112 #define AUDIO_SAIx_FS_PIN
 GPIO_PIN_9 /* PG9 */
00113 #define AUDIO_SAIx_SD_PIN
 GPIO_PIN_10 /* PG10 */
00114 #define AUDIO_SAIx_SD_FS_GPIO_PORT
 GPIOG
00115
00116
00117 /* SAI DMA Stream definitions */
00118 #define AUDIO_SAIx_DMAx_CLK_ENABLE()
 __HAL_RCC_DMA2_CLK_ENABLE()
00119 #define AUDIO_SAIx_DMAx_CLK_DISABLE()
 __HAL_RCC_DMA2_CLK_DISABLE()
00120 #define AUDIO_SAIx_DMAx_STREAM
 DMA2_Stream6
00121 #define AUDIO_SAIx_DMAx_CHANNEL
 DMA_CHANNEL_3
00122 #define AUDIO_SAIx_DMAx_IRQ
 DMA2_Stream6_IRQn
00123 #define AUDIO_SAIx_DMAx_PERIPH_DATA_SIZE
 DMA_PDATAALIGN_HALFWORD
00124 #define AUDIO_SAIx_DMAx_MEM_DATA_SIZE
 DMA_MDATAALIGN_HALFWORD
00125 #define DMA_MAX_SZE
 0xFFFF
00126
00127 #define AUDIO_SAIx_DMAx_IRQHandler
 DMA2_Stream6_IRQHandler
00128
00129 /* Select the interrupt preemption priority
for the DMA interrupt */
00130 #define AUDIO_OUT_IRQ_PREPRIO           ((ui
nt32_t)5)   /* Select the preemption priority leve
l(0 is the highest) */
00131
00132 /*-----------------------------------------
-----------------------------------
```

```
00133                                   AUDIO IN CONFIGURATI
ON
00134 ----------------------------------------------
----------------------------------*/
00135 /* SPI Configuration defines */
00136 #define AUDIO_I2Sx
SPI3
00137 #define AUDIO_I2Sx_CLK_ENABLE()
__HAL_RCC_SPI3_CLK_ENABLE()
00138 #define AUDIO_I2Sx_CLK_DISABLE()
__HAL_RCC_SPI3_CLK_DISABLE()
00139 #define AUDIO_I2Sx_SCK_PIN
 GPIO_PIN_3
00140 #define AUDIO_I2Sx_SCK_GPIO_PORT
 GPIOB
00141 #define AUDIO_I2Sx_SCK_GPIO_CLK_ENABLE()
 __HAL_RCC_GPIOB_CLK_ENABLE()
00142 #define AUDIO_I2Sx_SCK_GPIO_CLK_DISABLE()
 __HAL_RCC_GPIOB_CLK_DISABLE()
00143 #define AUDIO_I2Sx_SCK_AF
 GPIO_AF6_SPI3
00144
00145 #define AUDIO_I2Sx_SD_PIN
GPIO_PIN_1
00146 #define AUDIO_I2Sx_SD_GPIO_PORT
GPIOC
00147 #define AUDIO_I2Sx_SD_GPIO_CLK_ENABLE()
__HAL_RCC_GPIOC_CLK_ENABLE()
00148 #define AUDIO_I2Sx_SD_GPIO_CLK_DISABLE()
__HAL_RCC_GPIOC_CLK_DISABLE()
00149 #define AUDIO_I2Sx_SD_AF
GPIO_AF5_SPI3
00150
00151 /* I2S DMA Stream Rx definitions */
00152 #define AUDIO_I2Sx_DMAx_CLK_ENABLE()
__HAL_RCC_DMA1_CLK_ENABLE()
00153 #define AUDIO_I2Sx_DMAx_CLK_DISABLE()
```

```
__HAL_RCC_DMA1_CLK_DISABLE()
00154 #define AUDIO_I2Sx_DMAx_STREAM
DMA1_Stream2
00155 #define AUDIO_I2Sx_DMAx_CHANNEL
DMA_CHANNEL_0
00156 #define AUDIO_I2Sx_DMAx_IRQ
DMA1_Stream2_IRQn
00157 #define AUDIO_I2Sx_DMAx_PERIPH_DATA_SIZE
DMA_PDATAALIGN_HALFWORD
00158 #define AUDIO_I2Sx_DMAx_MEM_DATA_SIZE
DMA_MDATAALIGN_HALFWORD
00159
00160 #define AUDIO_I2Sx_DMAx_IRQHandler
DMA1_Stream2_IRQHandler
00161
00162 /* Select the interrupt preemption priority
and subpriority for the IT/DMA interrupt */
00163 #define AUDIO_IN_IRQ_PREPRIO
((uint32_t)6)   /* Select the preemption priority
level(0 is the highest) */
00164
00165
00166 /* Two channels are used:
00167    - one channel as input which is connected
 to I2S SCK in stereo mode
00168    - one channel as output which divides the
 frequency on the input
00169 */
00170
00171 #define AUDIO_TIMx_CLK_ENABLE()
__HAL_RCC_TIM4_CLK_ENABLE()
00172 #define AUDIO_TIMx_CLK_DISABLE()
__HAL_RCC_TIM4_CLK_DISABLE()
00173 #define AUDIO_TIMx
TIM4
00174 #define AUDIO_TIMx_IN_CHANNEL
TIM_CHANNEL_1
```

```
00175 #define AUDIO_TIMx_OUT_CHANNEL
TIM_CHANNEL_2 /* Select channel 2 as output */
00176 #define AUDIO_TIMx_GPIO_CLK_ENABLE()
__HAL_RCC_GPIOB_CLK_ENABLE()
00177 #define AUDIO_TIMx_GPIO_CLK_DISABLE()
__HAL_RCC_GPIOB_CLK_DISABLE()
00178 #define AUDIO_TIMx_GPIO_PORT
GPIOB
00179 #define AUDIO_TIMx_IN_GPIO_PIN
GPIO_PIN_6
00180 #define AUDIO_TIMx_OUT_GPIO_PIN
GPIO_PIN_7
00181 #define AUDIO_TIMx_AF
GPIO_AF2_TIM4
00182
00183 /*-----------------------------------------
-----------------------------------
00184              CONFIGURATION: Audio Driver Con
figuration parameters
00185 -----------------------------------------
-----------------------------------*/
00186
00187 #define AUDIODATA_SIZE
2   /* 16-bits audio data size */
00188
00189 /* Audio status definition */
00190 #define AUDIO_OK
((uint8_t)0)
00191 #define AUDIO_ERROR
((uint8_t)1)
00192 #define AUDIO_TIMEOUT
((uint8_t)2)
00193
00194 /* AudioFreq * DataSize (2 bytes) * NumChann
els (Stereo: 2) */
00195 #define DEFAULT_AUDIO_IN_FREQ
I2S_AUDIOFREQ_16K
```

```
00196 #define DEFAULT_AUDIO_IN_BIT_RESOLUTION
((uint8_t)16)
00197 #define DEFAULT_AUDIO_IN_CHANNEL_NBR
((uint8_t)2) /* Mono = 1, Stereo = 2 */
00198 #define DEFAULT_AUDIO_IN_VOLUME
((uint16_t)64)
00199
00200 /* PDM buffer input size */
00201 #define INTERNAL_BUFF_SIZE
(128*DEFAULT_AUDIO_IN_FREQ/16000*DEFAULT_AUDIO_IN_
CHANNEL_NBR)
00202 /* PCM buffer output size */
00203 #define PCM_OUT_SIZE
(DEFAULT_AUDIO_IN_FREQ/1000*2)
00204 #define CHANNEL_DEMUX_MASK
((uint8_t)0x55)
00205
00206 /*-----------------------------------------
-----------------------------------
00207                       OPTIONAL Configuration d
efines parameters
00208 -----------------------------------------
---------------------------------*/
00209
00210 /* Delay for the Codec to be correctly reset
 */
00211 #define CODEC_RESET_DELAY           ((uint8_
t)5)
00212
00213
00214 /*-----------------------------------------
-----------------------------------
00215                                 OUTPUT DEVICES d
efinition
00216 -----------------------------------------
---------------------------------*/
00217
```

```
00218 /* Alias on existing output devices to adapt
 for 2 headphones output */
00219 #define OUTPUT_DEVICE_HEADPHONE1 OUTPUT_DEVI
CE_HEADPHONE
00220 #define OUTPUT_DEVICE_HEADPHONE2 OUTPUT_DEVI
CE_SPEAKER /* Headphone2 is connected to Speaker o
utput of the wm8994 */
00221
00222 /**
00223   * @}
00224   */
00225
00226 /** @defgroup STM32446E_EVAL_AUDIO_Exported_
Variables  STM32446E EVAL AUDIO Exported Variables
00227   * @{
00228   */
00229 extern __IO uint16_t AudioInVolume;
00230  /**
00231   * @}
00232   */
00233
00234 /** @defgroup STM32446E_EVAL_AUDIO_Exported_
Macros  STM32446E EVAL AUDIO Exported Macros
00235   * @{
00236   */
00237 #define DMA_MAX(x)           (((x) <= DMA_MA
X_SZE)? (x):DMA_MAX_SZE)
00238 /**
00239   * @}
00240   */
00241
00242 /** @defgroup STM32446E_EVAL_AUDIO_OUT_Expor
ted_Functions  STM32446E EVAL AUDIO OUT Exported F
unctions
00243   * @{
00244   */
00245 uint8_t BSP_AUDIO_OUT_Init(uint16_t OutputDe
```

```
vice, uint8_t Volume, uint32_t AudioFreq);
00246 uint8_t BSP_AUDIO_OUT_Play(uint16_t* pBuffer
, uint32_t Size);
00247 void    BSP_AUDIO_OUT_ChangeBuffer(uint16_t
*pData, uint16_t Size);
00248 uint8_t BSP_AUDIO_OUT_Pause(void);
00249 uint8_t BSP_AUDIO_OUT_Resume(void);
00250 uint8_t BSP_AUDIO_OUT_Stop(uint32_t Option);
00251 uint8_t BSP_AUDIO_OUT_SetVolume(uint8_t Volu
me);
00252 void    BSP_AUDIO_OUT_SetFrequency(uint32_t
AudioFreq);
00253 void    BSP_AUDIO_OUT_SetAudioFrameSlot(uint
32_t AudioFrameSlot);
00254 uint8_t BSP_AUDIO_OUT_SetMute(uint32_t Cmd);
00255 uint8_t BSP_AUDIO_OUT_SetOutputMode(uint8_t
Output);
00256 void    BSP_AUDIO_OUT_DeInit(void);
00257
00258 /* User Callbacks: user has to implement the
se functions in his code if they are needed. */
00259 /* This function is called when the requeste
d data has been completely transferred.*/
00260 void    BSP_AUDIO_OUT_TransferComplete_CallB
ack(void);
00261
00262 /* This function is called when half of the
requested buffer has been transferred. */
00263 void    BSP_AUDIO_OUT_HalfTransfer_CallBack(
void);
00264
00265 /* This function is called when an Interrupt
 due to transfer error on or peripheral
00266    error occurs. */
00267 void    BSP_AUDIO_OUT_Error_CallBack(void);
00268
00269 /* These function can be modified in case th
```

```
e current settings (e.g. DMA stream)
00270    need to be changed for specific applicati
on needs */
00271 void  BSP_AUDIO_OUT_ClockConfig(SAI_HandleTy
peDef *hsai, uint32_t AudioFreq, void *Params);
00272 void  BSP_AUDIO_OUT_MspInit(SAI_HandleTypeDe
f *hsai, void *Params);
00273 void  BSP_AUDIO_OUT_MspDeInit(SAI_HandleType
Def *hsai, void *Params);
00274
00275 /**
00276   * @}
00277   */
00278
00279 /** @defgroup STM32446E_EVAL_AUDIO_IN_Export
ed_Functions  STM32446E EVAL AUDIO IN Exported Fun
ctions
00280   * @{
00281   */
00282 uint8_t BSP_AUDIO_IN_Init(uint32_t AudioFreq
, uint32_t BitRes, uint32_t ChnlNbr);
00283 uint8_t BSP_AUDIO_IN_Record(uint16_t *pData,
 uint32_t Size);
00284 uint8_t BSP_AUDIO_IN_Stop(void);
00285 uint8_t BSP_AUDIO_IN_Pause(void);
00286 uint8_t BSP_AUDIO_IN_Resume(void);
00287 uint8_t BSP_AUDIO_IN_SetVolume(uint8_t Volum
e);
00288 void    BSP_AUDIO_IN_DeInit(void);
00289 uint8_t BSP_AUDIO_IN_PDMToPCM(uint16_t* PDMB
uf, uint16_t* PCMBuf);
00290 /* User Callbacks: user has to implement the
se functions in his code if they are needed. */
00291 /* This function should be implemented by th
e user application.
00292    It is called into this driver when the cu
rrent buffer is filled to prepare the next
```

```
00293    buffer pointer and its size. */
00294 void    BSP_AUDIO_IN_TransferComplete_CallBa
ck(void);
00295 void    BSP_AUDIO_IN_HalfTransfer_CallBack(v
oid);
00296
00297 /* This function is called when an Interrupt
 due to transfer error on or peripheral
00298    error occurs. */
00299 void    BSP_AUDIO_IN_Error_Callback(void);
00300
00301 /* These function can be modified in case th
e current settings (e.g. DMA stream)
00302    need to be changed for specific applicati
on needs */
00303 void  BSP_AUDIO_IN_MspInit(I2S_HandleTypeDef
 *hi2s, void *Params);
00304 void  BSP_AUDIO_IN_MspDeInit(I2S_HandleTypeD
ef *hi2s, void *Params);
00305
00306 /**
00307   * @}
00308   */
00309
00310 /**
00311   * @}
00312   */
00313
00314 /**
00315   * @}
00316   */
00317
00318 /**
00319   * @}
00320   */
00321
00322 #ifdef __cplusplus
```

```
00323 }
00324 #endif
00325
00326 #endif /* __STM32446E_EVAL_AUDIO_H */
00327
00328 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32446E_EVAL BSP User Manual

stm32446e_eval_audio.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval_audio.c
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file provides the Audio driver for the STM32446E-EVAL evaluation board.
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 /*=========================================
=================================
00039                                   User NOTES
00040
00041 How To use this driver:
00042 -----------------------
00043    + This driver supports STM32F4xx devices
on STM32446E-EVAL (MB1045) Evaluation boards.
00044    + Call the function BSP_AUDIO_OUT_Init(
00045                                     OutputDe
vice: physical output mode (OUTPUT_DEVICE_SPEAKER,

00046
      OUTPUT_DEVICE_HEADPHONE or OUTPUT_DEVICE_BOT
H)
00047                                     Volume
    : Initial volume to be set (0 is min (mute), 1
00 is max (100%)
00048                                     AudioFre
q   : Audio frequency in Hz (8000, 16000, 22500, 3
2000...)
00049
      this parameter is relative to the audio file
/stream type.
00050                                    )
00051       This function configures all the hardw
are required for the audio application (codec, I2C
, SAI,
00052       GPIOs, DMA and interrupt if needed). T
his function returns AUDIO_OK if configuration is
OK.
00053       If the returned value is different fro
m AUDIO_OK or the function is stuck then the commu
nication with
```

```
00054       the codec or the MFX has failed (try t
o un-plug the power or reset device in this case).
00055       - OUTPUT_DEVICE_SPEAKER  : only speake
r will be set as output for the audio stream.
00056       - OUTPUT_DEVICE_HEADPHONE: only headph
ones will be set as output for the audio stream.
00057       - OUTPUT_DEVICE_BOTH     : both Speake
r and Headphone are used as outputs for the audio
stream
00058                                       at the same
 time.
00059       Note. On STM32446E-EVAL SAI_DMA is con
figured in CIRCULAR mode. Due to this the applicat
ion
00060         does NOT need to call BSP_AUDIO_OUT_
ChangeBuffer() to assure streaming.
00061    + Call the function BSP_EVAL_AUDIO_OUT_Pl
ay(
00062                                        pBuffer: p
ointer to the audio data file address
00063                                        Size   : s
ize of the buffer to be sent in Bytes
00064                                      )
00065       to start playing (for the first time)
from the audio file/stream.
00066    + Call the function BSP_AUDIO_OUT_Pause()
 to pause playing
00067    + Call the function BSP_AUDIO_OUT_Resume(
) to resume playing.
00068        Note. After calling BSP_AUDIO_OUT_Pau
se() function for pause, only BSP_AUDIO_OUT_Resume
() should be called
00069           for resume (it is not allowed to c
all BSP_AUDIO_OUT_Play() in this case).
00070        Note. This function should be called
only when the audio file is played or paused (not
stopped).
```

```
00071    + For each mode, you may need to implemen
t the relative callback functions into your code.
00072       The Callback functions are named AUDIO
_OUT_XXX_CallBack() and only their prototypes are
declared in
00073       the stm32446e_eval_audio.h file. (refe
r to the example for more details on the callbacks
 implementations)
00074    + To Stop playing, to modify the volume l
evel, the frequency, the audio frame slot,
00075       the device output mode the mute or the
 stop, use the functions: BSP_AUDIO_OUT_SetVolume(
),
00076       AUDIO_OUT_SetFrequency(), BSP_AUDIO_OU
T_SetAudioFrameSlot(), BSP_AUDIO_OUT_SetOutputMode
(),
00077       BSP_AUDIO_OUT_SetMute() and BSP_AUDIO_
OUT_Stop().
00078    + The driver API and the callback functio
ns are at the end of the stm32446e_eval_audio.h fi
le.
00079
00080
00081 Driver architecture:
00082 --------------------
00083    + This driver provides the High Audio Lay
er: consists of the function API exported in the s
tm32446e_eval_audio.h file
00084      (BSP_AUDIO_OUT_Init(), BSP_AUDIO_OUT_Pl
ay() ...)
00085    + This driver provide also the Media Acce
ss Layer (MAL): which consists of functions allowi
ng to access the media containing/
00086      providing the audio file/stream. These
functions are also included as local functions into

00087      the stm32446e_eval_audio_codec.c file (
```

```
I2Sx_Init(), I2Sx_DeInit(), SAIx_Init() and SAIx_D
eInit())
00088
00089 Known Limitations:
00090 -----------------
00091    1- If the TDM Format used to play in para
llel 2 audio Stream (the first Stream is configure
d in codec SLOT0 and second
00092       Stream in SLOT1) the Pause/Resume, vol
ume and mute feature will control the both streams.

00093    2- Parsing of audio file is not implement
ed (in order to determine audio file properties: M
ono/Stereo, Data size,
00094       File size, Audio Frequency, Audio Data
 header size ...). The configuration is fixed for
the given audio file.
00095    3- Supports only Stereo audio streaming.
00096    4- Supports only 16-bits audio data size.
00097 ==========================================
==================================*/
00098
00099 /* Includes ------------------------------
----------------------------------*/
00100 #include "stm32446e_eval_audio.h"
00101
00102 /** @addtogroup BSP
00103   * @{
00104   */
00105
00106 /** @addtogroup STM32446E_EVAL
00107   * @{
00108   */
00109
00110 /** @defgroup STM32446E_EVAL_AUDIO STM32446E
 EVAL AUDIO
00111   * @brief This file includes the low layer
```

```
driver for wm8994 Audio Codec
00112   *        available on STM32446E-EVAL evalu
ation board(MB1045).
00113   * @{
00114   */
00115
00116 /** @defgroup STM32446E_EVAL_AUDIO_Private_T
ypes STM32446E EVAL AUDIO Private Types
00117   * @{
00118   */
00119 /**
00120   * @}
00121   */
00122
00123 /** @defgroup STM32446E_EVAL_AUDIO_Private_D
efines STM32446E EVAL AUDIO Private Defines
00124   * @{
00125   */
00126 /**
00127   * @}
00128   */
00129
00130 /** @defgroup STM32446E_EVAL_AUDIO_Private_M
acros STM32446E EVAL AUDIO Private Macros
00131   * @{
00132   */
00133 /**
00134   * @}
00135   */
00136
00137 /** @defgroup STM32446E_EVAL_AUDIO_Private_V
ariables STM32446E EVAL AUDIO Private Variables
00138   * @{
00139   */
00140
00141 /*
00142 Note:
```

```
00143   these global variables are not compliant t
o naming rules (upper case without "_" ),
00144   but we keep this naming for compatibility,
 in fact these variables (not static)
00145   could have been used by the application, e
xample the stm32f4xx_it.c:
00146     void DMA2_Stream6_IRQHandler(void)
00147      {  HAL_DMA_IRQHandler(haudio_out_sai.hd
matx);  }
00148 */
00149 AUDIO_DrvTypeDef           *audio_drv;
00150 SAI_HandleTypeDef          haudio_out_sai;
00151 I2S_HandleTypeDef          haudio_in_i2s;
00152 TIM_HandleTypeDef          haudio_tim;
00153
00154 PDMFilter_InitStruct Filter[2];
00155 uint8_t Channel_Demux[128] = {
00156     0x00, 0x01, 0x00, 0x01, 0x02, 0x03, 0x02
, 0x03,
00157     0x00, 0x01, 0x00, 0x01, 0x02, 0x03, 0x02
, 0x03,
00158     0x04, 0x05, 0x04, 0x05, 0x06, 0x07, 0x06
, 0x07,
00159     0x04, 0x05, 0x04, 0x05, 0x06, 0x07, 0x06
, 0x07,
00160     0x00, 0x01, 0x00, 0x01, 0x02, 0x03, 0x02
, 0x03,
00161     0x00, 0x01, 0x00, 0x01, 0x02, 0x03, 0x02
, 0x03,
00162     0x04, 0x05, 0x04, 0x05, 0x06, 0x07, 0x06
, 0x07,
00163     0x04, 0x05, 0x04, 0x05, 0x06, 0x07, 0x06
, 0x07,
00164     0x08, 0x09, 0x08, 0x09, 0x0a, 0x0b, 0x0a
, 0x0b,
00165     0x08, 0x09, 0x08, 0x09, 0x0a, 0x0b, 0x0a
, 0x0b,
```

```
00166     0x0c, 0x0d, 0x0c, 0x0d, 0x0e, 0x0f, 0x0e
, 0x0f,
00167     0x0c, 0x0d, 0x0c, 0x0d, 0x0e, 0x0f, 0x0e
, 0x0f,
00168     0x08, 0x09, 0x08, 0x09, 0x0a, 0x0b, 0x0a
, 0x0b,
00169     0x08, 0x09, 0x08, 0x09, 0x0a, 0x0b, 0x0a
, 0x0b,
00170     0x0c, 0x0d, 0x0c, 0x0d, 0x0e, 0x0f, 0x0e
, 0x0f,
00171     0x0c, 0x0d, 0x0c, 0x0d, 0x0e, 0x0f, 0x0e
, 0x0f
00172 };
00173
00174 uint16_t __IO AudioInVolume = DEFAULT_AUDIO_
IN_VOLUME;
00175
00176 /**
00177   * @}
00178   */
00179
00180 /** @defgroup STM32446E_EVAL_AUDIO_Private_F
unction_Prototypes STM32446E EVAL AUDIO Private Fu
nction Prototypes
00181   * @{
00182   */
00183 static void SAIx_Init(uint32_t AudioFreq);
00184 static void SAIx_DeInit(void);
00185 static void I2Sx_Init(uint32_t AudioFreq);
00186 static void I2Sx_DeInit(void);
00187 static void TIMx_IC_MspInit(TIM_HandleTypeDe
f *htim);
00188 static void TIMx_IC_MspDeInit(TIM_HandleType
Def *htim);
00189 static void TIMx_Init(void);
00190 static void TIMx_DeInit(void);
00191 static void PDMDecoder_Init(uint32_t AudioFr
```

```
eq, uint32_t ChnlNbr);
00192 /**
00193   * @}
00194   */
00195
00196 /** @defgroup STM32446E_EVAL_AUDIO_out_Priva
te_Functions STM32446E EVAL AUDIO OUT Private Func
tions
00197   * @{
00198   */
00199
00200 /**
00201   * @brief  Configures the audio peripherals.

00202   * @param  OutputDevice: OUTPUT_DEVICE_SPEA
KER, OUTPUT_DEVICE_HEADPHONE,
00203   *                       or OUTPUT_DEVICE_B
OTH.
00204   * @param  Volume: Initial volume level (fr
om 0 (Mute) to 100 (Max))
00205   * @param  AudioFreq: Audio frequency used
to play the audio stream.
00206   * @note   The I2S PLL input clock must be
done in the user application.
00207   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00208   */
00209 uint8_t BSP_AUDIO_OUT_Init(uint16_t OutputDe
vice, uint8_t Volume, uint32_t AudioFreq)
00210 {
00211   uint8_t ret = AUDIO_ERROR;
00212   uint32_t deviceid = 0x00;
00213
00214   SAIx_DeInit();
00215
00216   /* PLL clock is set depending by the Audio
Freq (44.1khz vs 48khz groups) */
```

```
00217   BSP_AUDIO_OUT_ClockConfig(&haudio_out_sai,
 AudioFreq, NULL);
00218
00219   /* SAI data transfer preparation:
00220   Prepare the Media to be used for the audio
 transfer from memory to SAI peripheral */
00221   haudio_out_sai.Instance = AUDIO_SAIx;
00222   if(HAL_SAI_GetState(&haudio_out_sai) == HA
L_SAI_STATE_RESET)
00223   {
00224     /* Init the SAI MSP: this __weak functio
n can be redefined by the application*/
00225     BSP_AUDIO_OUT_MspInit(&haudio_out_sai, N
ULL);
00226   }
00227   SAIx_Init(AudioFreq);
00228
00229   /* wm8994 codec initialization */
00230   deviceid = wm8994_drv.ReadID(AUDIO_I2C_ADD
RESS);
00231
00232   if((deviceid) == WM8994_ID)
00233   {
00234     /* Initialize the audio driver structure
 */
00235     audio_drv = &wm8994_drv;
00236     ret = AUDIO_OK;
00237   }
00238   else
00239   {
00240     ret = AUDIO_ERROR;
00241   }
00242
00243   if(ret == AUDIO_OK)
00244   {
00245     /* Resets the audio codec. */
00246     audio_drv->Reset(AUDIO_I2C_ADDRESS);
```

```
00247     /* Initialize the codec internal registe
rs */
00248     audio_drv->Init(AUDIO_I2C_ADDRESS, Outpu
tDevice, Volume, AudioFreq);
00249   }
00250
00251   return ret;
00252 }
00253
00254 /**
00255   * @brief  Starts playing audio stream from
 a data buffer for a determined size.
00256   * @param  pBuffer: Pointer to the buffer
00257   * @param  Size: Number of audio data BYTES.

00258   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00259   */
00260 uint8_t BSP_AUDIO_OUT_Play(uint16_t* pBuffer
, uint32_t Size)
00261 {
00262   /* Call the audio Codec Play function */
00263   if(audio_drv->Play(AUDIO_I2C_ADDRESS, pBuf
fer, Size) != 0)
00264   {
00265     return AUDIO_ERROR;
00266   }
00267   else
00268   {
00269     /* Update the Media layer and enable it
for play */
00270     HAL_SAI_Transmit_DMA(&haudio_out_sai, (u
int8_t*) pBuffer, DMA_MAX(Size / AUDIODATA_SIZE));
00271
00272     return AUDIO_OK;
00273   }
00274 }
```

```
00275
00276 /**
00277   * @brief  Sends n-Bytes on the SAI interfa
ce.
00278   * @param  pData: pointer on data address
00279   * @param  Size: number of data to be writt
en
00280   */
00281 void BSP_AUDIO_OUT_ChangeBuffer(uint16_t *pD
ata, uint16_t Size)
00282 {
00283    HAL_SAI_Transmit_DMA(&haudio_out_sai, (ui
nt8_t*) pData, Size);
00284 }
00285
00286 /**
00287   * @brief  This function Pauses the audio f
ile stream. In case
00288   *         of using DMA, the DMA Pause feat
ure is used.
00289   * WARNING: When calling BSP_AUDIO_OUT_Paus
e() function for pause, only
00290   *          BSP_AUDIO_OUT_Resume() function
 should be called for resume (use of BSP_AUDIO_OUT
_Play()
00291   *          function for resume could lead
to unexpected behavior).
00292   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00293   */
00294 uint8_t BSP_AUDIO_OUT_Pause(void)
00295 {
00296   /* Call the Audio Codec Pause/Resume funct
ion */
00297   if(audio_drv->Pause(AUDIO_I2C_ADDRESS) !=
0)
00298   {
```

```
00299     return AUDIO_ERROR;
00300   }
00301   else
00302   {
00303     /* Call the Media layer pause function */

00304     HAL_SAI_DMAPause(&haudio_out_sai);
00305
00306     /* Return AUDIO_OK when all operations a
re correctly done */
00307     return AUDIO_OK;
00308   }
00309 }
00310
00311 /**
00312   * @brief  This function  Resumes the audio
 file stream.
00313   * WARNING: When calling BSP_AUDIO_OUT_Paus
e() function for pause, only
00314   *          BSP_AUDIO_OUT_Resume() function
 should be called for resume (use of BSP_AUDIO_OUT
_Play()
00315   *          function for resume could lead
to unexpected behavior).
00316   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00317   */
00318 uint8_t BSP_AUDIO_OUT_Resume(void)
00319 {
00320   /* Call the Audio Codec Pause/Resume funct
ion */
00321   if(audio_drv->Resume(AUDIO_I2C_ADDRESS) !=
 0)
00322   {
00323     return AUDIO_ERROR;
00324   }
00325   else
```

```
00326   {
00327     /* Call the Media layer pause/resume fun
ction */
00328     HAL_SAI_DMAResume(&haudio_out_sai);
00329
00330     /* Return AUDIO_OK when all operations a
re correctly done */
00331     return AUDIO_OK;
00332   }
00333 }
00334
00335 /**
00336   * @brief  Stops audio playing and Power do
wn the Audio Codec.
00337   * @param  Option: could be one of the foll
owing parameters
00338   *           - CODEC_PDWN_SW: for software
power off (by writing registers).
00339   *                                Then no need
to reconfigure the Codec after power on.
00340   *           - CODEC_PDWN_HW: completely sh
ut down the codec (physically).
00341   *                                Then need to
reconfigure the Codec after power on.
00342   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00343   */
00344 uint8_t BSP_AUDIO_OUT_Stop(uint32_t Option)
00345 {
00346   /* Call the Media layer stop function */
00347   HAL_SAI_DMAStop(&haudio_out_sai);
00348
00349   /* Call Audio Codec Stop function */
00350   if(audio_drv->Stop(AUDIO_I2C_ADDRESS, Opti
on) != 0)
00351   {
00352     return AUDIO_ERROR;
```

```
00353   }
00354   else
00355   {
00356     if(Option == CODEC_PDWN_HW)
00357     {
00358       /* Wait at least 100us */
00359       HAL_Delay(1);
00360     }
00361     /* Return AUDIO_OK when all operations a
re correctly done */
00362     return AUDIO_OK;
00363   }
00364 }
00365
00366 /**
00367   * @brief  Controls the current audio volum
e level.
00368   * @param  Volume: Volume level to be set i
n percentage from 0% to 100% (0 for
00369   *         Mute and 100 for Max volume leve
l).
00370   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00371   */
00372 uint8_t BSP_AUDIO_OUT_SetVolume(uint8_t Volu
me)
00373 {
00374   /* Call the codec volume control function
with converted volume value */
00375   if(audio_drv->SetVolume(AUDIO_I2C_ADDRESS,
 Volume) != 0)
00376   {
00377     return AUDIO_ERROR;
00378   }
00379   else
00380   {
00381     /* Return AUDIO_OK when all operations a
```

```
re correctly done */
00382     return AUDIO_OK;
00383   }
00384 }
00385
00386 /**
00387   * @brief  Enables or disables the MUTE mod
e by software
00388   * @param  Cmd: Could be AUDIO_MUTE_ON to m
ute sound or AUDIO_MUTE_OFF to
00389   *         unmute the codec and restore pre
vious volume level.
00390   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00391   */
00392 uint8_t BSP_AUDIO_OUT_SetMute(uint32_t Cmd)
00393 {
00394   /* Call the Codec Mute function */
00395   if(audio_drv->SetMute(AUDIO_I2C_ADDRESS, C
md) != 0)
00396   {
00397     return AUDIO_ERROR;
00398   }
00399   else
00400   {
00401     /* Return AUDIO_OK when all operations a
re correctly done */
00402     return AUDIO_OK;
00403   }
00404 }
00405
00406 /**
00407   * @brief  Switch dynamically (while audio
file is played) the output target
00408   *         (speaker or headphone).
00409   * @param  Output: The audio output target:
 OUTPUT_DEVICE_SPEAKER,
```

```
00410   *         OUTPUT_DEVICE_HEADPHONE or OUTPU
T_DEVICE_BOTH
00411   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00412   */
00413 uint8_t BSP_AUDIO_OUT_SetOutputMode(uint8_t
Output)
00414 {
00415   /* Call the Codec output device function */

00416   if(audio_drv->SetOutputMode(AUDIO_I2C_ADDR
ESS, Output) != 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  Updates the audio frequency.
00429   * @param  AudioFreq: Audio frequency used
to play the audio stream.
00430   * @note   This API should be called after
the BSP_AUDIO_OUT_Init() to adjust the
00431   *         audio frequency.
00432   */
00433 void BSP_AUDIO_OUT_SetFrequency(uint32_t Aud
ioFreq)
00434 {
00435   /* PLL clock is set depending by the Audio
Freq (44.1khz vs 48khz groups) */
00436   BSP_AUDIO_OUT_ClockConfig(&haudio_out_sai,
```

```
 AudioFreq, NULL);
00437
00438   /* Disable SAI peripheral to allow access
to SAI internal registers */
00439   __HAL_SAI_DISABLE(&haudio_out_sai);
00440
00441   /* Update the SAI audio frequency configur
ation */
00442   haudio_out_sai.Init.AudioFrequency = Audio
Freq;
00443   HAL_SAI_Init(&haudio_out_sai);
00444
00445   /* Enable SAI peripheral to generate MCLK
*/
00446   __HAL_SAI_ENABLE(&haudio_out_sai);
00447 }
00448
00449 /**
00450   * @brief  Updates the Audio frame slot con
figuration.
00451   * @param  AudioFrameSlot: specifies the au
dio Frame slot
00452   *         This parameter can be any value
of @ref CODEC_AudioFrame_SLOT_TDMMode
00453   * @note   This API should be called after
the BSP_AUDIO_OUT_Init() to adjust the
00454   *         audio frame slot.
00455   */
00456 void BSP_AUDIO_OUT_SetAudioFrameSlot(uint32_
t AudioFrameSlot)
00457 {
00458   /* Disable SAI peripheral to allow access
to SAI internal registers */
00459   __HAL_SAI_DISABLE(&haudio_out_sai);
00460
00461   /* Update the SAI audio frame slot configu
ration */
```

```
00462   haudio_out_sai.SlotInit.SlotActive = Audio
FrameSlot;
00463   HAL_SAI_Init(&haudio_out_sai);
00464
00465   /* Enable SAI peripheral to generate MCLK
*/
00466   __HAL_SAI_ENABLE(&haudio_out_sai);
00467 }
00468
00469 /**
00470   * @brief  Deinit the audio peripherals.
00471   */
00472 void BSP_AUDIO_OUT_DeInit(void)
00473 {
00474   SAIx_DeInit();
00475   /* DeInit the SAI MSP : this __weak functi
on can be rewritten by the applic */
00476   BSP_AUDIO_OUT_MspDeInit(&haudio_out_sai, N
ULL);
00477 }
00478
00479 /**
00480   * @brief  Tx Transfer completed callbacks.
00481   * @param  hsai: SAI handle
00482   */
00483 void HAL_SAI_TxCpltCallback(SAI_HandleTypeDe
f *hsai)
00484 {
00485   /* Manage the remaining file size and new
address offset: This function
00486      should be coded by user (its prototype
is already declared in stm32446e_eval_audio.h) */
00487   BSP_AUDIO_OUT_TransferComplete_CallBack();
00488 }
00489
00490 /**
00491   * @brief  Tx Half Transfer completed callb
```

```
acks.
00492   * @param  hsai: SAI handle
00493   */
00494 void HAL_SAI_TxHalfCpltCallback(SAI_HandleTy
peDef *hsai)
00495 {
00496   /* Manage the remaining file size and new
address offset: This function
00497      should be coded by user (its prototype
is already declared in stm32446e_eval_audio.h) */
00498   BSP_AUDIO_OUT_HalfTransfer_CallBack();
00499 }
00500
00501 /**
00502   * @brief  SAI error callbacks.
00503   * @param  hsai: SAI handle
00504   */
00505 void HAL_SAI_ErrorCallback(SAI_HandleTypeDef
 *hsai)
00506 {
00507   BSP_AUDIO_OUT_Error_CallBack();
00508 }
00509
00510 /**
00511   * @brief  Manages the DMA full Transfer co
mplete event.
00512   */
00513 __weak void BSP_AUDIO_OUT_TransferComplete_C
allBack(void)
00514 {
00515 }
00516
00517 /**
00518   * @brief  Manages the DMA Half Transfer co
mplete event.
00519   */
00520 __weak void BSP_AUDIO_OUT_HalfTransfer_CallB
```

```
ack(void)
00521 {
00522 }
00523
00524 /**
00525   * @brief  Manages the DMA FIFO error event.

00526   */
00527 __weak void BSP_AUDIO_OUT_Error_CallBack(void
)
00528 {
00529 }
00530
00531 /**
00532   * @brief  Initializes BSP_AUDIO_OUT MSP.
00533   * @param  hsai: SAI handle
00534   * @param  Params: pointer on additional co
nfiguration parameters, can be NULL.
00535   */
00536 __weak void BSP_AUDIO_OUT_MspInit(SAI_Handle
TypeDef *hsai, void *Params)
00537 {
00538   static DMA_HandleTypeDef hdma_sai_tx;
00539   GPIO_InitTypeDef  gpio_init_structure;
00540
00541   /* Enable SAI clock */
00542   AUDIO_SAIx_CLK_ENABLE();
00543
00544   /* Enable GPIO clock */
00545   AUDIO_SAIx_MCLK_SCK_ENABLE();
00546   AUDIO_SAIx_SD_FS_ENABLE();
00547
00548   /* CODEC_SAI pins configuration: FS, SCK,
MCK and SD pins ------------------*/
00549   gpio_init_structure.Mode = GPIO_MODE_AF_PP
;
00550   gpio_init_structure.Pull = GPIO_NOPULL;
```

```
00551   gpio_init_structure.Speed = GPIO_SPEED_HIG
H;
00552
00553   gpio_init_structure.Alternate = AUDIO_SAIx
_MCLK_SD_FS_AF;
00554   gpio_init_structure.Pin = AUDIO_SAIx_FS_PIN
 | AUDIO_SAIx_SD_PIN;
00555   HAL_GPIO_Init(AUDIO_SAIx_SD_FS_GPIO_PORT,
&gpio_init_structure);
00556
00557   gpio_init_structure.Alternate = AUDIO_SAIx
_MCLK_SD_FS_AF;
00558   gpio_init_structure.Pin = AUDIO_SAIx_MCK_P
IN;
00559   HAL_GPIO_Init(AUDIO_SAIx_MCLK_SCK_GPIO_PORT
, &gpio_init_structure);
00560
00561   gpio_init_structure.Alternate = AUDIO_SAIx
_SCK_AF;
00562   gpio_init_structure.Pin = AUDIO_SAIx_SCK_P
IN ;
00563   HAL_GPIO_Init(AUDIO_SAIx_MCLK_SCK_GPIO_PORT
, &gpio_init_structure);
00564
00565   /* Enable the DMA clock */
00566   AUDIO_SAIx_DMAx_CLK_ENABLE();
00567
00568   if(hsai->Instance == AUDIO_SAIx)
00569   {
00570     /* Configure the hdma_sai_tx handle para
meters */
00571     hdma_sai_tx.Init.Channel             = A
UDIO_SAIx_DMAx_CHANNEL;
00572     hdma_sai_tx.Init.Direction           = D
MA_MEMORY_TO_PERIPH;
00573     hdma_sai_tx.Init.PeriphInc           = D
MA_PINC_DISABLE;
```

```
00574     hdma_sai_tx.Init.MemInc                = D
MA_MINC_ENABLE;
00575     hdma_sai_tx.Init.PeriphDataAlignment = A
UDIO_SAIx_DMAx_PERIPH_DATA_SIZE;
00576     hdma_sai_tx.Init.MemDataAlignment    = A
UDIO_SAIx_DMAx_MEM_DATA_SIZE;
00577     hdma_sai_tx.Init.Mode                  = D
MA_CIRCULAR;
00578     hdma_sai_tx.Init.Priority              = D
MA_PRIORITY_HIGH;
00579     hdma_sai_tx.Init.FIFOMode              = D
MA_FIFOMODE_ENABLE;
00580     hdma_sai_tx.Init.FIFOThreshold         = D
MA_FIFO_THRESHOLD_FULL;
00581     hdma_sai_tx.Init.MemBurst              = D
MA_MBURST_SINGLE;
00582     hdma_sai_tx.Init.PeriphBurst           = D
MA_PBURST_SINGLE;
00583
00584     hdma_sai_tx.Instance = AUDIO_SAIx_DMAx_S
TREAM;
00585
00586     /* Associate the DMA handle */
00587     __HAL_LINKDMA(hsai, hdmatx, hdma_sai_tx)
;
00588
00589     /* Deinitialize the Stream for new trans
fer */
00590     HAL_DMA_DeInit(&hdma_sai_tx);
00591
00592     /* Configure the DMA Stream */
00593     HAL_DMA_Init(&hdma_sai_tx);
00594   }
00595
00596   /* SAI DMA IRQ Channel configuration */
00597   HAL_NVIC_SetPriority(AUDIO_SAIx_DMAx_IRQ,
AUDIO_OUT_IRQ_PREPRIO, 0);
```

```
00598   HAL_NVIC_EnableIRQ(AUDIO_SAIx_DMAx_IRQ);
00599 }
00600
00601 /**
00602   * @brief  Deinitializes SAI MSP.
00603   * @param  hsai: SAI handle
00604   * @param  Params: pointer on additional co
nfiguration parameters, can be NULL.
00605   */
00606 __weak void BSP_AUDIO_OUT_MspDeInit(SAI_Hand
leTypeDef *hsai, void *Params)
00607 {
00608     GPIO_InitTypeDef  gpio_init_structure;
00609
00610     /* SAI DMA IRQ Channel deactivation */
00611     HAL_NVIC_DisableIRQ(AUDIO_SAIx_DMAx_IRQ)
;
00612
00613     if(hsai->Instance == AUDIO_SAIx)
00614     {
00615       /* Deinitialize the DMA stream */
00616       HAL_DMA_DeInit(hsai->hdmatx);
00617     }
00618
00619     /* Disable SAI peripheral */
00620     __HAL_SAI_DISABLE(hsai);
00621
00622     /* Deactives CODEC_SAI pins FS, SCK, MCK
 and SD by putting them in input mode */
00623     gpio_init_structure.Pin = AUDIO_SAIx_FS_
PIN | AUDIO_SAIx_SD_PIN;
00624     HAL_GPIO_DeInit(AUDIO_SAIx_SD_FS_GPIO_PO
RT, gpio_init_structure.Pin);
00625
00626     gpio_init_structure.Pin = AUDIO_SAIx_MCK
_PIN;
00627     HAL_GPIO_DeInit(AUDIO_SAIx_MCLK_SCK_GPIO
```

```
_PORT, gpio_init_structure.Pin);
00628
00629     gpio_init_structure.Pin = AUDIO_SAIx_SCK
_PIN ;
00630     HAL_GPIO_DeInit(AUDIO_SAIx_MCLK_SCK_GPIO
_PORT, gpio_init_structure.Pin);
00631
00632     /* Disable SAI clock */
00633     AUDIO_SAIx_CLK_DISABLE();
00634
00635     /* GPIO pins clock and DMA clock can be
shut down in the application
00636        by surcharging this __weak function */

00637 }
00638
00639 /**
00640   * @brief  Clock Config.
00641   * @param  hsai: might be required to set a
udio peripheral predivider if any.
00642   * @param  AudioFreq: Audio frequency used
to play the audio stream.
00643   * @param  Params: pointer on additional co
nfiguration parameters, can be NULL.
00644   * @note   This API is called by BSP_AUDIO_
OUT_Init() and BSP_AUDIO_OUT_SetFrequency()
00645   *         Being __weak it can be overwritt
en by the application
00646   */
00647 __weak void BSP_AUDIO_OUT_ClockConfig(SAI_Ha
ndleTypeDef *hsai, uint32_t AudioFreq, void *Param
s)
00648 {
00649   RCC_PeriphCLKInitTypeDef rcc_ex_clk_init_s
truct;
00650
00651   HAL_RCCEx_GetPeriphCLKConfig(&rcc_ex_clk_i
```

```
nit_struct);
00652
00653   /* Set the PLL configuration according to
the audio frequency */
00654   if((AudioFreq == AUDIO_FREQUENCY_11K) || (
AudioFreq == AUDIO_FREQUENCY_22K) || (AudioFreq ==
 AUDIO_FREQUENCY_44K))
00655   {
00656     /* Configure PLLSAI prescalers */
00657     /* PLLSAI_VCO: VCO_429M
00658     SAI_CLK(first level) = PLLSAI_VCO/PLLSAI
Q = 429/2 = 214.5 Mhz
00659     SAI_CLK_x = SAI_CLK(first level)/PLLSAID
IVQ = 214.5/19 = 11.289 Mhz */
00660     rcc_ex_clk_init_struct.PeriphClockSelect
ion = RCC_PERIPHCLK_SAI2;
00661     rcc_ex_clk_init_struct.Sai2ClockSelectio
n = RCC_SAI2CLKSOURCE_PLLSAI;
00662     rcc_ex_clk_init_struct.PLLSAI.PLLSAIM =
8;
00663     rcc_ex_clk_init_struct.PLLSAI.PLLSAIN =
429;
00664     rcc_ex_clk_init_struct.PLLSAI.PLLSAIQ =
2;
00665     rcc_ex_clk_init_struct.PLLSAIDivQ = 19;
00666
00667     HAL_RCCEx_PeriphCLKConfig(&rcc_ex_clk_in
it_struct);
00668
00669   }
00670   else /* AUDIO_FREQUENCY_8K, AUDIO_FREQUENC
Y_16K, AUDIO_FREQUENCY_48K), AUDIO_FREQUENCY_96K */

00671   {
00672     /* SAI clock config
00673     PLLSAI_VCO: VCO_344M
00674     SAI_CLK(first level) = PLLSAI_VCO/PLLSAI
```

```
Q = 344/7 = 49.142 Mhz
00675     SAI_CLK_x = SAI_CLK(first level)/PLLSAID
IVQ = 49.142/1 = 49.142 Mhz */
00676     rcc_ex_clk_init_struct.PeriphClockSelect
ion = RCC_PERIPHCLK_SAI2;
00677     rcc_ex_clk_init_struct.Sai2ClockSelectio
n = RCC_SAI2CLKSOURCE_PLLSAI;
00678     rcc_ex_clk_init_struct.PLLSAI.PLLSAIM =
8;
00679     rcc_ex_clk_init_struct.PLLSAI.PLLSAIN =
344;
00680     rcc_ex_clk_init_struct.PLLSAI.PLLSAIQ =
7;
00681     rcc_ex_clk_init_struct.PLLSAIDivQ = 1;

00682
00683     HAL_RCCEx_PeriphCLKConfig(&rcc_ex_clk_in
it_struct);
00684   }
00685 }
00686
00687 /**********************************************
**********************************
00688                                   Static Functions
00689 **********************************************
**********************************/
00690
00691 /**
00692   * @brief  Initializes the Audio Codec audi
o interface (SAI).
00693   * @param  AudioFreq: Audio frequency to be
 configured for the SAI peripheral.
00694   * @note   The default SlotActive configura
tion is set to CODEC_AUDIOFRAME_SLOT_0123
00695   *         and user can update this configu
ration using
00696   */
```

```
00697 static void SAIx_Init(uint32_t AudioFreq)
00698 {
00699   /* Initialize the hAudioOutSai Instance pa
rameter */
00700   haudio_out_sai.Instance = AUDIO_SAIx;
00701
00702   /* Disable SAI peripheral to allow access
to SAI internal registers */
00703   __HAL_SAI_DISABLE(&haudio_out_sai);
00704
00705   /* Configure SAI_Block_x
00706   LSBFirst: Disabled
00707   DataSize: 16 */
00708   haudio_out_sai.Init.AudioFrequency = Audio
Freq;
00709   haudio_out_sai.Init.ClockSource = SAI_CLKS
OURCE_PLLSAI;
00710   haudio_out_sai.Init.AudioMode = SAI_MODEMA
STER_TX;
00711   haudio_out_sai.Init.NoDivider = SAI_MASTER
DIVIDER_ENABLE;
00712   haudio_out_sai.Init.Protocol = SAI_FREE_PR
OTOCOL;
00713   haudio_out_sai.Init.DataSize = SAI_DATASIZ
E_16;
00714   haudio_out_sai.Init.FirstBit = SAI_FIRSTBI
T_MSB;
00715   haudio_out_sai.Init.ClockStrobing = SAI_CL
OCKSTROBING_RISINGEDGE;
00716   haudio_out_sai.Init.Synchro = SAI_ASYNCHRO
NOUS;
00717   haudio_out_sai.Init.OutputDrive = SAI_OUTP
UTDRIVE_ENABLE;
00718   haudio_out_sai.Init.FIFOThreshold = SAI_FI
FOTHRESHOLD_1QF;
00719
00720   /* Configure SAI_Block_x Frame
```

```
00721   Frame Length: 64
00722   Frame active Length: 32
00723   FS Definition: Start frame + Channel Side
identification
00724   FS Polarity: FS active Low
00725   FS Offset: FS asserted one bit before the
first bit of slot 0 */
00726   haudio_out_sai.FrameInit.FrameLength = 64;

00727   haudio_out_sai.FrameInit.ActiveFrameLength
 = 32;
00728   haudio_out_sai.FrameInit.FSDefinition = SA
I_FS_CHANNEL_IDENTIFICATION;
00729   haudio_out_sai.FrameInit.FSPolarity = SAI_
FS_ACTIVE_LOW;
00730   haudio_out_sai.FrameInit.FSOffset = SAI_FS
_BEFOREFIRSTBIT;
00731
00732   /* Configure SAI Block_x Slot
00733   Slot First Bit Offset: 0
00734   Slot Size  : 16
00735   Slot Number: 4
00736   Slot Active: All slot actives */
00737   haudio_out_sai.SlotInit.FirstBitOffset = 0
;
00738   haudio_out_sai.SlotInit.SlotSize = SAI_SLO
TSIZE_DATASIZE;
00739   haudio_out_sai.SlotInit.SlotNumber = 4;
00740   haudio_out_sai.SlotInit.SlotActive = CODEC
_AUDIOFRAME_SLOT_0123;
00741
00742   HAL_SAI_Init(&haudio_out_sai);
00743
00744   /* Enable SAI peripheral to generate MCLK
*/
00745   __HAL_SAI_ENABLE(&haudio_out_sai);
00746 }
```

```
00747
00748
00749
00750 /**
00751   * @brief  Deinitializes the Audio Codec au
dio interface (SAI).
00752   */
00753 static void SAIx_DeInit(void)
00754 {
00755   /* Initialize the hAudioOutSai Instance pa
rameter */
00756   haudio_out_sai.Instance = AUDIO_SAIx;
00757
00758   /* Disable SAI peripheral */
00759   __HAL_SAI_DISABLE(&haudio_out_sai);
00760
00761   HAL_SAI_DeInit(&haudio_out_sai);
00762 }
00763
00764
00765 /**
00766   * @}
00767   */
00768
00769 /** @defgroup STM32446E_EVAL_AUDIO_in_Privat
e_Functions STM32446E EVAL AUDIO IN Private Functi
ons
00770   * @{
00771   */
00772
00773 /**
00774   * @brief  Initializes wave recording.
00775   * @note   This function assumes that the I
2S input clock (through PLL_R in
00776   *         Devices RevA/Z and through dedic
ated PLLI2S_R in Devices RevB/Y)
00777   *         is already configured and ready
```

```
to be used.
00778   * @param  AudioFreq: Audio frequency to be
 configured for the I2S peripheral.
00779   * @param  BitRes: Audio frequency to be co
nfigured for the I2S peripheral.
00780   * @param  ChnlNbr: Audio frequency to be c
onfigured for the I2S peripheral.
00781   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00782   */
00783 uint8_t BSP_AUDIO_IN_Init(uint32_t AudioFreq
, uint32_t BitRes, uint32_t ChnlNbr)
00784 {
00785   RCC_PeriphCLKInitTypeDef rcc_ex_clk_init_s
truct;
00786
00787   I2Sx_DeInit();
00788
00789   HAL_RCCEx_GetPeriphCLKConfig(&rcc_ex_clk_i
nit_struct);
00790   rcc_ex_clk_init_struct.PeriphClockSelectio
n = RCC_PERIPHCLK_I2S_APB1; /*SPI3 & TIM4 on APB1*/

00791   rcc_ex_clk_init_struct.PLLI2S.PLLI2SN = 38
4;
00792   rcc_ex_clk_init_struct.PLLI2S.PLLI2SR = 2;
00793   HAL_RCCEx_PeriphCLKConfig(&rcc_ex_clk_init
_struct);
00794
00795   /* Configure the PDM library */
00796   PDMDecoder_Init(AudioFreq, ChnlNbr);
00797
00798   /* Configure the I2S peripheral */
00799   haudio_in_i2s.Instance = AUDIO_I2Sx;
00800   if(HAL_I2S_GetState(&haudio_in_i2s) == HAL
_I2S_STATE_RESET)
00801   {
```

```
00802     /* Initialize the I2S Msp: this __weak f
unction can be rewritten by the application */
00803     BSP_AUDIO_IN_MspInit(&haudio_in_i2s, NUL
L);
00804   }
00805   I2Sx_Init(AudioFreq);
00806
00807   /* Return AUDIO_OK when all operations are
 correctly done */
00808   return AUDIO_OK;
00809 }
00810
00811 /**
00812   * @brief  Starts audio recording.
00813   * @param  pbuf: Main buffer pointer for th
e recorded data storing
00814   * @param  size: Current size of the record
ed buffer
00815   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00816   */
00817 uint8_t BSP_AUDIO_IN_Record(uint16_t* pbuf,
uint32_t size)
00818 {
00819   uint32_t ret = AUDIO_ERROR;
00820
00821   /* Start the process receive DMA */
00822   HAL_I2S_Receive_DMA(&haudio_in_i2s, pbuf,
size);
00823
00824   /* Return AUDIO_OK when all operations are
 correctly done */
00825   ret = AUDIO_OK;
00826
00827   return ret;
00828 }
00829
```

```
00830 /**
00831   * @brief  Stops audio recording.
00832   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00833   */
00834 uint8_t BSP_AUDIO_IN_Stop(void)
00835 {
00836   uint32_t ret = AUDIO_ERROR;
00837
00838   /* Call the Media layer pause function */
00839   HAL_I2S_DMAPause(&haudio_in_i2s);
00840
00841   /* TIMx Peripheral clock disable */
00842   AUDIO_TIMx_CLK_DISABLE();
00843
00844   /* Return AUDIO_OK when all operations are
 correctly done */
00845   ret = AUDIO_OK;
00846
00847   return ret;
00848 }
00849
00850 /**
00851   * @brief  Pauses the audio file stream.
00852   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00853   */
00854 uint8_t BSP_AUDIO_IN_Pause(void)
00855 {
00856   /* Call the Media layer pause function */
00857   HAL_I2S_DMAPause(&haudio_in_i2s);
00858
00859   /* Return AUDIO_OK when all operations are
 correctly done */
00860   return AUDIO_OK;
00861 }
00862
```

```
00863 /**
00864   * @brief  Resumes the audio file stream.

00865   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00866   */
00867 uint8_t BSP_AUDIO_IN_Resume(void)
00868 {
00869   /* Call the Media layer pause/resume funct
ion */
00870   HAL_I2S_DMAResume(&haudio_in_i2s);
00871
00872   /* Return AUDIO_OK when all operations are
 correctly done */
00873   return AUDIO_OK;
00874 }
00875
00876 /**
00877   * @brief  Controls the audio in volume lev
el.
00878   * @param  Volume: Volume level to be set i
n percentage from 0% to 100% (0 for
00879   *         Mute and 100 for Max volume leve
l).
00880   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00881   */
00882 uint8_t BSP_AUDIO_IN_SetVolume(uint8_t Volum
e)
00883 {
00884   /* Set the Global variable AudioInVolume
*/
00885   AudioInVolume = Volume;
00886
00887   /* Return AUDIO_OK when all operations are
 correctly done */
00888   return AUDIO_OK;
```

```
00889 }
00890
00891 /**
00892   * @brief  Deinit the audio IN peripherals.
00893   */
00894 void BSP_AUDIO_IN_DeInit(void)
00895 {
00896   I2Sx_DeInit();
00897     /* DeInit the I2S MSP : this __weak func
tion can be rewritten by the applic */
00898   BSP_AUDIO_IN_MspDeInit(&haudio_in_i2s, NUL
L);
00899   TIMx_DeInit();
00900 }
00901
00902 /**
00903   * @brief  Converts audio format from PDM t
o PCM.
00904   * @param  PDMBuf: Pointer to data PDM buff
er
00905   * @param  PCMBuf: Pointer to data PCM buff
er
00906   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00907   */
00908 uint8_t BSP_AUDIO_IN_PDMToPCM(uint16_t* PDMB
uf, uint16_t* PCMBuf)
00909 {
00910   uint8_t app_pdm[INTERNAL_BUFF_SIZE*2];
00911   uint8_t byte1 = 0, byte2 = 0;
00912   uint32_t index = 0;
00913
00914   /* PDM Demux */
00915   for(index = 0; index<INTERNAL_BUFF_SIZE/2;
 index++)
00916   {
00917     byte2 = (PDMBuf[index] >> 8)& 0xFF;
```

```
00918     byte1 = (PDMBuf[index] & 0xFF);
00919     app_pdm[(index*2)+1] = Channel_Demux[byt
e1 & CHANNEL_DEMUX_MASK] | Channel_Demux[byte2 & C
HANNEL_DEMUX_MASK] << 4;
00920     app_pdm[(index*2)] = Channel_Demux[(byte
1 >> 1) & CHANNEL_DEMUX_MASK] | Channel_Demux[(byt
e2 >> 1) & CHANNEL_DEMUX_MASK] << 4;
00921   }
00922
00923   for(index = 0; index < DEFAULT_AUDIO_IN_CH
ANNEL_NBR; index++)
00924   {
00925     /* PDM to PCM filter */
00926     PDM_Filter_64_LSB((uint8_t*)&app_pdm[ind
ex], (uint16_t*)&(PCMBuf[index]), AudioInVolume ,
(PDMFilter_InitStruct *)&Filter[index]);
00927   }
00928
00929   /* Return AUDIO_OK when all operations are
 correctly done */
00930   return AUDIO_OK;
00931 }
00932
00933  /**
00934   * @brief  Rx Transfer completed callbacks.
00935   * @param  hi2s: I2S handle
00936   */
00937 void HAL_I2S_RxCpltCallback(I2S_HandleTypeDe
f *hi2s)
00938 {
00939   /* Call the record update function to get
the next buffer to fill and its size (size is igno
red) */
00940   BSP_AUDIO_IN_TransferComplete_CallBack();
00941 }
00942
00943 /**
```

```
00944   * @brief  Rx Half Transfer completed callb
acks.
00945   * @param  hi2s: I2S handle
00946   */
00947 void HAL_I2S_RxHalfCpltCallback(I2S_HandleTy
peDef *hi2s)
00948 {
00949   /* Manage the remaining file size and new
address offset: This function
00950      should be coded by user (its prototype
is already declared in stm32446e_eval_audio.h) */
00951   BSP_AUDIO_IN_HalfTransfer_CallBack();
00952 }
00953
00954 /**
00955   * @brief  I2S error callbacks.
00956   * @param  hi2s: I2S handle
00957   */
00958 void HAL_I2S_ErrorCallback(I2S_HandleTypeDef
 *hi2s)
00959 {
00960   /* Manage the error generated on DMA FIFO:
 This function
00961      should be coded by user (its prototype
is already declared in stm32446e_eval_audio.h) */
00962   BSP_AUDIO_IN_Error_Callback();
00963 }
00964
00965 /**
00966   * @brief  User callback when record buffer
 is filled.
00967   */
00968 __weak void BSP_AUDIO_IN_TransferComplete_Ca
llBack(void)
00969 {
00970   /* This function should be implemented by
the user application.
```

```
00971      It is called into this driver when the
current buffer is filled
00972      to prepare the next buffer pointer and
its size. */
00973 }
00974
00975 /**
00976   * @brief  Manages the DMA Half Transfer co
mplete event.
00977   */
00978 __weak void BSP_AUDIO_IN_HalfTransfer_CallBa
ck(void)
00979 {
00980   /* This function should be implemented by
the user application.
00981      It is called into this driver when the
current buffer is filled
00982      to prepare the next buffer pointer and
its size. */
00983 }
00984
00985 /**
00986   * @brief  Audio IN Error callback function.

00987   */
00988 __weak void BSP_AUDIO_IN_Error_Callback(void
)
00989 {
00990   /* This function is called when an Interru
pt due to transfer error on or peripheral
00991      error occurs. */
00992 }
00993
00994 /**
00995   * @brief  Initializes BSP_AUDIO_IN MSP.
00996   * @param  hi2s: I2S handle
00997   * @param  Params: pointer on additional co
```

```
nfiguration parameters, can be NULL.
00998   */
00999 __weak void BSP_AUDIO_IN_MspInit(I2S_HandleT
ypeDef *hi2s, void *Params)
01000 {
01001   static DMA_HandleTypeDef hdma_i2s_rx;
01002   GPIO_InitTypeDef  gpio_init_structure;
01003
01004   /* Configure the Timer which clocks the ME
MS */
01005   /* Moved inside MSP to allow applic to red
efine the TIMx_MspInit */
01006   TIMx_Init();
01007
01008   /* Enable I2S clock */
01009   AUDIO_I2Sx_CLK_ENABLE();
01010
01011   /* Enable SCK and SD GPIO clock */
01012   AUDIO_I2Sx_SD_GPIO_CLK_ENABLE();
01013   AUDIO_I2Sx_SCK_GPIO_CLK_ENABLE();
01014   /* CODEC_I2S pins configuration: SCK and S
D pins */
01015   gpio_init_structure.Pin = AUDIO_I2Sx_SCK_P
IN;
01016   gpio_init_structure.Mode = GPIO_MODE_AF_PP
;
01017   gpio_init_structure.Pull = GPIO_NOPULL;
01018   gpio_init_structure.Speed = GPIO_SPEED_FAS
T;
01019   gpio_init_structure.Alternate = AUDIO_I2Sx
_SCK_AF;
01020   HAL_GPIO_Init(AUDIO_I2Sx_SCK_GPIO_PORT, &g
pio_init_structure);
01021
01022   gpio_init_structure.Pin = AUDIO_I2Sx_SD_PIN
;
01023   gpio_init_structure.Alternate = AUDIO_I2Sx
```

```
_SD_AF;
01024   HAL_GPIO_Init(AUDIO_I2Sx_SD_GPIO_PORT, &gp
io_init_structure);
01025
01026   /* Enable the DMA clock */
01027   AUDIO_I2Sx_DMAx_CLK_ENABLE();
01028
01029   if(hi2s->Instance == AUDIO_I2Sx)
01030   {
01031     /* Configure the hdma_i2s_rx handle para
meters */
01032     hdma_i2s_rx.Init.Channel             = A
UDIO_I2Sx_DMAx_CHANNEL;
01033     hdma_i2s_rx.Init.Direction           = D
MA_PERIPH_TO_MEMORY;
01034     hdma_i2s_rx.Init.PeriphInc           = D
MA_PINC_DISABLE;
01035     hdma_i2s_rx.Init.MemInc              = D
MA_MINC_ENABLE;
01036     hdma_i2s_rx.Init.PeriphDataAlignment = A
UDIO_I2Sx_DMAx_PERIPH_DATA_SIZE;
01037     hdma_i2s_rx.Init.MemDataAlignment    = A
UDIO_I2Sx_DMAx_MEM_DATA_SIZE;
01038     hdma_i2s_rx.Init.Mode                = D
MA_CIRCULAR;
01039     hdma_i2s_rx.Init.Priority            = D
MA_PRIORITY_HIGH;
01040     hdma_i2s_rx.Init.FIFOMode            = D
MA_FIFOMODE_DISABLE;
01041     hdma_i2s_rx.Init.FIFOThreshold       = D
MA_FIFO_THRESHOLD_FULL;
01042     hdma_i2s_rx.Init.MemBurst            = D
MA_MBURST_SINGLE;
01043     hdma_i2s_rx.Init.PeriphBurst         = D
MA_MBURST_SINGLE;
01044
01045     hdma_i2s_rx.Instance = AUDIO_I2Sx_DMAx_S
```

```
TREAM;
01046
01047     /* Associate the DMA handle */
01048     __HAL_LINKDMA(hi2s, hdmarx, hdma_i2s_rx)
;
01049
01050     /* Deinitialize the Stream for new trans
fer */
01051     HAL_DMA_DeInit(&hdma_i2s_rx);
01052
01053     /* Configure the DMA Stream */
01054     HAL_DMA_Init(&hdma_i2s_rx);
01055   }
01056
01057   /* I2S DMA IRQ Channel configuration */
01058   HAL_NVIC_SetPriority(AUDIO_I2Sx_DMAx_IRQ,
AUDIO_IN_IRQ_PREPRIO, 0);
01059   HAL_NVIC_EnableIRQ(AUDIO_I2Sx_DMAx_IRQ);
01060 }
01061
01062 /**
01063   * @brief  DeInitializes BSP_AUDIO_IN MSP.
01064   * @param  hi2s: I2S handle
01065   * @param  Params: pointer on additional co
nfiguration parameters, can be NULL.
01066   */
01067 __weak void BSP_AUDIO_IN_MspDeInit(I2S_Handl
eTypeDef *hi2s, void *Params)
01068 {
01069     GPIO_InitTypeDef  gpio_init_structure;
01070
01071     static DMA_HandleTypeDef hdma_i2s_rx;
01072
01073     /* I2S DMA IRQ Channel deactivation */
01074     HAL_NVIC_DisableIRQ(AUDIO_I2Sx_DMAx_IRQ)
;
01075
```

```
    if(hi2s->Instance == AUDIO_I2Sx)
    {
      /* Deinitialize the Stream for new transfer */
      HAL_DMA_DeInit(&hdma_i2s_rx);
    }

   /* Disable I2S block */
    __HAL_I2S_DISABLE(hi2s);

    /* Disable pins: SCK and SD pins */
    gpio_init_structure.Pin = AUDIO_I2Sx_SCK_PIN;
    HAL_GPIO_DeInit(AUDIO_I2Sx_SCK_GPIO_PORT, gpio_init_structure.Pin);
    gpio_init_structure.Pin = AUDIO_I2Sx_SD_PIN;
    HAL_GPIO_DeInit(AUDIO_I2Sx_SD_GPIO_PORT, gpio_init_structure.Pin);

    /* Disable I2S clock */
    AUDIO_I2Sx_CLK_DISABLE();

    /* GPIO pins clock and DMA clock can be shut down in the applic
       by surcgarging this __weak function */

}


/*****************************************************************************
                                Static Functions
*****************************************************************************/

/**
```

```
01104   * @brief  Initializes the PDM library.
01105   * @param  AudioFreq: Audio sampling freque
ncy
01106   * @param  ChnlNbr: Number of audio channel
s (1: mono; 2: stereo)
01107   */
01108 static void PDMDecoder_Init(uint32_t AudioFr
eq, uint32_t ChnlNbr)
01109 {
01110   uint32_t i = 0;
01111
01112   /* Enable CRC peripheral to unlock the PDM
 library */
01113   __HAL_RCC_CRC_CLK_ENABLE();
01114
01115   for(i = 0; i < ChnlNbr; i++)
01116   {
01117     /* Filter LP & HP Init */
01118     Filter[i].LP_HZ = AudioFreq/2;
01119     Filter[i].HP_HZ = 10;
01120     Filter[i].Fs = AudioFreq;
01121     Filter[i].Out_MicChannels = ChnlNbr;
01122     Filter[i].In_MicChannels = ChnlNbr;
01123     PDM_Filter_Init((PDMFilter_InitStruct *)&
Filter[i]);
01124   }
01125 }
01126
01127 /**
01128   * @brief  Initializes the Audio Codec audi
o interface (I2S)
01129   * @note   This function assumes that the I
2S input clock (through PLL_R in
01130   *         Devices RevA/Z and through dedic
ated PLLI2S_R in Devices RevB/Y)
01131   *         is already configured and ready
to be used.
```

```
01132   * @param  AudioFreq: Audio frequency to be
 configured for the I2S peripheral.
01133   */
01134 static void I2Sx_Init(uint32_t AudioFreq)
01135 {
01136   /* Initialize the hAudioInI2s Instance par
ameter */
01137   haudio_in_i2s.Instance = AUDIO_I2Sx;
01138
01139  /* Disable I2S block */
01140   __HAL_I2S_DISABLE(&haudio_in_i2s);
01141
01142   /* I2S2 peripheral configuration */
01143   haudio_in_i2s.Init.AudioFreq = 4 * AudioFr
eq;
01144   haudio_in_i2s.Init.ClockSource = I2S_CLOCK
_PLL;
01145   haudio_in_i2s.Init.CPOL = I2S_CPOL_HIGH;
01146   haudio_in_i2s.Init.DataFormat = I2S_DATAFO
RMAT_16B;
01147   haudio_in_i2s.Init.MCLKOutput = I2S_MCLKOU
TPUT_DISABLE;
01148   haudio_in_i2s.Init.Mode = I2S_MODE_MASTER_
RX;
01149   haudio_in_i2s.Init.Standard = I2S_STANDARD
_LSB;
01150
01151   /* Init the I2S */
01152   HAL_I2S_Init(&haudio_in_i2s);
01153
01154  /* Disable I2S block */
01155   __HAL_I2S_ENABLE(&haudio_in_i2s);
01156
01157 }
01158
01159 /**
01160   * @brief  Deinitializes the Audio Codec au
```

```
dio interface (I2S).
01161   */
01162 static void I2Sx_DeInit(void)
01163 {
01164   /* Initialize the hAudioInI2s Instance par
ameter */
01165   haudio_in_i2s.Instance = AUDIO_I2Sx;
01166
01167  /* Disable I2S block */
01168   __HAL_I2S_DISABLE(&haudio_in_i2s);
01169
01170   /* DeInit the I2S */
01171   HAL_I2S_DeInit(&haudio_in_i2s);
01172 }
01173
01174
01175 /**
01176   * @brief  Initializes the TIM INput Captur
e MSP.
01177   * @param  htim: TIM handle
01178   */
01179 static void TIMx_IC_MspInit(TIM_HandleTypeDe
f *htim)
01180 {
01181   GPIO_InitTypeDef   gpio_init_structure;
01182
01183   /* Enable peripherals and GPIO Clocks ----
-----------------------------------*/
01184   /* TIMx Peripheral clock enable */
01185   AUDIO_TIMx_CLK_ENABLE();
01186
01187   /* Enable GPIO Channels Clock */
01188   AUDIO_TIMx_GPIO_CLK_ENABLE();
01189
01190   /* Configure I/Os ------------------------
-----------------------------------*/
01191   /* Common configuration for all channels */
```

```
01192   gpio_init_structure.Mode = GPIO_MODE_AF_PP
;
01193   gpio_init_structure.Pull = GPIO_NOPULL;
01194   gpio_init_structure.Speed = GPIO_SPEED_HIG
H;
01195   gpio_init_structure.Alternate = AUDIO_TIMx
_AF;
01196
01197   /* Configure TIM input channel */
01198   gpio_init_structure.Pin = AUDIO_TIMx_IN_GP
IO_PIN;
01199   HAL_GPIO_Init(AUDIO_TIMx_GPIO_PORT, &gpio_
init_structure);
01200
01201   /* Configure TIM output channel */
01202   gpio_init_structure.Pin = AUDIO_TIMx_OUT_G
PIO_PIN;
01203   HAL_GPIO_Init(AUDIO_TIMx_GPIO_PORT, &gpio_
init_structure);
01204
01205 }
01206
01207 /**
01208   * @brief  Initializes the TIM INput Captur
e MSP.
01209   * @param  htim: TIM handle
01210   */
01211 static void TIMx_IC_MspDeInit(TIM_HandleType
Def *htim)
01212 {
01213     /* Disable TIMx Peripheral clock  */
01214     AUDIO_TIMx_CLK_DISABLE();
01215
01216     /* GPIO pins clock and DMA clock can be
shut down in the applic
01217        by surcgarging this __weak function */
```

```
01218 }
01219
01220 /**
01221   * @brief  Configure TIM as a clock divider
 by 2.
01222   *         I2S_SCK is externally connected
to TIMx input channel
01223   */
01224 static void TIMx_Init(void)
01225 {
01226   TIM_IC_InitTypeDef     s_ic_config;
01227   TIM_OC_InitTypeDef     s_oc_config;
01228   TIM_ClockConfigTypeDef s_clk_source_config
;
01229   TIM_SlaveConfigTypeDef s_slave_config;
01230
01231   /* Configure the TIM peripheral ----------
-----------------------------------*/
01232   /* Set TIMx instance */
01233   haudio_tim.Instance = AUDIO_TIMx;
01234   /* Timer Input Capture Configuration Struc
ture declaration */
01235    /* Initialize TIMx peripheral as follow:
01236        + Period = 0xFFFF
01237        + Prescaler = 0
01238        + ClockDivision = 0
01239        + Counter direction = Up
01240   */
01241   haudio_tim.Init.Period        = 1;
01242   haudio_tim.Init.Prescaler     = 0;
01243   haudio_tim.Init.ClockDivision = 0;
01244   haudio_tim.Init.CounterMode   = TIM_COUNTE
RMODE_UP;
01245
01246   /* Initialize the TIMx peripheral with the
 structure above */
```

```
01247   TIMx_IC_MspInit(&haudio_tim);
01248   HAL_TIM_IC_Init(&haudio_tim);
01249
01250   /* Configure the Input Capture channel ---
----------------------------------*/
01251   /* Configure the Input Capture of channel
2 */
01252   s_ic_config.ICPolarity  = TIM_ICPOLARITY_F
ALLING;
01253   s_ic_config.ICSelection = TIM_ICSELECTION_
DIRECTTI;
01254   s_ic_config.ICPrescaler = TIM_ICPSC_DIV1;
01255   s_ic_config.ICFilter    = 0;
01256   HAL_TIM_IC_ConfigChannel(&haudio_tim, &s_i
c_config, AUDIO_TIMx_IN_CHANNEL);
01257
01258   /* Select external clock mode 1 */
01259   s_clk_source_config.ClockSource = TIM_CLOC
KSOURCE_ETRMODE1;
01260   s_clk_source_config.ClockPolarity = TIM_CL
OCKPOLARITY_NONINVERTED;
01261   s_clk_source_config.ClockPrescaler = TIM_C
LOCKPRESCALER_DIV1;
01262   s_clk_source_config.ClockFilter = 0;
01263   HAL_TIM_ConfigClockSource(&haudio_tim, &s_
clk_source_config);
01264
01265   /* Select Input Channel as input trigger */

01266   s_slave_config.InputTrigger = TIM_TS_TI1FP
1;
01267   s_slave_config.SlaveMode = TIM_SLAVEMODE_E
XTERNAL1;
01268   s_slave_config.TriggerPolarity = TIM_TRIGG
ERPOLARITY_NONINVERTED;
01269   s_slave_config.TriggerPrescaler = TIM_CLOC
KPRESCALER_DIV1;
```

```
01270   s_slave_config.TriggerFilter = 0;
01271   HAL_TIM_SlaveConfigSynchronization(&haudio
_tim, &s_slave_config);
01272
01273   /* Output Compare PWM Mode configuration:
Channel2 */
01274   s_oc_config.OCMode = TIM_OCMODE_PWM1;
01275   s_oc_config.OCIdleState = TIM_OCIDLESTATE_
SET;
01276   s_oc_config.Pulse = 1;
01277   s_oc_config.OCPolarity = TIM_OCPOLARITY_HI
GH;
01278   s_oc_config.OCNPolarity = TIM_OCNPOLARITY_
HIGH;
01279   s_oc_config.OCFastMode = TIM_OCFAST_DISABL
E;
01280   s_oc_config.OCNIdleState = TIM_OCNIDLESTAT
E_SET;
01281
01282   /* Initialize the TIM4 Channel2 with the s
tructure above */
01283   HAL_TIM_PWM_ConfigChannel(&haudio_tim, &s_
oc_config, AUDIO_TIMx_OUT_CHANNEL);
01284
01285   /* Start the TIM4 Channel2 */
01286   HAL_TIM_PWM_Start(&haudio_tim, AUDIO_TIMx_
OUT_CHANNEL);
01287
01288   /* Start the TIM4 Channel1 */
01289   HAL_TIM_IC_Start(&haudio_tim, AUDIO_TIMx_I
N_CHANNEL);
01290 }
01291
01292 /**
01293   * @brief  Configure TIM as a clock divider
 by 2.
01294   *         I2S_SCK is externally connected
```

```
to TIMx input channel
01295   */
01296 static void TIMx_DeInit(void)
01297 {
01298   haudio_tim.Instance = AUDIO_TIMx;
01299
01300   /* Stop the TIM4 Channel2 */
01301   HAL_TIM_PWM_Stop(&haudio_tim, AUDIO_TIMx_O
UT_CHANNEL);
01302   /* Stop the TIM4 Channel1 */
01303   HAL_TIM_IC_Stop(&haudio_tim, AUDIO_TIMx_IN
_CHANNEL);
01304
01305   HAL_TIM_IC_DeInit(&haudio_tim);
01306
01307   /* Initialize the TIMx peripheral with the
 structure above */
01308   TIMx_IC_MspDeInit(&haudio_tim);
01309 }
01310
01311 /**
01312   * @}
01313   */
01314
01315 /**
01316   * @}
01317   */
01318
01319 /**
01320   * @}
01321   */
01322
01323 /**
01324   * @}
01325   */
01326
01327 /************************ (C) COPYRIGHT STMi
```

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

STM32446E_EVAL BSP User Manual

stm32446e_eval_camera.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval_camera.h
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm32446e_eval_camera.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 __STM32446E_EVAL_CAMERA_H
00041 #define __STM32446E_EVAL_CAMERA_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------
-----------------------------------*/
00048 /* Include Camera component Driver */
00049 #include "../Components/s5k5cag/s5k5cag.h"
00050
00051 /* Include IO Driver */
00052 #include "stm32446e_eval_io.h"
00053
00054 /** @addtogroup BSP
00055   * @{
00056   */
00057
00058 /** @addtogroup STM32446E_EVAL
00059   * @{
00060   */
00061
00062 /** @addtogroup STM32446E_EVAL_CAMERA
00063   * @{
00064   */
00065
00066 /** @defgroup STM32446E_EVAL_CAMERA_Exported
_Types STM32446E EVAL CAMERA Exported Types
00067   * @{
```

```
00068   */
00069
00070 /**
00071   * @brief  Camera State structures definiti
on
00072   */
00073 typedef enum
00074 {
00075   CAMERA_OK       = 0x00,
00076   CAMERA_ERROR    = 0x01,
00077   CAMERA_TIMEOUT  = 0x02
00078 }Camera_StatusTypeDef;
00079
00080 #define RESOLUTION_R160x120      CAMERA_R160
x120      /* QQVGA Resolution     */
00081 #define RESOLUTION_R320x240      CAMERA_R320
x240      /* QVGA Resolution      */
00082 #define RESOLUTION_R480x272      CAMERA_R480
x272      /* 480x272 Resolution   */
00083 #define RESOLUTION_R640x480      CAMERA_R640
x480      /* VGA Resolution       */
00084 /**
00085   * @}
00086   */
00087
00088 /** @defgroup STM32446E_EVAL_CAMERA_Exported
_Constants STM32446E EVAL CAMERA Exported Constants

00089   * @{
00090   */
00091 /**
00092   * @}
00093   */
00094
00095 /** @defgroup STM32446E_EVAL_CAMERA_Exported
_Functions STM32446E EVAL CAMERA Exported Functions
```

```
00096   * @{
00097   */
00098 uint8_t BSP_CAMERA_Init(uint32_t Resolution)
;
00099 uint8_t BSP_CAMERA_DeInit(void);
00100 void    BSP_CAMERA_ContinuousStart(uint8_t *
buff);
00101 void    BSP_CAMERA_SnapshotStart(uint8_t *bu
ff);
00102 void    BSP_CAMERA_Suspend(void);
00103 void    BSP_CAMERA_Resume(void);
00104 uint8_t BSP_CAMERA_Stop(void);
00105 void    BSP_CAMERA_HwReset(void);
00106 void    BSP_CAMERA_PwrDown(void);
00107 void    BSP_CAMERA_LineEventCallback(void);
00108 void    BSP_CAMERA_VsyncEventCallback(void);
00109 void    BSP_CAMERA_FrameEventCallback(void);
00110 void    BSP_CAMERA_ErrorCallback(void);
00111
00112 /* Camera features functions prototype */
00113 void    BSP_CAMERA_ContrastBrightnessConfig(
uint32_t contrast_level, uint32_t brightness_level
);
00114 void    BSP_CAMERA_BlackWhiteConfig(uint32_t
 Mode);
00115 void    BSP_CAMERA_ColorEffectConfig(uint32_
t Effect);
00116
00117 /* To be called in DCMI_IRQHandler function
*/
00118 void    BSP_CAMERA_IRQHandler(void);
00119 /* To be called in DMA2_Stream1_IRQHandler f
unction */
00120 void    BSP_CAMERA_DMA_IRQHandler(void);
00121
00122 /* These functions can be modified in case t
he current settings (e.g. DMA stream)
```

```
00123    need to be changed for specific applicati
on needs */
00124 void BSP_CAMERA_MspInit(DCMI_HandleTypeDef *
hdcmi, void *Params);
00125 void BSP_CAMERA_MspDeInit(DCMI_HandleTypeDef
 *hdcmi, void *Params);
00126
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 /* __STM32446E_EVAL_CAMERA_H */
00149
00150 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32446E_EVAL BSP User Manual

stm32446e_eval_camera.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval_camera.c
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file includes the driver for Camera modules mounted on
00008   *          STM32446E-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 camera.

00044    - The S5K5CAG component driver MUST be in
cluded with this driver.
00045
00046 2. Driver description:
00047 ---------------------
00048   + Initialization steps:
00049      o Initialize the camera using the BSP_C
AMERA_Init() function.
00050      o Start the camera capture/snapshot usi
ng the CAMERA_Start() function.
00051      o Suspend, resume or stop the camera ca
pture using the following functions:
00052        - BSP_CAMERA_Suspend()
00053        - BSP_CAMERA_Resume()
00054        - BSP_CAMERA_Stop()
00055
00056   + Options
00057      o Increase or decrease on the fly the b
rightness and/or contrast
00058          using the following function:
00059          - BSP_CAMERA_ContrastBrightnessConfig

00060      o Add a special effect on the fly using
 the following functions:
```

```
00061         - BSP_CAMERA_BlackWhiteConfig()
00062         - BSP_CAMERA_ColorEffectConfig()
00063
00064 ------------------------------------------
-----------------------------------*/
00065
00066 /* Includes ------------------------------
-----------------------------------*/
00067 #include "stm32446e_eval_camera.h"
00068
00069 /** @addtogroup BSP
00070   * @{
00071   */
00072
00073 /** @addtogroup STM32446E_EVAL
00074   * @{
00075   */
00076
00077 /** @defgroup STM32446E_EVAL_CAMERA STM32446
E-EVAL CAMERA
00078   * @{
00079   */
00080
00081 /** @defgroup STM32446E_EVAL_CAMERA_Private_
TypesDefinitions STM32446E EVAL CAMERA Private Typ
esDefinitions
00082   * @{
00083   */
00084 /**
00085   * @}
00086   */
00087
00088 /** @defgroup STM32446E_EVAL_CAMERA_Private_
Defines STM32446E EVAL CAMERA Private Defines
00089   * @{
00090   */
00091 /**
```

```
00092   * @}
00093   */
00094
00095 /** @defgroup STM32446E_EVAL_CAMERA_Private_
Macros STM32446E EVAL CAMERA Private Macros
00096   * @{
00097   */
00098 /**
00099   * @}
00100   */
00101
00102 /** @defgroup STM32446E_EVAL_CAMERA_Private_
Variables STM32446E EVAL CAMERA Private Variables
00103   * @{
00104   */
00105
00106   /*
00107 Note:
00108   camera_drv variable is not compliant to na
ming rules (upper case without "_" ),
00109   but we keep this naming for compatibility,
 in fact these variables (not static)
00110   could have been used by the application, e
xample the stm32f4xx_it.c.
00111 */
00112 CAMERA_DrvTypeDef   *camera_drv;
00113
00114 static DCMI_HandleTypeDef  hDcmiEval;
00115 uint32_t currentResolution;
00116 static uint32_t  cameraHwAddress;
00117 /**
00118   * @}
00119   */
00120
00121 /** @defgroup STM32446E_EVAL_CAMERA_Private_
FunctionPrototypes STM32446E EVAL CAMERA Private F
unctionPrototypes
```

```
00122   * @{
00123   */
00124 static uint32_t GetSize(uint32_t Resolution);

00125 /**
00126   * @}
00127   */
00128
00129 /** @defgroup STM32446E_EVAL_CAMERA_Private_
Functions STM32446E EVAL CAMERA Private Functions
00130   * @{
00131   */
00132
00133 /**
00134   * @brief  Initializes the camera.
00135   * @param  Resolution: camera resolution
00136   * @retval Camera status
00137   */
00138 uint8_t BSP_CAMERA_Init(uint32_t Resolution)
00139 {
00140   DCMI_HandleTypeDef *phdcmi;
00141
00142   uint8_t ret = CAMERA_ERROR;
00143
00144   /* Get the DCMI handle structure */
00145   phdcmi = &hDcmiEval;
00146
00147   /*** Configures the DCMI to interface with
 the camera module ***/
00148   /* DCMI configuration */
00149   phdcmi->Init.CaptureRate      = DCMI_CR_AL
L_FRAME;
00150   phdcmi->Init.HSPolarity       = DCMI_HSPOL
ARITY_HIGH;
00151   phdcmi->Init.SynchroMode      = DCMI_SYNCH
RO_HARDWARE;
00152   phdcmi->Init.VSPolarity       = DCMI_VSPOL
```

```
ARITY_HIGH;
00153   phdcmi->Init.ExtendedDataMode = DCMI_EXTEN
D_DATA_8B;
00154   phdcmi->Init.PCKPolarity      = DCMI_PCKPO
LARITY_RISING;
00155   phdcmi->Init.ByteSelectMode   = DCMI_BSM_A
LL;
00156   phdcmi->Init.ByteSelectStart  = DCMI_OEBS_
ODD;
00157   phdcmi->Init.LineSelectMode   = DCMI_LSM_A
LL;
00158   phdcmi->Init.LineSelectStart  = DCMI_OELS_
ODD;
00159   phdcmi->Instance              = DCMI;
00160
00161   /* Configure IO functionalities for CAMERA
 detect pin */
00162   BSP_IO_Init();
00163
00164   /* Apply Camera hardware reset */
00165   BSP_CAMERA_HwReset();
00166
00167   /* Check if the CAMERA is plugged */
00168   if(BSP_IO_ReadPin(CAM_PLUG_PIN))
00169   {
00170     return CAMERA_ERROR;
00171   }
00172
00173   if (s5k5cag_ReadID(CAMERA_I2C_ADDRESS) ==
S5K5CAG_ID)
00174   {
00175     /* Initialize the camera driver structur
e */
00176     camera_drv = &s5k5cag_drv;
00177     cameraHwAddress = CAMERA_I2C_ADDRESS;
00178
00179     /* DCMI Initialization */
```

```
00180     BSP_CAMERA_MspInit(&hDcmiEval, NULL);
00181     HAL_DCMI_Init(phdcmi);
00182
00183     /* Camera Init */
00184     camera_drv->Init(cameraHwAddress, Resolu
tion);
00185
00186     currentResolution = Resolution;
00187
00188     /* Return CAMERA_OK status */
00189     ret = CAMERA_OK;
00190   }
00191   else
00192   {
00193     /* No supported camera sensor found */
00194     ret = CAMERA_ERROR;
00195   }
00196
00197   return ret;
00198 }
00199
00200
00201 /**
00202   * @brief  DeInitializes the camera.
00203   * @retval Camera status
00204   */
00205 uint8_t BSP_CAMERA_DeInit(void)
00206 {
00207   hDcmiEval.Instance              = DCMI;
00208
00209   HAL_DCMI_DeInit(&hDcmiEval);
00210   BSP_CAMERA_MspDeInit(&hDcmiEval, NULL);
00211   return CAMERA_OK;
00212 }
00213
00214 /**
00215   * @brief  Starts the camera capture in con
```

```
tinuous mode.
00216   * @param  buff: pointer to the camera outp
ut buffer
00217   */
00218 void BSP_CAMERA_ContinuousStart(uint8_t *buf
f)
00219 {
00220   /* Start the camera capture */
00221   HAL_DCMI_Start_DMA(&hDcmiEval, DCMI_MODE_C
ONTINUOUS, (uint32_t)buff, GetSize(currentResoluti
on));
00222 }
00223
00224 /**
00225   * @brief  Starts the camera capture in sna
pshot mode.
00226   * @param  buff: pointer to the camera outp
ut buffer
00227   */
00228 void BSP_CAMERA_SnapshotStart(uint8_t *buff)
00229 {
00230   /* Start the camera capture */
00231   HAL_DCMI_Start_DMA(&hDcmiEval, DCMI_MODE_S
NAPSHOT, (uint32_t)buff, GetSize(currentResolution
));
00232 }
00233
00234 /**
00235   * @brief Suspend the CAMERA capture
00236   */
00237 void BSP_CAMERA_Suspend(void)
00238 {
00239   /* Disable the DMA */
00240   __HAL_DMA_DISABLE(hDcmiEval.DMA_Handle);
00241   /* Disable the DCMI */
00242   __HAL_DCMI_DISABLE(&hDcmiEval);
00243
```

```
00244 }
00245
00246 /**
00247   * @brief Resume the CAMERA capture
00248   */
00249 void BSP_CAMERA_Resume(void)
00250 {
00251   /* Enable the DCMI */
00252   __HAL_DCMI_ENABLE(&hDcmiEval);
00253   /* Enable the DMA */
00254   __HAL_DMA_ENABLE(hDcmiEval.DMA_Handle);
00255 }
00256
00257 /**
00258   * @brief  Stop the CAMERA capture
00259   * @retval Camera status
00260   */
00261 uint8_t BSP_CAMERA_Stop(void)
00262 {
00263   DCMI_HandleTypeDef *phdcmi;
00264
00265   uint8_t ret = CAMERA_ERROR;
00266
00267   /* Get the DCMI handle structure */
00268   phdcmi = &hDcmiEval;
00269
00270   if(HAL_DCMI_Stop(phdcmi) == HAL_OK)
00271   {
00272      ret = CAMERA_OK;
00273   }
00274
00275   /* Set Camera in Power Down */
00276   BSP_CAMERA_PwrDown();
00277
00278   return ret;
00279 }
00280
```

```
00281 /**
00282   * @brief  CAMERA hardware reset
00283   */
00284 void BSP_CAMERA_HwReset(void)
00285 {
00286   /* Camera sensor RESET sequence */
00287   BSP_IO_ConfigPin(RSTI_PIN, IO_MODE_OUTPUT)
;
00288   BSP_IO_ConfigPin(XSDN_PIN, IO_MODE_OUTPUT)
;
00289
00290   /* Assert the camera STANDBY pin (active h
igh)  */
00291   BSP_IO_WritePin(XSDN_PIN, BSP_IO_PIN_SET);
00292
00293   /* Assert the camera RSTI pin (active low)
 */
00294   BSP_IO_WritePin(RSTI_PIN, BSP_IO_PIN_RESET
);
00295
00296   HAL_Delay(100);   /* RST and XSDN signals
asserted during 100ms */
00297
00298   /* De-assert the camera STANDBY pin (activ
e high) */
00299   BSP_IO_WritePin(XSDN_PIN, BSP_IO_PIN_RESET
);
00300
00301   HAL_Delay(3);     /* RST de-asserted and X
SDN asserted during 3ms */
00302
00303   /* De-assert the camera RSTI pin (active l
ow) */
00304   BSP_IO_WritePin(RSTI_PIN, BSP_IO_PIN_SET);
00305
00306   HAL_Delay(6);     /* RST de-asserted durin
g 3ms */
```

```
00307 }
00308
00309 /**
00310   * @brief  CAMERA power down
00311   */
00312 void BSP_CAMERA_PwrDown(void)
00313 {
00314   /* Camera power down sequence */
00315   BSP_IO_ConfigPin(RSTI_PIN, IO_MODE_OUTPUT)
;
00316   BSP_IO_ConfigPin(XSDN_PIN, IO_MODE_OUTPUT)
;
00317
00318   /* De-assert the camera STANDBY pin (activ
e high) */
00319   BSP_IO_WritePin(XSDN_PIN, BSP_IO_PIN_RESET
);
00320
00321   /* Assert the camera RSTI pin (active low)
 */
00322   BSP_IO_WritePin(RSTI_PIN, BSP_IO_PIN_RESET
);
00323 }
00324
00325 /**
00326   * @brief  Configures the camera contrast a
nd brightness.
00327   * @param  contrast_level: Contrast level
00328   *          This parameter can be one of th
e following values:
00329   *            @arg  CAMERA_CONTRAST_LEVEL4:
 for contrast +2
00330   *            @arg  CAMERA_CONTRAST_LEVEL3:
 for contrast +1
00331   *            @arg  CAMERA_CONTRAST_LEVEL2:
 for contrast  0
00332   *            @arg  CAMERA_CONTRAST_LEVEL1:
```

```
 for contrast -1
00333   *            @arg  CAMERA_CONTRAST_LEVEL0:
 for contrast -2
00334   * @param  brightness_level: Contrast level
00335   *          This parameter can be one of th
e following values:
00336   *            @arg  CAMERA_BRIGHTNESS_LEVEL
4: for brightness +2
00337   *            @arg  CAMERA_BRIGHTNESS_LEVEL
3: for brightness +1
00338   *            @arg  CAMERA_BRIGHTNESS_LEVEL
2: for brightness  0
00339   *            @arg  CAMERA_BRIGHTNESS_LEVEL
1: for brightness -1
00340   *            @arg  CAMERA_BRIGHTNESS_LEVEL
0: for brightness -2
00341   */
00342 void BSP_CAMERA_ContrastBrightnessConfig(uin
t32_t contrast_level, uint32_t brightness_level)
00343 {
00344   if(camera_drv->Config != NULL)
00345   {
00346     camera_drv->Config(cameraHwAddress, CAME
RA_CONTRAST_BRIGHTNESS, contrast_level, brightness
_level);
00347   }
00348 }
00349
00350 /**
00351   * @brief  Configures the camera white bala
nce.
00352   * @param  Mode: black_white mode
00353   *          This parameter can be one of th
e following values:
00354   *            @arg  CAMERA_BLACK_WHITE_BW
00355   *            @arg  CAMERA_BLACK_WHITE_NEGA
TIVE
```

```
00356   *             @arg  CAMERA_BLACK_WHITE_BW_N
EGATIVE
00357   *             @arg  CAMERA_BLACK_WHITE_NORM
AL
00358   */
00359 void BSP_CAMERA_BlackWhiteConfig(uint32_t Mo
de)
00360 {
00361   if(camera_drv->Config != NULL)
00362   {
00363     camera_drv->Config(cameraHwAddress, CAME
RA_BLACK_WHITE, Mode, 0);
00364   }
00365 }
00366
00367 /**
00368   * @brief  Configures the camera color effe
ct.
00369   * @param  Effect: Color effect
00370   *          This parameter can be one of th
e following values:
00371   *             @arg  CAMERA_COLOR_EFFECT_ANT
IQUE
00372   *             @arg  CAMERA_COLOR_EFFECT_BLU
E
00373   *             @arg  CAMERA_COLOR_EFFECT_GRE
EN
00374   *             @arg  CAMERA_COLOR_EFFECT_RED

00375   */
00376 void BSP_CAMERA_ColorEffectConfig(uint32_t E
ffect)
00377 {
00378   if(camera_drv->Config != NULL)
00379   {
00380     camera_drv->Config(cameraHwAddress, CAME
RA_COLOR_EFFECT, Effect, 0);
```

```
00381   }
00382 }
00383
00384 /**
00385   * @brief  Handles DCMI interrupt request.
00386   */
00387 void BSP_CAMERA_IRQHandler(void)
00388 {
00389   HAL_DCMI_IRQHandler(&hDcmiEval);
00390 }
00391
00392 /**
00393   * @brief  Handles DMA interrupt request.
00394   */
00395 void BSP_CAMERA_DMA_IRQHandler(void)
00396 {
00397   HAL_DMA_IRQHandler(hDcmiEval.DMA_Handle);
00398 }
00399
00400 /**
00401   * @brief  Get the capture size.
00402   * @param  Resolution: the current resoluti
on.
00403   * @retval capture size.
00404   */
00405 static uint32_t GetSize(uint32_t Resolution)
00406 {
00407   uint32_t size = 0;
00408
00409   /* Get capture size */
00410   switch (Resolution)
00411   {
00412   case CAMERA_R160x120:
00413     {
00414       size =  0x2580;
00415     }
00416     break;
```

```
00417   case CAMERA_R320x240:
00418     {
00419       size =  0x9600;
00420     }
00421     break;
00422   case CAMERA_R480x272:
00423     {
00424       size =  0xFF00;
00425     }
00426     break;
00427   case CAMERA_R640x480:
00428     {
00429       size =  0x25800;
00430     }
00431     break;
00432   default:
00433     {
00434       break;
00435     }
00436   }
00437
00438   return size;
00439 }
00440
00441 /**
00442   * @brief  Initializes the DCMI MSP.
00443   * @param  hdcmi: HDMI handle
00444   * @param  Params: pointer on additional co
nfiguration parameters, can be NULL.
00445   */
00446 __weak void BSP_CAMERA_MspInit(DCMI_HandleTy
peDef *hdcmi, void *Params)
00447 {
00448   static DMA_HandleTypeDef hdma_eval;
00449   GPIO_InitTypeDef gpio_init_structure;
00450
00451   /*** Enable peripherals and GPIO clocks **
```

```
*/
00452   /* Enable DCMI clock */
00453   __HAL_RCC_DCMI_CLK_ENABLE();
00454
00455   /* Enable DMA2 clock */
00456   __HAL_RCC_DMA2_CLK_ENABLE();
00457
00458   /* Enable GPIO clocks */
00459   __HAL_RCC_GPIOA_CLK_ENABLE();
00460   __HAL_RCC_GPIOB_CLK_ENABLE();
00461   __HAL_RCC_GPIOC_CLK_ENABLE();
00462   __HAL_RCC_GPIOD_CLK_ENABLE();
00463   __HAL_RCC_GPIOE_CLK_ENABLE();
00464
00465   /*** Configure the GPIO ***/
00466   /* Configure DCMI GPIO as alternate functi
on */
00467   gpio_init_structure.Pin       = GPIO_PIN_4
 | GPIO_PIN_6;
00468   gpio_init_structure.Mode      = GPIO_MODE_
AF_PP;
00469   gpio_init_structure.Pull      = GPIO_PULLU
P;
00470   gpio_init_structure.Speed     = GPIO_SPEED
_HIGH;
00471   gpio_init_structure.Alternate = GPIO_AF13_
DCMI;
00472   HAL_GPIO_Init(GPIOA, &gpio_init_structure)
;
00473
00474 #if !defined(USE_STM32446E_EVAL_REVA)
00475   gpio_init_structure.Pin       = GPIO_PIN_6
 | GPIO_PIN_7 | GPIO_PIN_8 | GPIO_PIN_9;
00476 #else
00477   gpio_init_structure.Pin       = GPIO_PIN_7
 | GPIO_PIN_8 | GPIO_PIN_9;
00478 #endif // !USE_STM32446E_EVAL_REVA
```

```
00479   gpio_init_structure.Mode      = GPIO_MODE_
AF_PP;
00480   gpio_init_structure.Pull      = GPIO_PULLU
P;
00481   gpio_init_structure.Speed     = GPIO_SPEED
_HIGH;
00482   gpio_init_structure.Alternate = GPIO_AF13_
DCMI;
00483   HAL_GPIO_Init(GPIOB, &gpio_init_structure)
;
00484
00485   gpio_init_structure.Pin       = GPIO_PIN_6
 | GPIO_PIN_7  | GPIO_PIN_8  |\
00486                                   GPIO_PIN_9
 | GPIO_PIN_10 | GPIO_PIN_11 |\
00487                                   GPIO_PIN_1
2;
00488   gpio_init_structure.Mode      = GPIO_MODE_
AF_PP;
00489   gpio_init_structure.Pull      = GPIO_PULLU
P;
00490   gpio_init_structure.Speed     = GPIO_SPEED
_HIGH;
00491   gpio_init_structure.Alternate = GPIO_AF13_
DCMI;
00492   HAL_GPIO_Init(GPIOC, &gpio_init_structure)
;
00493
00494 #if !defined(USE_STM32446E_EVAL_REVA)
00495   gpio_init_structure.Pin       = GPIO_PIN_2
 | GPIO_PIN_6;
00496 #else
00497   gpio_init_structure.Pin       = GPIO_PIN_2
 | GPIO_PIN_3 | GPIO_PIN_6;
00498 #endif // !USE_STM32446E_EVAL_REVA
00499   gpio_init_structure.Mode      = GPIO_MODE_
AF_PP;
```

```
00500   gpio_init_structure.Pull      = GPIO_PULLU
P;
00501   gpio_init_structure.Speed     = GPIO_SPEED
_HIGH;
00502   gpio_init_structure.Alternate = GPIO_AF13_
DCMI;
00503   HAL_GPIO_Init(GPIOD, &gpio_init_structure)
;
00504
00505
00506   /*** Configure the DMA ***/
00507   /* Set the parameters to be configured */
00508   hdma_eval.Init.Channel             = DMA_C
HANNEL_1;
00509   hdma_eval.Init.Direction           = DMA_P
ERIPH_TO_MEMORY;
00510   hdma_eval.Init.PeriphInc           = DMA_P
INC_DISABLE;
00511   hdma_eval.Init.MemInc              = DMA_M
INC_ENABLE;
00512   hdma_eval.Init.PeriphDataAlignment = DMA_P
DATAALIGN_WORD;
00513   hdma_eval.Init.MemDataAlignment    = DMA_M
DATAALIGN_WORD;
00514   hdma_eval.Init.Mode                = DMA_C
IRCULAR;
00515   hdma_eval.Init.Priority            = DMA_P
RIORITY_HIGH;
00516   hdma_eval.Init.FIFOMode            = DMA_F
IFOMODE_DISABLE;
00517   hdma_eval.Init.FIFOThreshold       = DMA_F
IFO_THRESHOLD_FULL;
00518   hdma_eval.Init.MemBurst            = DMA_M
BURST_SINGLE;
00519   hdma_eval.Init.PeriphBurst         = DMA_P
BURST_SINGLE;
00520
```

```
00521   hdma_eval.Instance = DMA2_Stream1;
00522
00523   /* Associate the initialized DMA handle to
 the DCMI handle */
00524   __HAL_LINKDMA(hdcmi, DMA_Handle, hdma_eval
);
00525
00526   /*** Configure the NVIC for DCMI and DMA *
**/
00527   /* NVIC configuration for DCMI transfer co
mplete interrupt */
00528   HAL_NVIC_SetPriority(DCMI_IRQn, 5, 0);
00529   HAL_NVIC_EnableIRQ(DCMI_IRQn);
00530
00531   /* NVIC configuration for DMA2D transfer c
omplete interrupt */
00532   HAL_NVIC_SetPriority(DMA2_Stream1_IRQn, 5,
 0);
00533   HAL_NVIC_EnableIRQ(DMA2_Stream1_IRQn);
00534
00535   /* Configure the DMA stream */
00536   HAL_DMA_Init(hdcmi->DMA_Handle);
00537 }
00538
00539
00540 /**
00541   * @brief  DeInitializes the DCMI MSP.
00542   * @param  hdcmi: HDMI handle
00543   * @param  Params: pointer on additional co
nfiguration parameters, can be NULL.
00544   */
00545 __weak void BSP_CAMERA_MspDeInit(DCMI_Handle
TypeDef *hdcmi, void *Params)
00546 {
00547     /* Disable NVIC  for DCMI transfer compl
ete interrupt */
00548     HAL_NVIC_DisableIRQ(DCMI_IRQn);
```

```
00549
00550     /* Disable NVIC for DMA2 transfer comple
te interrupt */
00551     HAL_NVIC_DisableIRQ(DMA2_Stream1_IRQn);
00552
00553     /* Configure the DMA stream */
00554     HAL_DMA_DeInit(hdcmi->DMA_Handle);
00555
00556     /* Disable DCMI clock */
00557     __HAL_RCC_DCMI_CLK_DISABLE();
00558
00559     /* GPIO pins clock and DMA clock can be
shut down in the application
00560        by surcharging this __weak function */

00561 }
00562
00563 /**
00564   * @brief  Line event callback
00565   * @param  hdcmi: pointer to the DCMI handl
e
00566   */
00567 void HAL_DCMI_LineEventCallback(DCMI_HandleT
ypeDef *hdcmi)
00568 {
00569   BSP_CAMERA_LineEventCallback();
00570 }
00571
00572 /**
00573   * @brief  Line Event callback.
00574   */
00575 __weak void BSP_CAMERA_LineEventCallback(void
)
00576 {
00577   /* NOTE : This function Should not be modi
fied, when the callback is needed,
00578             the HAL_DCMI_LineEventCallback c
```

```
ould be implemented in the user file
00579    */
00580 }
00581
00582 /**
00583   * @brief  VSYNC event callback
00584   * @param  hdcmi: pointer to the DCMI handl
e
00585   */
00586 void HAL_DCMI_VsyncEventCallback(DCMI_Handle
TypeDef *hdcmi)
00587 {
00588   BSP_CAMERA_VsyncEventCallback();
00589 }
00590
00591 /**
00592   * @brief  VSYNC Event callback.
00593   */
00594 __weak void BSP_CAMERA_VsyncEventCallback(vo
id)
00595 {
00596   /* NOTE : This function Should not be modi
fied, when the callback is needed,
00597             the HAL_DCMI_VsyncEventCallback
could be implemented in the user file
00598    */
00599 }
00600
00601 /**
00602   * @brief  Frame event callback
00603   * @param  hdcmi: pointer to the DCMI handl
e
00604   */
00605 void HAL_DCMI_FrameEventCallback(DCMI_Handle
TypeDef *hdcmi)
00606 {
00607   BSP_CAMERA_FrameEventCallback();
```

```
00608 }
00609
00610 /**
00611   * @brief  Frame Event callback.
00612   */
00613 __weak void BSP_CAMERA_FrameEventCallback(vo
id)
00614 {
00615   /* NOTE : This function Should not be modi
fied, when the callback is needed,
00616             the HAL_DCMI_FrameEventCallback
could be implemented in the user file
00617    */
00618 }
00619
00620 /**
00621   * @brief  Error callback
00622   * @param  hdcmi: pointer to the DCMI handl
e
00623   */
00624 void HAL_DCMI_ErrorCallback(DCMI_HandleTypeD
ef *hdcmi)
00625 {
00626   BSP_CAMERA_ErrorCallback();
00627 }
00628
00629 /**
00630   * @brief  Error callback.
00631   */
00632 __weak void BSP_CAMERA_ErrorCallback(void)
00633 {
00634   /* NOTE : This function Should not be modi
fied, when the callback is needed,
00635             the HAL_DCMI_ErrorCallback could
 be implemented in the user file
00636    */
00637 }
```

```
00638
00639 /**
00640   * @}
00641   */
00642
00643 /**
00644   * @}
00645   */
00646
00647 /**
00648   * @}
00649   */
00650
00651 /**
00652   * @}
00653   */
00654
00655 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32446E_EVAL BSP User Manual

stm32446e_eval_io.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval_io.h
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm32446e_eval_io.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 __STM32446E_EVAL_IO_H
00041 #define __STM32446E_EVAL_IO_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------------------------------------------*/
00048 #include "stm32446e_eval.h"
00049 /* Include IO component driver */
00050 #include "../Components/mfxstm32l152/mfxstm32l152.h"
00051
00052 /** @addtogroup BSP
00053   * @{
00054   */
00055
00056 /** @addtogroup STM32446E_EVAL
00057   * @{
00058   */
00059
00060 /** @defgroup STM32446E_EVAL_IO STM32446E EVAL IO
00061   * @{
00062   */
00063
00064 /** @defgroup STM32446E_EVAL_IO_Exported_Types STM32446E EVAL IO Exported Types
00065   * @{
00066   */
```

```
00067
00068 typedef enum
00069 {
00070   BSP_IO_PIN_RESET = 0,
00071   BSP_IO_PIN_SET   = 1
00072 }BSP_IO_PinStateTypeDef;
00073
00074 typedef enum
00075 {
00076   IO_OK       = 0,
00077   IO_ERROR    = 1,
00078   IO_TIMEOUT  = 2
00079 }IO_StatusTypeDef;
00080
00081 /**
00082   * @}
00083   */
00084
00085 /** @defgroup STM32446E_EVAL_IO_Exported_Con
stants STM32446E EVAL IO Exported Constants
00086   * @{
00087   */
00088 #define IO_PIN_0                    ((uint32_t
)0x0001)
00089 #define IO_PIN_1                    ((uint32_t
)0x0002)
00090 #define IO_PIN_2                    ((uint32_t
)0x0004)
00091 #define IO_PIN_3                    ((uint32_t
)0x0008)
00092 #define IO_PIN_4                    ((uint32_t
)0x0010)
00093 #define IO_PIN_5                    ((uint32_t
)0x0020)
00094 #define IO_PIN_6                    ((uint32_t
)0x0040)
00095 #define IO_PIN_7                    ((uint32_t
```

```
)0x0080)
00096 #define IO_PIN_8                    ((uint32_t
)0x0100)
00097 #define IO_PIN_9                    ((uint32_t
)0x0200)
00098 #define IO_PIN_10                   ((uint32_t
)0x0400)
00099 #define IO_PIN_11                   ((uint32_t
)0x0800)
00100 #define IO_PIN_12                   ((uint32_t
)0x1000)
00101 #define IO_PIN_13                   ((uint32_t
)0x2000)
00102 #define IO_PIN_14                   ((uint32_t
)0x4000)
00103 #define IO_PIN_15                   ((uint32_t
)0x8000)
00104 #define IO_PIN_16                  ((uint32_t)0
x010000)
00105 #define IO_PIN_17                  ((uint32_t)0
x020000)
00106 #define IO_PIN_18                  ((uint32_t)0
x040000)
00107 #define IO_PIN_19                  ((uint32_t)0
x080000)
00108 #define IO_PIN_20                  ((uint32_t)0
x100000)
00109 #define IO_PIN_21                  ((uint32_t)0
x200000)
00110 #define IO_PIN_22                  ((uint32_t)0
x400000)
00111 #define IO_PIN_23                  ((uint32_t)0
x800000)
00112 #define IO_PIN_ALL                 ((uint32_t)0
xFFFFFF)
00113 /**
00114   * @}
```

```
00115   */
00116
00117 /** @defgroup STM32446E_EVAL_IO_Exported_Mac
ro STM32446E EVAL IO Exported Macro
00118   * @{
00119   */
00120 /**
00121   * @}
00122   */
00123
00124 /** @defgroup STM32446E_EVAL_IO_Exported_Fun
ctions STM32446E EVAL IO Exported Functions
00125   * @{
00126   */
00127 uint8_t  BSP_IO_Init(void);
00128 uint8_t  BSP_IO_DeInit(void);
00129 uint8_t  BSP_IO_ConfigIrqOutPin(uint8_t IoIr
qOutPinPolarity, uint8_t IoIrqOutPinType);
00130 uint32_t BSP_IO_ITGetStatus(uint32_t IoPin);
00131 void     BSP_IO_ITClear(void);
00132 uint8_t  BSP_IO_ConfigPin(uint32_t IoPin, IO
_ModeTypedef IoMode);
00133 void     BSP_IO_WritePin(uint32_t IoPin, BSP
_IO_PinStateTypeDef PinState);
00134 uint32_t BSP_IO_ReadPin(uint32_t IoPin);
00135 void     BSP_IO_TogglePin(uint32_t IoPin);
00136
00137 /**
00138   * @}
00139   */
00140
00141 /**
00142   * @}
00143   */
00144
00145 /**
00146   * @}
```

```
00147   */
00148
00149 /**
00150   * @}
00151   */
00152
00153 #ifdef __cplusplus
00154 }
00155 #endif
00156
00157 #endif /* __STM32446E_EVAL_IO_H */
00158
00159 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM32446E_EVAL BSP User Manual

stm32446e_eval_eeprom.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval_eeprom.h
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file contains all the functions prototypes for
00008   *          the stm32446e_eval_eeprom.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 __STM32F446E_EVAL_EEPROM_H
00041 #define __STM32F446E_EVAL_EEPROM_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------
-----------------------------------*/
00048 #include "stm32446e_eval.h"
00049
00050 /** @addtogroup BSP
00051   * @{
00052   */
00053
00054 /** @addtogroup STM32446E_EVAL
00055   * @{
00056   */
00057
00058 /** @defgroup STM32446E_EVAL_EEPROM STM32446
E EVAL EEPROM
00059   * @brief This file includes the I2C EEPROM
 driver of STM32446E-EVAL evaluation board.
00060   * @{
00061   */
00062
00063 /** @defgroup STM32446E_EVAL_EEPROM_Exported
_Types STM32446E EVAL EEPROM Exported Types
00064   * @{
00065   */
```

```
00066 /**
00067   * @}
00068   */
00069
00070 /** @defgroup STM32446E_EVAL_EEPROM_Exported
_Constants STM32446E EVAL EEPROM Exported Constants

00071   * @{
00072   */
00073 /* EEPROM hardware address and page size */
00074 #define EEPROM_PAGESIZE             ((uint8_
t)4)
00075 #define EEPROM_MAX_SIZE             ((uint16
_t)0x2000) /* 64Kbit */
00076
00077
00078 /* Maximum number of trials for EEPROM_WaitE
epromStandbyState() function */
00079 #define EEPROM_MAX_TRIALS           ((uint32
_t)3000)
00080
00081 #define EEPROM_OK                   ((uint32
_t)0)
00082 #define EEPROM_FAIL                 ((uint32
_t)1)
00083 #define EEPROM_TIMEOUT              ((uint32
_t)2)
00084 /**
00085   * @}
00086   */
00087
00088 /** @defgroup STM32446E_EVAL_EEPROM_Exported
_Macros STM32446E EVAL EEPROM Exported Macros
00089   * @{
00090   */
00091 /**
00092   * @}
```

```
00093   */
00094
00095 /** @defgroup STM32446E_EVAL_EEPROM_Exported
_Functions STM32446E EVAL EEPROM Exported Functions

00096   * @{
00097   */
00098 uint32_t BSP_EEPROM_Init(void);
00099 uint8_t BSP_EEPROM_DeInit(void);
00100 uint32_t BSP_EEPROM_ReadBuffer(uint8_t* pBuf
fer, uint16_t ReadAddr, uint16_t* NumByteToRead);
00101 uint32_t BSP_EEPROM_WritePage(uint8_t* pBuff
er, uint16_t WriteAddr, uint8_t* NumByteToWrite);
00102 uint32_t BSP_EEPROM_WriteBuffer(uint8_t* pBu
ffer, uint16_t WriteAddr, uint16_t NumByteToWrite)
;
00103 uint32_t BSP_EEPROM_WaitEepromStandbyState(v
oid);
00104
00105 /* USER Callbacks: This function is declared
 as __weak in EEPROM driver and
00106    should be implemented into user applicati
on.
00107    BSP_EEPROM_TIMEOUT_UserCallback() functio
n is called whenever a timeout condition
00108    occurs during communication (waiting on a
n event that doesn't occur, bus
00109    errors, busy devices ...). */
00110 void     BSP_EEPROM_TIMEOUT_UserCallback(void
);
00111
00112 /* Link function for I2C EEPROM peripheral */

00113 void              EEPROM_IO_Init(void);
00114 HAL_StatusTypeDef EEPROM_IO_WriteData(uint16
_t DevAddress, uint16_t MemAddress, uint8_t *pBuff
er, uint32_t BufferSize);
```

```
00115 HAL_StatusTypeDef EEPROM_IO_ReadData(uint16_
t DevAddress, uint16_t MemAddress, uint8_t *pBuffe
r, uint32_t BufferSize);
00116 HAL_StatusTypeDef EEPROM_IO_IsDeviceReady(ui
nt16_t DevAddress, uint32_t Trials);
00117
00118 /**
00119   * @}
00120   */
00121
00122 /**
00123   * @}
00124   */
00125
00126 /**
00127   * @}
00128   */
00129
00130 /**
00131   * @}
00132   */
00133
00134 #ifdef __cplusplus
00135 }
00136 #endif
00137
00138 #endif /* __STM32446E_EVAL_EEPROM_H */
00139
00140 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32446E_EVAL BSP User Manual

stm32446e_eval_eeprom.c

Go to the documentation of this file.

```
00001 /**
00002   ********************************************************************************
00003   * @file    stm32446e_eval_eeprom.c
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file provides a set of functions needed to manage an I2C M24LR64
00008   *          EEPROM memory.
00009   *          To be able to use this driver, the switch EE_M24LR64 must be defined
00010   *          in your toolchain compiler preprocessor
00011   *
00012   *          ===================================================================
00013   *          Notes:
00014   *           - This driver is intended for STM32F4xx families devices only.
00015   *           - The I2C EEPROM memory (M24LR64) is available on separate daughter
```

```
00016   *              board ANT7-M24LR-A, which is
 not provided with the STM32446E_EVAL
00017   *              board.
00018   *              To use this driver you have
to connect the ANT7-M24LR-A to CN3
00019   *              connector of STM32446E_EVAL
board.
00020   *           ==============================
==================================
00021   *
00022   *           It implements a high level comm
unication layer for read and write
00023   *           from/to this memory. The needed
 STM32F4xx hardware resources (I2C and
00024   *           GPIO) are defined in stm32446e_
eval.h file, and the initialization is
00025   *           performed in EEPROM_IO_Init() f
unction declared in stm32446e_eval.c
00026   *           file.
00027   *           You can easily tailor this driv
er to any other development board,
00028   *           by just adapting the defines fo
r hardware resources and
00029   *           EEPROM_IO_Init() function.
00030   *
00031   *           @note In this driver, basic rea
d and write functions (BSP_EEPROM_ReadBuffer()
00032   *                 and BSP_EEPROM_WritePage(
)) use DMA mode to perform the data
00033   *                 transfer to/from EEPROM m
emory.
00034   *
00035   *         @note   Regarding BSP_EEPROM_Wri
tePage(), it is a optimized function to perform
00036   *                 small write (less than 1
page) BUT The number of bytes (combined to write s
tart address) must not
```

```
00037   *                cross the EEPROM page bou
ndary. This function can only write into
00038   *                the boundaries of an EEPR
OM page.
00039   *                This function doesn't che
ck on boundaries condition (in this driver
00040   *                the function BSP_EEPROM_W
riteBuffer() which calls BSP_EEPROM_WritePage() is

00041   *                responsible of checking o
n Page boundaries).
00042   *
00043   *
00044   *     +-----------------------------------
--------------------------------+
00045   *     |                Pin assignment for M
24LR64 EEPROM                  |
00046   *     +-----------------------------------
----+-----------+-------------+
00047   *     |  STM32F4xx I2C Pins
    |   EEPROM  |   Pin       |
00048   *     +-----------------------------------
----+-----------+-------------+
00049   *     | .
    |   E0(GND) |    1  (0V)  |
00050   *     | .
    |   AC0     |    2        |
00051   *     | .
    |   AC1     |    3        |
00052   *     | .
    |   VSS     |    4  (0V)  |
00053   *     | SDA
    |   SDA     |    5        |
00054   *     | SCL
    |   SCL     |    6        |
00055   *     | .
    |   E1(GND) |    7  (0V)  |
```

```
00056   *     | .
    |   VDD     |    8 (3.3V) |
00057   *     +-----------------------------------
----+-----------+-------------+
00058   *
00059   ****************************************
**********************************
00060   * @attention
00061   *
00062   * <h2><center>&copy; COPYRIGHT(c) 2015 STM
icroelectronics</center></h2>
00063   *
00064   * Redistribution and use in source and bin
ary forms, with or without modification,
00065   * are permitted provided that the followin
g conditions are met:
00066   *   1. Redistributions of source code must
 retain the above copyright notice,
00067   *      this list of conditions and the fol
lowing disclaimer.
00068   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00069   *      this list of conditions and the fol
lowing disclaimer in the documentation
00070   *      and/or other materials provided wit
h the distribution.
00071   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00072   *      may be used to endorse or promote p
roducts derived from this software
00073   *      without specific prior written perm
ission.
00074   *
00075   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00076   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
```

```
   * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
   * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
   * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
   * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
   * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
   *
   ******************************************************************************
   */
/* Includes ------------------------------------------------------------------*/
#include "stm32446e_eval_eeprom.h"

/** @addtogroup BSP
   * @{
   */

/** @addtogroup STM32446E_EVAL
   * @{
   */

/** @defgroup STM32446E_EVAL_EEPROM STM32446E EVAL EEPROM
   * @brief This file includes the I2C EEPROM driver of STM32446E-EVAL evaluation board.
   * @{
```

```
00102   */
00103
00104 /** @defgroup STM32446E_EVAL_EEPROM_Private_
Types STM32446E EVAL EEPROM Private Types
00105   * @{
00106   */
00107 /**
00108   * @}
00109   */
00110
00111 /** @defgroup STM32446E_EVAL_EEPROM_Private_
Defines STM32446E EVAL EEPROM Private Defines
00112   * @{
00113   */
00114 /**
00115   * @}
00116   */
00117
00118 /** @defgroup STM32446E_EVAL_EEPROM_Private_
Macros STM32446E EVAL EEPROM Private Macros
00119   * @{
00120   */
00121 /**
00122   * @}
00123   */
00124
00125 /** @defgroup STM32446E_EVAL_EEPROM_Private_
Variables STM32446E EVAL EEPROM Private Variables
00126   * @{
00127   */
00128 __IO uint16_t EEPROMAddress = 0;
00129 __IO uint16_t EEPROMDataRead;
00130 __IO uint8_t  EEPROMDataWrite;
00131 /**
00132   * @}
00133   */
00134
```

```
00135 /** @defgroup STM32446E_EVAL_EEPROM_Private_
Function_Prototypes STM32446E EVAL EEPROM Private
Function Prototypes
00136   * @{
00137   */
00138 /**
00139   * @}
00140   */
00141
00142 /** @defgroup STM32446E_EVAL_EEPROM_Private_
Functions STM32446E EVAL EEPROM Private Functions
00143   * @{
00144   */
00145
00146 /**
00147   * @brief  Initializes peripherals used by
the I2C EEPROM driver.
00148   *
00149   * @note   There are 2 different versions o
f M24LR64 (A01 & A02).
00150   *             Then try to connect on 1st o
ne (EEPROM_I2C_ADDRESS_A01)
00151   *             and if problem, check the 2n
d one (EEPROM_I2C_ADDRESS_A02)
00152   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00153   *         different from EEPROM_OK (0)
00154   */
00155 uint32_t BSP_EEPROM_Init(void)
00156 {
00157   /* I2C Initialization */
00158   EEPROM_IO_Init();
00159
00160   /* Select the EEPROM address for A01 and c
heck if OK */
00161   EEPROMAddress = EEPROM_I2C_ADDRESS_A01;
00162   if(EEPROM_IO_IsDeviceReady(EEPROMAddress,
```

```
EEPROM_MAX_TRIALS) != HAL_OK)
00163   {
00164     /* Select the EEPROM address for A02 and
 check if OK */
00165     EEPROMAddress = EEPROM_I2C_ADDRESS_A02;
00166     if(EEPROM_IO_IsDeviceReady(EEPROMAddress
, EEPROM_MAX_TRIALS) != HAL_OK)
00167     {
00168       return EEPROM_FAIL;
00169     }
00170   }
00171   return EEPROM_OK;
00172 }
00173
00174 /**
00175   * @brief  DeInitializes the EEPROM.
00176   * @retval EEPROM state
00177   */
00178 uint8_t BSP_EEPROM_DeInit(void)
00179 {
00180   /* I2C won't be disabled because common to
 other functionalities */
00181   return EEPROM_OK;
00182 }
00183
00184 /**
00185   * @brief  Reads a block of data from the E
EPROM.
00186   * @param  pBuffer: pointer to the buffer t
hat receives the data read from
00187   *         the EEPROM.
00188   * @param  ReadAddr: EEPROM's internal addr
ess to start reading from.
00189   * @param  NumByteToRead: pointer to the va
riable holding number of bytes to
00190   *         be read from the EEPROM.
00191   *
```

```
00192   *        @note The variable pointed by Num
ByteToRead is reset to 0 when all the
00193   *              data are read from the EEPR
OM. Application should monitor this
00194   *              variable in order know when
 the transfer is complete.
00195   *
00196   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00197   *         different from EEPROM_OK (0) or
the timeout user callback.
00198   */
00199 uint32_t BSP_EEPROM_ReadBuffer(uint8_t* pBuf
fer, uint16_t ReadAddr, uint16_t* NumByteToRead)
00200 {
00201   uint32_t buffersize = *NumByteToRead;
00202
00203   /* Set the pointer to the Number of data t
o be read. This pointer will be used
00204      by the DMA Transfer Completer interrupt
 Handler in order to reset the
00205      variable to 0. User should check on thi
s variable in order to know if the
00206      DMA transfer has been complete or not.
*/
00207   EEPROMDataRead = *NumByteToRead;
00208
00209   if(EEPROM_IO_ReadData(EEPROMAddress, ReadA
ddr, pBuffer, buffersize) != HAL_OK)
00210   {
00211     BSP_EEPROM_TIMEOUT_UserCallback();
00212     return EEPROM_FAIL;
00213   }
00214
00215   /* If all operations OK, return EEPROM_OK
(0) */
00216   return EEPROM_OK;
```

```
00217 }
00218
00219 /**
00220   * @brief  Writes more than one byte to the
 EEPROM with a single WRITE cycle.
00221   *
00222   * @note   The number of bytes (combined to
 write start address) must not
00223   *         cross the EEPROM page boundary.
This function can only write into
00224   *         the boundaries of an EEPROM page.

00225   *         This function doesn't check on b
oundaries condition (in this driver
00226   *         the function BSP_EEPROM_WriteBuf
fer() which calls BSP_EEPROM_WritePage() is
00227   *         responsible of checking on Page
boundaries).
00228   *
00229   * @param  pBuffer: pointer to the buffer c
ontaining the data to be written to
00230   *         the EEPROM.
00231   * @param  WriteAddr: EEPROM's internal add
ress to write to.
00232   * @param  NumByteToWrite: pointer to the v
ariable holding number of bytes to
00233   *         be written into the EEPROM.
00234   *
00235   *        @note The variable pointed by Num
ByteToWrite is reset to 0 when all the
00236   *              data are written to the EEP
ROM. Application should monitor this
00237   *              variable in order know when
 the transfer is complete.
00238   *
00239   *        @note This function just configur
e the communication and enable the DMA
```

```
00240   *              channel to transfer data. M
eanwhile, the user application may perform
00241   *              other tasks in parallel.
00242   *
00243   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00244   *         different from EEPROM_OK (0) or
the timeout user callback.
00245   */
00246 uint32_t BSP_EEPROM_WritePage(uint8_t* pBuff
er, uint16_t WriteAddr, uint8_t* NumByteToWrite)
00247 {
00248   uint32_t buffersize = *NumByteToWrite;
00249   uint32_t status = EEPROM_OK;
00250
00251   /* Set the pointer to the Number of data t
o be written. This pointer will be used
00252       by the DMA Transfer Completer interrup
t Handler in order to reset the
00253       variable to 0. User should check on th
is variable in order to know if the
00254       DMA transfer has been complete or not.
 */
00255   EEPROMDataWrite = *NumByteToWrite;
00256
00257   if(EEPROM_IO_WriteData(EEPROMAddress, Writ
eAddr, pBuffer, buffersize) != HAL_OK)
00258   {
00259     BSP_EEPROM_TIMEOUT_UserCallback();
00260     status = EEPROM_FAIL;
00261   }
00262
00263   if(BSP_EEPROM_WaitEepromStandbyState() !=
EEPROM_OK)
00264   {
00265     return EEPROM_FAIL;
00266   }
```

```
00267
00268   /* If all operations OK, return EEPROM_OK
(0) */
00269   return status;
00270 }
00271
00272 /**
00273   * @brief  Writes buffer of data to the I2C
 EEPROM.
00274   * @param  pBuffer: pointer to the buffer
containing the data to be written
00275   *         to the EEPROM.
00276   * @param  WriteAddr: EEPROM's internal add
ress to write to.
00277   * @param  NumByteToWrite: number of bytes
to write to the EEPROM.
00278   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00279   *         different from EEPROM_OK (0) or
the timeout user callback.
00280   */
00281 uint32_t BSP_EEPROM_WriteBuffer(uint8_t *pBu
ffer, uint16_t WriteAddr, uint16_t NumByteToWrite)
00282 {
00283   uint16_t numofpage = 0, numofsingle = 0, c
ount = 0;
00284   uint16_t addr = 0;
00285   uint8_t  dataindex = 0;
00286   uint32_t status = EEPROM_OK;
00287
00288   addr = WriteAddr % EEPROM_PAGESIZE;
00289   count = EEPROM_PAGESIZE - addr;
00290   numofpage =  NumByteToWrite / EEPROM_PAGES
IZE;
00291   numofsingle = NumByteToWrite % EEPROM_PAGE
SIZE;
00292
```

```
00293   /* If WriteAddr is EEPROM_PAGESIZE aligned
 */
00294   if(addr == 0)
00295   {
00296     /* If NumByteToWrite < EEPROM_PAGESIZE */

00297     if(numofpage == 0)
00298     {
00299       /* Store the number of data to be writ
ten */
00300       dataindex = numofsingle;
00301       /* Start writing data */
00302       status = BSP_EEPROM_WritePage(pBuffer,
 WriteAddr, (uint8_t*)(&dataindex));
00303       if(status != EEPROM_OK)
00304       {
00305         return status;
00306       }
00307     }
00308     /* If NumByteToWrite > EEPROM_PAGESIZE */

00309     else
00310     {
00311       while(numofpage--)
00312       {
00313         /* Store the number of data to be wr
itten */
00314         dataindex = EEPROM_PAGESIZE;
00315         status = BSP_EEPROM_WritePage(pBuffe
r, WriteAddr, (uint8_t*)(&dataindex));
00316         if(status != EEPROM_OK)
00317         {
00318           return status;
00319         }
00320
00321         WriteAddr +=  EEPROM_PAGESIZE;
00322         pBuffer += EEPROM_PAGESIZE;
```

```
00323       }
00324
00325       if(numofsingle!=0)
00326       {
00327         /* Store the number of data to be wr
itten */
00328         dataindex = numofsingle;
00329         status = BSP_EEPROM_WritePage(pBuffe
r, WriteAddr, (uint8_t*)(&dataindex));
00330         if(status != EEPROM_OK)
00331         {
00332           return status;
00333         }
00334       }
00335     }
00336   }
00337   /* If WriteAddr is not EEPROM_PAGESIZE ali
gned */
00338   else
00339   {
00340     /* If NumByteToWrite < EEPROM_PAGESIZE */

00341     if(numofpage== 0)
00342     {
00343       /* If the number of data to be written
 is more than the remaining space
00344       in the current page: */
00345       if(NumByteToWrite > count)
00346       {
00347         /* Store the number of data to be wr
itten */
00348         dataindex = count;
00349         /* Write the data contained in same
page */
00350         status = BSP_EEPROM_WritePage(pBuffe
r, WriteAddr, (uint8_t*)(&dataindex));
00351         if(status != EEPROM_OK)
```

```
00352         {
00353           return status;
00354         }
00355 
00356         /* Store the number of data to be wr
itten */
00357         dataindex = (NumByteToWrite - count)
;
00358         /* Write the remaining data in the f
ollowing page */
00359         status = BSP_EEPROM_WritePage((uint8
_t*)(pBuffer + count), (WriteAddr + count), (uint8
_t*)(&dataindex));
00360         if(status != EEPROM_OK)
00361         {
00362           return status;
00363         }
00364       }
00365       else
00366       {
00367         /* Store the number of data to be wr
itten */
00368         dataindex = numofsingle;
00369         status = BSP_EEPROM_WritePage(pBuffe
r, WriteAddr, (uint8_t*)(&dataindex));
00370         if(status != EEPROM_OK)
00371         {
00372           return status;
00373         }
00374       }
00375     }
00376     /* If NumByteToWrite > EEPROM_PAGESIZE */

00377     else
00378     {
00379       NumByteToWrite -= count;
00380       numofpage =  NumByteToWrite / EEPROM_P
```

```
AGESIZE;
00381       numofsingle = NumByteToWrite % EEPROM_
PAGESIZE;
00382
00383       if(count != 0)
00384       {
00385         /* Store the number of data to be wr
itten */
00386         dataindex = count;
00387         status = BSP_EEPROM_WritePage(pBuffe
r, WriteAddr, (uint8_t*)(&dataindex));
00388         if(status != EEPROM_OK)
00389         {
00390           return status;
00391         }
00392         WriteAddr += count;
00393         pBuffer += count;
00394       }
00395
00396       while(numofpage--)
00397       {
00398         /* Store the number of data to be wr
itten */
00399         dataindex = EEPROM_PAGESIZE;

00400         status = BSP_EEPROM_WritePage(pBuffe
r, WriteAddr, (uint8_t*)(&dataindex));
00401         if(status != EEPROM_OK)
00402         {
00403           return status;
00404         }
00405         WriteAddr +=  EEPROM_PAGESIZE;
00406         pBuffer += EEPROM_PAGESIZE;
00407       }
00408       if(numofsingle != 0)
00409       {
00410         /* Store the number of data to be wr
```

```
itten */
00411         dataindex = numofsingle;
00412         status = BSP_EEPROM_WritePage(pBuffe
r, WriteAddr, (uint8_t*)(&dataindex));
00413         if(status != EEPROM_OK)
00414         {
00415           return status;
00416         }
00417       }
00418     }
00419   }
00420
00421   /* If all operations OK, return EEPROM_OK
(0) */
00422   return EEPROM_OK;
00423 }
00424
00425 /**
00426   * @brief  Wait for EEPROM Standby state.
00427   *
00428   * @note  This function allows to wait and
check that EEPROM has finished the
00429   *        last operation. It is mostly used
 after Write operation: after receiving
00430   *        the buffer to be written, the EEP
ROM may need additional time to actually
00431   *        perform the write operation. Duri
ng this time, it doesn't answer to
00432   *        I2C packets addressed to it. Once
 the write operation is complete
00433   *        the EEPROM responds to its addres
s.
00434   *
00435   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00436   *         different from EEPROM_OK (0) or
the timeout user callback.
```

```
00437   */
00438 uint32_t BSP_EEPROM_WaitEepromStandbyState(v
oid)
00439 {
00440   /* Check if the maximum allowed number of
trials has bee reached */
00441   if(EEPROM_IO_IsDeviceReady(EEPROMAddress,
EEPROM_MAX_TRIALS) != HAL_OK)
00442   {
00443     /* If the maximum number of trials has b
een reached, exit the function */
00444     BSP_EEPROM_TIMEOUT_UserCallback();
00445     return EEPROM_TIMEOUT;
00446   }
00447   return EEPROM_OK;
00448 }
00449
00450 /**
00451   * @brief  Basic management of the timeout
situation.
00452   */
00453 __weak void BSP_EEPROM_TIMEOUT_UserCallback(
void)
00454 {
00455 }
00456
00457 /**
00458   * @}
00459   */
00460
00461 /**
00462   * @}
00463   */
00464
00465 /**
00466   * @}
00467   */
```

```
00468
00469 /**
00470   * @}
00471   */
00472
00473 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM32446E_EVAL BSP User Manual

stm32446e_eval_io.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval_io.c
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file provides a set of functions needed to manage the IO pins
00008   *          on STM32446E-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 IO mod
ule of the STM32446E-EVAL evaluation
00044      board.
00045    - The MFXSTM32L152 IO expander device com
ponent driver must be included with this
00046      driver in order to run the IO functiona
lities commanded by the IO expander (MFX)
00047      device mounted on the evaluation board.
00048
00049 2. Driver description:
00050 ---------------------
00051   + Initialization steps:
00052      o Initialize the IO module using the BS
P_IO_Init() function. This
00053        function includes the MSP layer hardw
are resources initialization and the
00054        communication layer configuration to
start the IO functionalities use.
00055
00056   + IO functionalities use
00057      o The IO pin mode is configured when ca
lling the function BSP_IO_ConfigPin(), you
00058        must specify the desired IO mode by c
hoosing the "IO_ModeTypedef" parameter
00059        predefined value.
00060      o If an IO pin is used in interrupt mod
```

```
e, the function BSP_IO_ITGetStatus() is
00061        needed to get the interrupt status. T
o clear the IT pending bits, you should
00062        call the function BSP_IO_ITClear() wi
th specifying the IO pending bit to clear.
00063      o The IT is handled using the correspon
ding external interrupt IRQ handler,
00064        the user IT callback treatment is imp
lemented on the same external interrupt
00065        callback.
00066      o The IRQ_OUT pin (common for all funct
ionalities: TS, JOY, SD, etc)  can be
00067        configured using the function BSP_IO_
ConfigIrqOutPin()
00068      o To get/set an IO pin combination stat
e you can use the functions
00069        BSP_IO_ReadPin()/BSP_IO_WritePin() or
 the function BSP_IO_TogglePin() to toggle the pin

00070        state.
00071
00072 ----------------------------------------
----------------------------------*/
00073
00074 /* Includes --------------------------------
----------------------------------*/
00075 #include "stm32446e_eval_io.h"
00076
00077 /** @addtogroup BSP
00078   * @{
00079   */
00080
00081 /** @addtogroup STM32446E_EVAL
00082   * @{
00083   */
00084
00085 /** @defgroup STM32446E_EVAL_IO STM32446E EV
```

```
AL IO
00086   * @{
00087   */
00088
00089 /** @defgroup STM32446E_EVAL_IO_Private_Type
s_Definitions STM32446E EVAL IO Private Types Defi
nitions
00090   * @{
00091   */
00092 /**
00093   * @}
00094   */
00095
00096 /** @defgroup STM32446E_EVAL_IO_Private_Defi
nes STM32446E EVAL IO Private Defines
00097   * @{
00098   */
00099 /**
00100   * @}
00101   */
00102
00103 /** @defgroup STM32446E_EVAL_IO_Private_Macr
os STM32446E EVAL IO Private Macros
00104   * @{
00105   */
00106 /**
00107   * @}
00108   */
00109
00110 /** @defgroup STM32446E_EVAL_IO_Private_Vari
ables STM32446E EVAL IO Private Variables
00111   * @{
00112   */
00113 static IO_DrvTypeDef *IoDrv = NULL;
00114 static  uint8_t mfxstm32l152Identifier;
00115
00116 /**
```

```
00117   * @}
00118   */
00119
00120 /** @defgroup STM32446E_EVAL_IO_Private_Func
tion_Prototypes STM32446E EVAL IO Private Function
 Prototypes
00121   * @{
00122   */
00123 /**
00124   * @}
00125   */
00126
00127 /** @defgroup STM32446E_EVAL_IO_Private_Func
tions STM32446E EVAL IO Private Functions
00128   * @{
00129   */
00130
00131 /**
00132   * @brief  Initializes and configures the I
O functionalities and configures all
00133   *         necessary hardware resources (MF
X, ...).
00134   * @note   BSP_IO_Init() is using HAL_Delay
() function to ensure that MFXSTM32L152
00135   *         IO Expander is correctly reset.
HAL_Delay() function provides accurate
00136   *         delay (in milliseconds) based on
 variable incremented in SysTick ISR.
00137   *         This implies that if BSP_IO_Init
() is called from a peripheral ISR process,
00138   *         then the SysTick interrupt must
have higher priority (numerically lower)
00139   *         than the peripheral interrupt. O
therwise the caller ISR process will be blocked.
00140   * @retval IO_OK if all initializations are
 OK. Other value if error.
00141   */
```

```
00142 uint8_t BSP_IO_Init(void)
00143 {
00144   uint8_t ret = IO_OK;
00145
00146   if (  IoDrv == NULL)
00147   {
00148     /* Read ID and verify the IO expander is
 ready */
00149     mfxstm32l152Identifier=0;
00150     mfxstm32l152Identifier = mfxstm32l152_io
_drv.ReadID(IO_I2C_ADDRESS);
00151     if((mfxstm32l152Identifier == MFXSTM32L1
52_ID_1) || (mfxstm32l152Identifier == MFXSTM32L15
2_ID_2))
00152     {
00153       /* Initialize the IO driver structure
*/
00154       IoDrv = &mfxstm32l152_io_drv;
00155     }
00156     else
00157     {
00158       ret = IO_ERROR;
00159     }
00160
00161     if(ret == IO_OK)
00162     {
00163       IoDrv->Init(IO_I2C_ADDRESS);
00164       IoDrv->Start(IO_I2C_ADDRESS, IO_PIN_ALL
);
00165     }
00166   }
00167
00168   return ret;
00169 }
00170
00171 /**
00172   * @brief  DeInit allows Mfx Initialization
```

```
 to be executed again
00173   * @note   BSP_IO_Init() has no effect if t
he IoDrv is already initialized
00174   *         BSP_IO_DeInit() allows to erase
the pointer such to allow init to be effective
00175   * @retval IO_OK
00176   */
00177 uint8_t BSP_IO_DeInit(void)
00178 {
00179   IoDrv = NULL;
00180   return IO_OK;
00181 }
00182
00183 /**
00184   * @brief  Gets the selected pins IT status.

00185   * @param  IoPin: Selected pins to check th
e status.
00186   *          This parameter can be any combi
nation of the IO pins.
00187   * @retval IO_OK if read status OK. Other v
alue if error.
00188   */
00189 uint32_t BSP_IO_ITGetStatus(uint32_t IoPin)
00190 {
00191   /* Return the IO Pin IT status */
00192   return (IoDrv->ITStatus(IO_I2C_ADDRESS, Io
Pin));
00193 }
00194
00195 /**
00196   * @brief  Clears all the IO IT pending bit
s.
00197   */
00198 void BSP_IO_ITClear(void)
00199 {
00200   /* Clear all IO IT pending bits */
```

```
00201   IoDrv->ClearIT(IO_I2C_ADDRESS, MFXSTM32L15
2_GPIO_PINS_ALL);
00202 }
00203
00204 /**
00205   * @brief  Configures the IO pin(s) accordi
ng to IO mode structure value.
00206   * @param  IoPin: IO pin(s) to be configure
d.
00207   *          This parameter can be one of th
e following values:
00208   *            @arg  MFXSTM32L152_GPIO_PIN_x
: where x can be from 0 to 23.
00209   * @param  IoMode: IO pin mode to configure
00210   *          This parameter can be one of th
e following values:
00211   *            @arg  IO_MODE_INPUT
00212   *            @arg  IO_MODE_OUTPUT
00213   *            @arg  IO_MODE_IT_RISING_EDGE
00214   *            @arg  IO_MODE_IT_FALLING_EDGE
00215   *            @arg  IO_MODE_IT_LOW_LEVEL
00216   *            @arg  IO_MODE_IT_HIGH_LEVEL

00217   *            @arg  IO_MODE_ANALOG
00218   *            @arg  IO_MODE_OFF
00219   *            @arg  IO_MODE_INPUT_PU,
00220   *            @arg  IO_MODE_INPUT_PD,
00221   *            @arg  IO_MODE_OUTPUT_OD,
00222   *            @arg  IO_MODE_OUTPUT_OD_PU,
00223   *            @arg  IO_MODE_OUTPUT_OD_PD,
00224   *            @arg  IO_MODE_OUTPUT_PP,
00225   *            @arg  IO_MODE_OUTPUT_PP_PU,
00226   *            @arg  IO_MODE_OUTPUT_PP_PD,
00227   *            @arg  IO_MODE_IT_RISING_EDGE_
PU
00228   *            @arg  IO_MODE_IT_FALLING_EDGE
_PU
```

```
00229   *             @arg  IO_MODE_IT_LOW_LEVEL_PU
00230   *             @arg  IO_MODE_IT_HIGH_LEVEL_PU

00231   *             @arg  IO_MODE_IT_RISING_EDGE_
PD
00232   *             @arg  IO_MODE_IT_FALLING_EDGE
_PD
00233   *             @arg  IO_MODE_IT_LOW_LEVEL_PD
00234   *             @arg  IO_MODE_IT_HIGH_LEVEL_PD

00235   * @retval IO_OK if all initializations are
 OK. Other value if error.
00236   */
00237 uint8_t BSP_IO_ConfigPin(uint32_t IoPin, IO_
ModeTypedef IoMode)
00238 {
00239   /* Configure the selected IO pin(s) mode */

00240   IoDrv->Config(IO_I2C_ADDRESS, IoPin, IoMod
e);
00241
00242   return IO_OK;
00243 }
00244
00245 /**
00246   * @brief  Sets the IRQ_OUT pin polarity an
d type
00247   * @param  IoIrqOutPinPolarity: High/Low
00248   * @param  IoIrqOutPinType:     OpenDrain/P
ushPull
00249   * @retval OK
00250   */
00251 uint8_t BSP_IO_ConfigIrqOutPin(uint8_t IoIrq
OutPinPolarity, uint8_t IoIrqOutPinType)
00252 {
00253   if((mfxstm32l152Identifier == MFXSTM32L152
_ID_1) || (mfxstm32l152Identifier == MFXSTM32L152_
```

```
ID_2))
00254   {
00255     /* Initialize the IO driver structure */
00256     mfxstm32l152_SetIrqOutPinPolarity(IO_I2C
_ADDRESS, IoIrqOutPinPolarity);
00257     mfxstm32l152_SetIrqOutPinType(IO_I2C_ADD
RESS, IoIrqOutPinType);
00258   }
00259
00260   return IO_OK;
00261 }
00262
00263 /**
00264   * @brief  Sets the selected pins state.
00265   * @param  IoPin: Selected pins to write.
00266   *          This parameter can be any combi
nation of the IO pins.
00267   * @param  PinState: New pins state to writ
e
00268   */
00269 void BSP_IO_WritePin(uint32_t IoPin, BSP_IO_
PinStateTypeDef PinState)
00270 {
00271   /* Set the Pin state */
00272   IoDrv->WritePin(IO_I2C_ADDRESS, IoPin, Pin
State);
00273 }
00274
00275 /**
00276   * @brief  Gets the selected pins current s
tate.
00277   * @param  IoPin: Selected pins to read.
00278   *          This parameter can be any combi
nation of the IO pins.
00279   * @retval The current pins state
00280   */
00281 uint32_t BSP_IO_ReadPin(uint32_t IoPin)
```

```
00282 {
00283   return(IoDrv->ReadPin(IO_I2C_ADDRESS, IoPin
));
00284 }
00285
00286 /**
00287   * @brief  Toggles the selected pins state.
00288   * @param  IoPin: Selected pins to toggle.
00289   *          This parameter can be any combi
nation of the IO pins.
00290   * @note   This function is only used to to
ggle one pin in the same time
00291   */
00292 void BSP_IO_TogglePin(uint32_t IoPin)
00293 {
00294   /* Toggle the current pin state */
00295   if(IoDrv->ReadPin(IO_I2C_ADDRESS, IoPin) !
= 0) /* Set */
00296   {
00297     IoDrv->WritePin(IO_I2C_ADDRESS, IoPin, 0
); /* Reset */
00298   }
00299   else
00300   {
00301     IoDrv->WritePin(IO_I2C_ADDRESS, IoPin, 1
); /* Set */
00302   }
00303 }
00304
00305 /**
00306   * @}
00307   */
00308
00309 /**
00310   * @}
00311   */
00312
```

```
00313 /**
00314   * @}
00315   */
00316
00317 /**
00318   * @}
00319   */
00320
00321 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32446E_EVAL BSP User Manual

stm32446e_eval_sd.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval_sd.h
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm32446e_eval_sd.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 __STM32446E_EVAL_SD_H
00041 #define __STM32446E_EVAL_SD_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------------------------------------------*/
00048 #include "stm32446e_eval.h"
00049 #include "stm32446e_eval_io.h"
00050
00051 /** @addtogroup BSP
00052   * @{
00053   */
00054
00055 /** @addtogroup STM32446E_EVAL
00056   * @{
00057   */
00058
00059 /** @addtogroup STM32446E_EVAL_SD
00060   * @{
00061   */
00062
00063 /** @defgroup STM32446E_EVAL_SD_Exported_Types STM32446E EVAL SD Exported Types
00064   * @{
00065   */
00066
00067 /**
00068   * @brief SD Card information structure
```

```
00069   */
00070 #define SD_CardInfo HAL_SD_CardInfoTypedef
00071 /**
00072   * @}
00073   */
00074
00075 /**
00076   * @brief  SD status structure definition
00077   */
00078 #define   MSD_OK                        ((ui
nt8_t)0x00)
00079 #define   MSD_ERROR                     ((ui
nt8_t)0x01)
00080 #define   MSD_ERROR_SD_NOT_PRESENT      ((ui
nt8_t)0x02)
00081
00082 /** @defgroup STM32446E_EVAL_SD_Exported_Con
stants STM32446E EVAL SD Exported Constants
00083   * @{
00084   */
00085 #define SD_PRESENT               ((uint8_t)0
x01)
00086 #define SD_NOT_PRESENT           ((uint8_t)0
x00)
00087
00088 #define SD_DATATIMEOUT           ((uint32_t)
100000000)
00089
00090 /* DMA definitions for SD DMA transfer */
00091 #define __DMAx_TxRx_CLK_ENABLE             __
HAL_RCC_DMA2_CLK_ENABLE
00092 #define SD_DMAx_Tx_CHANNEL                 DM
A_CHANNEL_4
00093 #define SD_DMAx_Rx_CHANNEL                 DM
A_CHANNEL_4
00094 #define SD_DMAx_Tx_STREAM                  DM
A2_Stream6
```

```
00095 #define SD_DMAx_Rx_STREAM                        DM
A2_Stream3
00096 #define SD_DMAx_Tx_IRQn                          DM
A2_Stream6_IRQn
00097 #define SD_DMAx_Rx_IRQn                          DM
A2_Stream3_IRQn
00098 #define SD_DMAx_Tx_IRQHandler                    DM
A2_Stream6_IRQHandler
00099 #define SD_DMAx_Rx_IRQHandler                    DM
A2_Stream3_IRQHandler
00100 #define SD_DetectIRQHandler()                    HA
L_GPIO_EXTI_IRQHandler(GPIO_PIN_8)
00101 /**
00102   * @}
00103   */
00104
00105 /** @defgroup STM32446E_EVAL_SD_Exported_Mac
ro STM32446E EVAL SD Exported Macro
00106   * @{
00107   */
00108 /**
00109   * @}
00110   */
00111
00112 /** @defgroup STM32446E_EVAL_SD_Exported_Fun
ctions STM32446E EVAL SD Exported Functions
00113   * @{
00114   */
00115 uint8_t BSP_SD_Init(void);
00116 uint8_t BSP_SD_DeInit(void);
00117 uint8_t BSP_SD_ITConfig(void);
00118 void    BSP_SD_DetectIT(void);
00119 void    BSP_SD_DetectCallback(void);
00120 uint8_t BSP_SD_ReadBlocks(uint32_t *pData, u
int64_t ReadAddr, uint32_t BlockSize, uint32_t Num
OfBlocks);
00121 uint8_t BSP_SD_WriteBlocks(uint32_t *pData,
```

```
uint64_t WriteAddr, uint32_t BlockSize, uint32_t N
umOfBlocks);
00122 uint8_t BSP_SD_ReadBlocks_DMA(uint32_t *pDat
a, uint64_t ReadAddr, uint32_t BlockSize, uint32_t
 NumOfBlocks);
00123 uint8_t BSP_SD_WriteBlocks_DMA(uint32_t *pDa
ta, uint64_t WriteAddr, uint32_t BlockSize, uint32
_t NumOfBlocks);
00124 uint8_t BSP_SD_Erase(uint64_t StartAddr, uin
t64_t EndAddr);
00125 void    BSP_SD_IRQHandler(void);
00126 void    BSP_SD_DMA_Tx_IRQHandler(void);
00127 void    BSP_SD_DMA_Rx_IRQHandler(void);
00128 HAL_SD_TransferStateTypedef BSP_SD_GetStatus(
void);
00129 void    BSP_SD_GetCardInfo(HAL_SD_CardInfoTy
pedef *CardInfo);
00130 uint8_t BSP_SD_IsDetected(void);
00131
00132 /* These function can be modified in case th
e current settings (e.g. DMA stream)
00133    need to be changed for specific applicati
on needs */
00134 void    BSP_SD_MspInit(SD_HandleTypeDef *hsd
, void *Params);
00135 void    BSP_SD_MspDeInit(SD_HandleTypeDef *h
sd, void *Params);
00136
00137 /**
00138   * @}
00139   */
00140
00141 /**
00142   * @}
00143   */
00144
00145 /**
```

```
00146   * @}
00147   */
00148
00149 /**
00150   * @}
00151   */
00152
00153 #ifdef __cplusplus
00154 }
00155 #endif
00156
00157 #endif /* __STM32446E_EVAL_SD_H */
00158
00159 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM32446E_EVAL BSP User Manual

stm32446e_eval_sd.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval_sd.c
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file includes the uSD card driver mounted on STM32446E-EVAL
00008   *          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 micro
SD external card mounted on STM32446E-EVAL
00044      evaluation board.
00045    - This driver does not need a specific co
mponent driver for the micro SD device
00046      to be included with.
00047
00048 2. Driver description:
00049 ---------------------
00050   + Initialization steps:
00051      o Initialize the micro SD card using th
e BSP_SD_Init() function. This
00052        function includes the MSP layer hardw
are resources initialization and the
00053        SDIO interface configuration to inter
face with the external micro SD. It
00054        also includes the micro SD initializa
tion sequence.
00055      o To check the SD card presence you can
 use the function BSP_SD_IsDetected() which
00056        returns the detection status
00057      o If SD presence detection interrupt mo
de is desired, you must configure the
00058        SD detection interrupt mode by callin
g the function BSP_SD_ITConfig(). The interrupt
00059        is generated as an external interrupt
 whenever the micro SD card is
```

```
00060         plugged/unplugged in/from the evaluat
ion board. The SD detection interrupt
00061         is handled by calling the function BS
P_SD_DetectIT() which is called in the IRQ
00062         handler file, the user callback is im
plemented in the function BSP_SD_DetectCallback().
00063       o The function BSP_SD_GetCardInfo() is
used to get the micro SD card information
00064         which is stored in the structure "HAL
_SD_CardInfoTypedef".
00065
00066     + Micro SD card operations
00067       o The micro SD card can be accessed wit
h read/write block(s) operations once
00068         it is ready for access. The access ca
n be performed whether using the polling
00069         mode by calling the functions BSP_SD_
ReadBlocks()/BSP_SD_WriteBlocks(), or by DMA
00070         transfer using the functions BSP_SD_R
eadBlocks_DMA()/BSP_SD_WriteBlocks_DMA()
00071       o The DMA transfer complete is used wit
h interrupt mode. Once the SD transfer
00072         is complete, the SD interrupt is hand
led using the function BSP_SD_IRQHandler(),
00073         the DMA Tx/Rx transfer complete are h
andled using the functions
00074         BSP_SD_DMA_Tx_IRQHandler()/BSP_SD_DMA
_Rx_IRQHandler(). The corresponding user callbacks

00075         are implemented by the user at applic
ation level.
00076       o The SD erase block(s) is performed us
ing the function BSP_SD_Erase() with specifying
00077         the number of blocks to erase.
00078       o The SD runtime status is returned whe
n calling the function BSP_SD_GetStatus().
00079
```

```
00080 ------------------------------------------
-----------------------------------*/
00081
00082 /* Includes ------------------------------
-----------------------------------*/
00083 #include "stm32446e_eval_sd.h"
00084
00085 /** @addtogroup BSP
00086   * @{
00087   */
00088
00089 /** @addtogroup STM32446E_EVAL
00090   * @{
00091   */
00092
00093 /** @defgroup STM32446E_EVAL_SD STM32446E EV
AL SD
00094   * @{
00095   */
00096
00097
00098 /** @defgroup STM32446E_EVAL_SD_Private_Type
sDefinitions STM32446E EVAL SD Private TypesDefini
tions
00099   * @{
00100   */
00101 /**
00102   * @}
00103   */
00104
00105 /** @defgroup STM32446E_EVAL_SD_Private_Defi
nes STM32446E EVAL SD Private Defines
00106   * @{
00107   */
00108 /**
00109   * @}
00110   */
```

```
00111
00112 /** @defgroup STM32446E_EVAL_SD_Private_Macr
os STM32446E EVAL SD Private Macros
00113   * @{
00114   */
00115 /**
00116   * @}
00117   */
00118
00119 /** @defgroup STM32446E_EVAL_SD_Private_Vari
ables STM32446E EVAL SD Private Variables
00120   * @{
00121   */
00122 static SD_HandleTypeDef uSdHandle;
00123 static SD_CardInfo uSdCardInfo;
00124 static uint8_t UseExtiModeDetection = 0;
00125
00126 /**
00127   * @}
00128   */
00129
00130 /** @defgroup STM32446E_EVAL_SD_Private_Func
tionPrototypes STM32446E EVAL SD Private FunctionP
rototypes
00131   * @{
00132   */
00133 /**
00134   * @}
00135   */
00136
00137 /** @defgroup STM32446E_EVAL_SD_Private_Func
tions STM32446E EVAL SD Private Functions
00138   * @{
00139   */
00140
00141 /**
00142   * @brief  Initializes the SD card device.
```

```
00143   * @retval SD status
00144   */
00145 uint8_t BSP_SD_Init(void)
00146 {
00147   uint8_t sd_state = MSD_OK;
00148
00149   /* uSD device interface configuration */
00150   uSdHandle.Instance = SDIO;
00151
00152   uSdHandle.Init.ClockEdge           = SDIO_
CLOCK_EDGE_RISING;
00153   uSdHandle.Init.ClockBypass         = SDIO_
CLOCK_BYPASS_DISABLE;
00154   uSdHandle.Init.ClockPowerSave      = SDIO_
CLOCK_POWER_SAVE_DISABLE;
00155   uSdHandle.Init.BusWide             = SDIO_
BUS_WIDE_1B;
00156   uSdHandle.Init.HardwareFlowControl = SDIO_
HARDWARE_FLOW_CONTROL_ENABLE;
00157   uSdHandle.Init.ClockDiv            = SDIO_
TRANSFER_CLK_DIV;
00158
00159   /* Initialize IO functionalities (MFX) use
d by SD detect pin */
00160   BSP_IO_Init();
00161
00162   /* Check if the SD card is plugged in the
slot */
00163   BSP_IO_ConfigPin(SD_DETECT_PIN, IO_MODE_IN
PUT);
00164   if(BSP_SD_IsDetected() != SD_PRESENT)
00165   {
00166     return MSD_ERROR_SD_NOT_PRESENT;
00167   }
00168
00169   /* Msp SD initialization */
00170   BSP_SD_MspInit(&uSdHandle, NULL);
```

```
00171
00172   /* HAL SD initialization */
00173   if(HAL_SD_Init(&uSdHandle, &uSdCardInfo) !
= SD_OK)
00174   {
00175     sd_state = MSD_ERROR;
00176   }
00177
00178   /* Configure SD Bus width */
00179   if(sd_state == MSD_OK)
00180   {
00181     /* Enable wide operation */
00182     if(HAL_SD_WideBusOperation_Config(&uSdHa
ndle, SDIO_BUS_WIDE_4B) != SD_OK)
00183     {
00184       sd_state = MSD_ERROR;
00185     }
00186     else
00187     {
00188       sd_state = MSD_OK;
00189     }
00190   }
00191
00192   return  sd_state;
00193 }
00194
00195 /**
00196   * @brief  DeInitializes the SD card device.

00197   * @retval SD status
00198   */
00199 uint8_t BSP_SD_DeInit(void)
00200 {
00201   uint8_t sd_state = MSD_OK;
00202
00203   uSdHandle.Instance = SDIO;
00204
```

```
00205   /* Set back Mfx pin to INPUT mode in case
it was in exti */
00206   UseExtiModeDetection = 0;
00207   BSP_IO_ConfigPin(SD_DETECT_PIN, IO_MODE_IN
PUT);
00208
00209   /* HAL SD deinitialization */
00210   if(HAL_SD_DeInit(&uSdHandle) != HAL_OK)
00211   {
00212     sd_state = MSD_ERROR;
00213   }
00214
00215   /* Msp SD deinitialization */
00216   uSdHandle.Instance = SDIO;
00217   BSP_SD_MspDeInit(&uSdHandle, NULL);
00218
00219   return  sd_state;
00220 }
00221
00222 /**
00223   * @brief  Configures Interrupt mode for SD
 detection pin.
00224   * @retval Returns 0
00225   */
00226 uint8_t BSP_SD_ITConfig(void)
00227 {
00228   /* Configure Interrupt mode for SD detecti
on pin */
00229   /* Note: disabling exti mode can be done c
alling SD_DeInit() */
00230   UseExtiModeDetection = 1;
00231   BSP_SD_IsDetected();
00232
00233   return 0;
00234 }
00235
00236 /**
```

```
00237  * @brief  Detects if SD card is correctly p
lugged in the memory slot or not.
00238  * @retval Returns if SD is detected or not
00239  */
00240 uint8_t BSP_SD_IsDetected(void)
00241 {
00242   __IO uint8_t status = SD_PRESENT;
00243
00244   /* Check SD card detect pin */
00245   if((BSP_IO_ReadPin(SD_DETECT_PIN)&SD_DETEC
T_PIN) != SD_DETECT_PIN)
00246   {
00247      if (UseExtiModeDetection)
00248      {
00249       BSP_IO_ConfigPin(SD_DETECT_PIN, IO_MOD
E_IT_RISING_EDGE);
00250      }
00251
00252   }
00253   else
00254   {
00255     status = SD_NOT_PRESENT;
00256     if (UseExtiModeDetection)
00257     {
00258       BSP_IO_ConfigPin(SD_DETECT_PIN, IO_MOD
E_IT_FALLING_EDGE);
00259     }
00260   }
00261
00262   return status;
00263 }
00264
00265
00266
00267 /**
00268   * @brief  Reads block(s) from a specified
address in an SD card, in polling 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(uint32_t *pData, u
int64_t ReadAddr, uint32_t BlockSize, uint32_t Num
OfBlocks)
00276 {
00277   if(HAL_SD_ReadBlocks(&uSdHandle, pData, Re
adAddr, BlockSize, NumOfBlocks) != SD_OK)
00278   {
00279     return MSD_ERROR;
00280   }
00281   else
00282   {
00283     return MSD_OK;
00284   }
00285 }
00286
00287 /**
00288   * @brief  Writes block(s) to a specified a
ddress in an SD card, in polling mode.
00289   * @param  pData: Pointer to the buffer tha
t will contain the data to transmit
00290   * @param  WriteAddr: Address from where da
ta is to be written
00291   * @param  BlockSize: SD card data block si
ze, that should be 512
00292   * @param  NumOfBlocks: Number of SD blocks
 to write
00293   * @retval SD status
```

```
00294   */
00295 uint8_t BSP_SD_WriteBlocks(uint32_t *pData,
uint64_t WriteAddr, uint32_t BlockSize, uint32_t N
umOfBlocks)
00296 {
00297   if(HAL_SD_WriteBlocks(&uSdHandle, pData, W
riteAddr, BlockSize, NumOfBlocks) != SD_OK)
00298   {
00299     return MSD_ERROR;
00300   }
00301   else
00302   {
00303     return MSD_OK;
00304   }
00305 }
00306
00307 /**
00308   * @brief  Reads block(s) from a specified
address in an SD card, in DMA mode.
00309   * @param  pData: Pointer to the buffer tha
t will contain the data to transmit
00310   * @param  ReadAddr: Address from where dat
a is to be read
00311   * @param  BlockSize: SD card data block si
ze, that should be 512
00312   * @param  NumOfBlocks: Number of SD blocks
 to read
00313   * @retval SD status
00314   */
00315 uint8_t BSP_SD_ReadBlocks_DMA(uint32_t *pDat
a, uint64_t ReadAddr, uint32_t BlockSize, uint32_t
 NumOfBlocks)
00316 {
00317   uint8_t sd_state = MSD_OK;
00318
00319   /* Read block(s) in DMA transfer mode */
00320   if(HAL_SD_ReadBlocks_DMA(&uSdHandle, pData
```

```
, ReadAddr, BlockSize, NumOfBlocks) != SD_OK)
00321   {
00322     sd_state = MSD_ERROR;
00323   }
00324
00325   /* Wait until transfer is complete */
00326   if(sd_state == MSD_OK)
00327   {
00328     if(HAL_SD_CheckReadOperation(&uSdHandle,
 (uint32_t)SD_DATATIMEOUT) != SD_OK)
00329     {
00330       sd_state = MSD_ERROR;
00331     }
00332     else
00333     {
00334       sd_state = MSD_OK;
00335     }
00336   }
00337
00338   return sd_state;
00339 }
00340
00341 /**
00342   * @brief  Writes block(s) to a specified a
ddress in an SD card, in DMA mode.
00343   * @param  pData: Pointer to the buffer tha
t will contain the data to transmit
00344   * @param  WriteAddr: Address from where da
ta is to be written
00345   * @param  BlockSize: SD card data block si
ze, that should be 512
00346   * @param  NumOfBlocks: Number of SD blocks
 to write
00347   * @retval SD status
00348   */
00349 uint8_t BSP_SD_WriteBlocks_DMA(uint32_t *pDa
ta, uint64_t WriteAddr, uint32_t BlockSize, uint32
```

```
_t NumOfBlocks)
00350 {
00351   uint8_t sd_state = MSD_OK;
00352
00353   /* Write block(s) in DMA transfer mode */
00354   if(HAL_SD_WriteBlocks_DMA(&uSdHandle, pDat
a, WriteAddr, BlockSize, NumOfBlocks) != SD_OK)
00355   {
00356     sd_state = MSD_ERROR;
00357   }
00358
00359   /* Wait until transfer is complete */
00360   if(sd_state == MSD_OK)
00361   {
00362     if(HAL_SD_CheckWriteOperation(&uSdHandle
, (uint32_t)SD_DATATIMEOUT) != SD_OK)
00363     {
00364       sd_state = MSD_ERROR;
00365     }
00366     else
00367     {
00368       sd_state = MSD_OK;
00369     }
00370   }
00371
00372   return sd_state;
00373 }
00374
00375 /**
00376   * @brief  Erases the specified memory area
 of the given SD card.
00377   * @param  StartAddr: Start byte address
00378   * @param  EndAddr: End byte address
00379   * @retval SD status
00380   */
00381 uint8_t BSP_SD_Erase(uint64_t StartAddr, uin
t64_t EndAddr)
```

```
00382 {
00383   if(HAL_SD_Erase(&uSdHandle, StartAddr, End
Addr) != SD_OK)
00384   {
00385     return MSD_ERROR;
00386   }
00387   else
00388   {
00389     return MSD_OK;
00390   }
00391 }
00392
00393 /**
00394   * @brief  Initializes the SD MSP.
00395   * @param  hsd: SD handle
00396   * @param  Params: pointer on additional co
nfiguration parameters, can be NULL.
00397   */
00398 __weak void BSP_SD_MspInit(SD_HandleTypeDef
*hsd, void *Params)
00399 {
00400   static DMA_HandleTypeDef dma_rx_handle;
00401   static DMA_HandleTypeDef dma_tx_handle;
00402   GPIO_InitTypeDef gpio_init_structure;
00403
00404   /* SD pins are in conflict with Camera pin
s therefore Camera is power down */
00405   /* __weak function can be modified by the
application */
00406   BSP_IO_ConfigPin(RSTI_PIN, IO_MODE_OUTPUT)
;
00407   BSP_IO_ConfigPin(XSDN_PIN, IO_MODE_OUTPUT)
;
00408   /* De-assert the camera STANDBY pin (activ
e high) */
00409   BSP_IO_WritePin(XSDN_PIN, BSP_IO_PIN_RESET
);
```

```
00410   /* Assert the camera RSTI pin (active low)
 */
00411   BSP_IO_WritePin(RSTI_PIN, BSP_IO_PIN_RESET
);
00412   HAL_Delay(100);
00413
00414   /* Enable SDIO clock */
00415   __HAL_RCC_SDIO_CLK_ENABLE();
00416
00417   /* Enable DMA2 clocks */
00418   __DMAx_TxRx_CLK_ENABLE();
00419
00420   /* Enable GPIOs clock */
00421   __HAL_RCC_GPIOC_CLK_ENABLE();
00422   __HAL_RCC_GPIOD_CLK_ENABLE();
00423
00424   /* Common GPIO configuration */
00425   gpio_init_structure.Mode      = GPIO_MODE_
AF_PP;
00426   gpio_init_structure.Pull      = GPIO_PULLU
P;
00427   gpio_init_structure.Speed     = GPIO_SPEED
_HIGH;
00428   gpio_init_structure.Alternate = GPIO_AF12_
SDIO;
00429
00430   /* GPIOC configuration */
00431   gpio_init_structure.Pin = GPIO_PIN_8 | GPI
O_PIN_9 | GPIO_PIN_10 | GPIO_PIN_11 | GPIO_PIN_12;
00432
00433   HAL_GPIO_Init(GPIOC, &gpio_init_structure)
;
00434
00435   /* GPIOD configuration */
00436   gpio_init_structure.Pin = GPIO_PIN_2;
00437   HAL_GPIO_Init(GPIOD, &gpio_init_structure)
;
```

```
00438
00439   /* NVIC configuration for SDIO interrupts
*/
00440   HAL_NVIC_SetPriority(SDIO_IRQn, 5, 0);
00441   HAL_NVIC_EnableIRQ(SDIO_IRQn);
00442
00443   /* Configure DMA Rx parameters */
00444   dma_rx_handle.Init.Channel             = S
D_DMAx_Rx_CHANNEL;
00445   dma_rx_handle.Init.Direction           = D
MA_PERIPH_TO_MEMORY;
00446   dma_rx_handle.Init.PeriphInc           = D
MA_PINC_DISABLE;
00447   dma_rx_handle.Init.MemInc              = D
MA_MINC_ENABLE;
00448   dma_rx_handle.Init.PeriphDataAlignment = D
MA_PDATAALIGN_WORD;
00449   dma_rx_handle.Init.MemDataAlignment    = D
MA_MDATAALIGN_WORD;
00450   dma_rx_handle.Init.Mode                = D
MA_PFCTRL;
00451   dma_rx_handle.Init.Priority            = D
MA_PRIORITY_VERY_HIGH;
00452   dma_rx_handle.Init.FIFOMode            = D
MA_FIFOMODE_ENABLE;
00453   dma_rx_handle.Init.FIFOThreshold       = D
MA_FIFO_THRESHOLD_FULL;
00454   dma_rx_handle.Init.MemBurst            = D
MA_MBURST_INC4;
00455   dma_rx_handle.Init.PeriphBurst         = D
MA_PBURST_INC4;
00456
00457   dma_rx_handle.Instance = SD_DMAx_Rx_STREAM
;
00458
00459   /* Associate the DMA handle */
00460   __HAL_LINKDMA(hsd, hdmarx, dma_rx_handle);
```

```
00461
00462   /* Deinitialize the stream for new transfe
r */
00463   HAL_DMA_DeInit(&dma_rx_handle);
00464
00465   /* Configure the DMA stream */
00466   HAL_DMA_Init(&dma_rx_handle);
00467
00468   /* Configure DMA Tx parameters */
00469   dma_tx_handle.Init.Channel             = S
D_DMAx_Tx_CHANNEL;
00470   dma_tx_handle.Init.Direction           = D
MA_MEMORY_TO_PERIPH;
00471   dma_tx_handle.Init.PeriphInc           = D
MA_PINC_DISABLE;
00472   dma_tx_handle.Init.MemInc              = D
MA_MINC_ENABLE;
00473   dma_tx_handle.Init.PeriphDataAlignment = D
MA_PDATAALIGN_WORD;
00474   dma_tx_handle.Init.MemDataAlignment    = D
MA_MDATAALIGN_WORD;
00475   dma_tx_handle.Init.Mode                = D
MA_PFCTRL;
00476   dma_tx_handle.Init.Priority            = D
MA_PRIORITY_VERY_HIGH;
00477   dma_tx_handle.Init.FIFOMode            = D
MA_FIFOMODE_ENABLE;
00478   dma_tx_handle.Init.FIFOThreshold       = D
MA_FIFO_THRESHOLD_FULL;
00479   dma_tx_handle.Init.MemBurst            = D
MA_MBURST_INC4;
00480   dma_tx_handle.Init.PeriphBurst         = D
MA_PBURST_INC4;
00481
00482   dma_tx_handle.Instance = SD_DMAx_Tx_STREAM
;
00483
```

```
00484   /* Associate the DMA handle */
00485   __HAL_LINKDMA(hsd, hdmatx, dma_tx_handle);
00486
00487   /* Deinitialize the stream for new transfe
r */
00488   HAL_DMA_DeInit(&dma_tx_handle);
00489
00490   /* Configure the DMA stream */
00491   HAL_DMA_Init(&dma_tx_handle);
00492
00493   /* NVIC configuration for DMA transfer com
plete interrupt */
00494   HAL_NVIC_SetPriority(SD_DMAx_Rx_IRQn, 6, 0
);
00495   HAL_NVIC_EnableIRQ(SD_DMAx_Rx_IRQn);
00496
00497   /* NVIC configuration for DMA transfer com
plete interrupt */
00498   HAL_NVIC_SetPriority(SD_DMAx_Tx_IRQn, 6, 0
);
00499   HAL_NVIC_EnableIRQ(SD_DMAx_Tx_IRQn);
00500 }
00501
00502 /**
00503   * @brief  DeInitializes the SD MSP.
00504   * @param  hsd: SD handle
00505   * @param  Params: pointer on additional co
nfiguration parameters, can be NULL.
00506   */
00507 __weak void BSP_SD_MspDeInit(SD_HandleTypeDe
f *hsd, void *Params)
00508 {
00509     static DMA_HandleTypeDef dma_rx_handle;
00510     static DMA_HandleTypeDef dma_tx_handle;
00511
00512     /* Disable NVIC for DMA transfer complet
e interrupts */
```

```
00513     HAL_NVIC_DisableIRQ(SD_DMAx_Rx_IRQn);
00514     HAL_NVIC_DisableIRQ(SD_DMAx_Tx_IRQn);
00515
00516     /* Deinitialize the stream for new trans
fer */
00517     dma_rx_handle.Instance = SD_DMAx_Rx_STRE
AM;
00518     HAL_DMA_DeInit(&dma_rx_handle);
00519
00520     /* Deinitialize the stream for new trans
fer */
00521     dma_tx_handle.Instance = SD_DMAx_Tx_STRE
AM;
00522     HAL_DMA_DeInit(&dma_tx_handle);
00523
00524     /* Disable NVIC for SDIO interrupts */
00525     HAL_NVIC_DisableIRQ(SDIO_IRQn);
00526
00527     /* DeInit GPIO pins can be done in the a
pplication
00528        (by surcharging this __weak function)
 */
00529
00530     /* Disable SDIO clock */
00531     __HAL_RCC_SDIO_CLK_DISABLE();
00532
00533     /* GPIO pins clock and DMA clocks can be
 shut down in the application
00534        by surcharging this __weak function */

00535 }
00536
00537 /**
00538   * @brief  Handles SD card interrupt reques
t.
00539   */
00540 void BSP_SD_IRQHandler(void)
```

```
00541 {
00542   HAL_SD_IRQHandler(&uSdHandle);
00543 }
00544
00545 /**
00546   * @brief  Handles SD DMA Tx transfer inter
rupt request.
00547   */
00548 void BSP_SD_DMA_Tx_IRQHandler(void)
00549 {
00550   HAL_DMA_IRQHandler(uSdHandle.hdmatx);
00551 }
00552
00553 /**
00554   * @brief  Handles SD DMA Rx transfer inter
rupt request.
00555   */
00556 void BSP_SD_DMA_Rx_IRQHandler(void)
00557 {
00558   HAL_DMA_IRQHandler(uSdHandle.hdmarx);
00559 }
00560
00561 /**
00562   * @brief  Gets the current SD card data st
atus.
00563   * @retval Data transfer state.
00564   *          This value can be one of the fo
llowing values:
00565   *            @arg  SD_TRANSFER_OK: No data
 transfer is acting
00566   *            @arg  SD_TRANSFER_BUSY: Data
transfer is acting
00567   *            @arg  SD_TRANSFER_ERROR: Data
 transfer error
00568   */
00569 HAL_SD_TransferStateTypedef BSP_SD_GetStatus(
void)
```

```
00570 {
00571   return(HAL_SD_GetStatus(&uSdHandle));
00572 }
00573
00574 /**
00575   * @brief  Get SD information about specifi
c SD card.
00576   * @param  CardInfo: Pointer to HAL_SD_Card
InfoTypedef structure
00577   */
00578 void BSP_SD_GetCardInfo(HAL_SD_CardInfoTyped
ef *CardInfo)
00579 {
00580   /* Get SD card Information */
00581   HAL_SD_Get_CardInfo(&uSdHandle, CardInfo);
00582 }
00583
00584 /**
00585   * @}
00586   */
00587
00588 /**
00589   * @}
00590   */
00591
00592 /**
00593   * @}
00594   */
00595
00596 /**
00597   * @}
00598   */
00599
00600 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

Generated on Fri Jan 15 2016 10:06:21 for STM32446E_EVAL BSP
User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

stm32446e_eval_sdram.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval_sdram.h
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm32446e_eval_sdram.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 __STM32446E_EVAL_SDRAM_H
00041 #define __STM32446E_EVAL_SDRAM_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------
-----------------------------------*/
00048 #include "stm32f4xx_hal.h"
00049
00050 /** @addtogroup BSP
00051   * @{
00052   */
00053
00054 /** @addtogroup STM32446E_EVAL
00055   * @{
00056   */
00057
00058 /** @defgroup STM32446E_EVAL_SDRAM STM32446E
 EVAL SDRAM
00059   * @{
00060   */
00061
00062 /** @defgroup STM32446E_EVAL_SDRAM_Exported_
Types STM32446E EVAL SDRAM Exported Types
00063   * @{
00064   */
00065
00066 /**
```

```
00067   * @brief  SDRAM status structure definitio
n
00068   */
00069 #define   SDRAM_OK         ((uint8_t)0x00)
00070 #define   SDRAM_ERROR      ((uint8_t)0x01)
00071
00072 /**
00073   * @}
00074   */
00075
00076 /** @defgroup STM32446E_EVAL_SDRAM_Exported_
Constants STM32446E EVAL SDRAM Exported Constants
00077   * @{
00078   */
00079 #define SDRAM_DEVICE_ADDR  ((uint32_t)0xC000
0000)
00080 #define SDRAM_DEVICE_SIZE  ((uint32_t)0x8000
00)  /* SDRAM device size in MBytes */
00081
00082 /* #define SDRAM_MEMORY_WIDTH            FMC
_SDRAM_MEM_BUS_WIDTH_8  */
00083 #define SDRAM_MEMORY_WIDTH            FMC_SD
RAM_MEM_BUS_WIDTH_16
00084
00085 #define SDCLOCK_PERIOD                FMC
_SDRAM_CLOCK_PERIOD_2
00086 /* #define SDCLOCK_PERIOD                FMC
_SDRAM_CLOCK_PERIOD_3 */
00087
00088 #define REFRESH_COUNT                 ((u
int32_t)0x0569)   /* SDRAM refresh counter (90Mhz
SD clock) */
00089
00090 #define SDRAM_TIMEOUT     ((uint32_t)0xFFFF)

00091
00092 /* DMA definitions for SDRAM DMA transfer */
```

```
00093 #define __DMAx_CLK_ENABLE                   __
HAL_RCC_DMA2_CLK_ENABLE
00094 #define __DMAx_CLK_DISABLE                  __
HAL_RCC_DMA2_CLK_DISABLE
00095 #define SDRAM_DMAx_CHANNEL                  DM
A_CHANNEL_0
00096 #define SDRAM_DMAx_STREAM                   DM
A2_Stream0
00097 #define SDRAM_DMAx_IRQn                     DM
A2_Stream0_IRQn
00098 #define SDRAM_DMAx_IRQHandler               DM
A2_Stream0_IRQHandler
00099
00100
00101 /**
00102   * @brief  FMC SDRAM Mode definition regist
er defines
00103   */
00104 #define SDRAM_MODEREG_BURST_LENGTH_1
     ((uint16_t)0x0000)
00105 #define SDRAM_MODEREG_BURST_LENGTH_2
     ((uint16_t)0x0001)
00106 #define SDRAM_MODEREG_BURST_LENGTH_4
     ((uint16_t)0x0002)
00107 #define SDRAM_MODEREG_BURST_LENGTH_8
     ((uint16_t)0x0004)
00108 #define SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL
     ((uint16_t)0x0000)
00109 #define SDRAM_MODEREG_BURST_TYPE_INTERLEAVED
     ((uint16_t)0x0008)
00110 #define SDRAM_MODEREG_CAS_LATENCY_2
     ((uint16_t)0x0020)
00111 #define SDRAM_MODEREG_CAS_LATENCY_3
     ((uint16_t)0x0030)
00112 #define SDRAM_MODEREG_OPERATING_MODE_STANDAR
D    ((uint16_t)0x0000)
00113 #define SDRAM_MODEREG_WRITEBURST_MODE_PROGRA
```

```
MMED ((uint16_t)0x0000)
00114 #define SDRAM_MODEREG_WRITEBURST_MODE_SINGLE
     ((uint16_t)0x0200)
00115 /**
00116   * @}
00117   */
00118
00119 /** @defgroup STM32446E_EVAL_SDRAM_Exported_
Macro STM32446E EVAL SDRAM Exported Macro
00120   * @{
00121   */
00122 /**
00123   * @}
00124   */
00125
00126 /** @defgroup STM32446E_EVAL_SDRAM_Exported_
Functions STM32446E EVAL SDRAM Exported Functions
00127   * @{
00128   */
00129 uint8_t BSP_SDRAM_Init(void);
00130 uint8_t BSP_SDRAM_DeInit(void);
00131 void    BSP_SDRAM_Initialization_sequence(ui
nt32_t RefreshCount);
00132 uint8_t BSP_SDRAM_ReadData(uint32_t uwStartA
ddress, uint32_t *pData, uint32_t uwDataSize);
00133 uint8_t BSP_SDRAM_ReadData_DMA(uint32_t uwSt
artAddress, uint32_t *pData, uint32_t uwDataSize);
00134 uint8_t BSP_SDRAM_WriteData(uint32_t uwStart
Address, uint32_t *pData, uint32_t uwDataSize);
00135 uint8_t BSP_SDRAM_WriteData_DMA(uint32_t uwS
tartAddress, uint32_t *pData, uint32_t uwDataSize)
;
00136 uint8_t BSP_SDRAM_Sendcmd(FMC_SDRAM_CommandT
ypeDef *SdramCmd);
00137 void    BSP_SDRAM_DMA_IRQHandler(void);
00138
00139 /* These function can be modified in case th
```

```
e current settings (e.g. DMA stream)
00140    need to be changed for specific applicati
on needs */
00141 void    BSP_SDRAM_MspInit(SDRAM_HandleTypeDe
f  *hsdram, void *Params);
00142 void    BSP_SDRAM_MspDeInit(SDRAM_HandleType
Def  *hsdram, void *Params);
00143
00144
00145 /**
00146   * @}
00147   */
00148
00149 /**
00150   * @}
00151   */
00152
00153 /**
00154   * @}
00155   */
00156
00157 /**
00158   * @}
00159   */
00160
00161 #ifdef __cplusplus
00162 }
00163 #endif
00164
00165 #endif /* __STM32446E_EVAL_SDRAM_H */
00166
00167 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM32446E_EVAL BSP User Manual

stm32446e_eval_sdram.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm32446e_eval_sdram.c
00004   * @author  MCD Application Team
00005   * @version V1.1.1
00006   * @date    13-January-2016
00007   * @brief   This file includes the SDRAM driver for the MT48LC4M32B2B5-7 memory
00008   *          device mounted on STM32446E-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 MT48LC
4M32B2B5-7 SDRAM external memory mounted
00044      on STM32446E-EVAL evaluation board.
00045    - This driver does not need a specific co
mponent driver for the SDRAM device
00046      to be included with.
00047
00048 2. Driver description:
00049 ---------------------
00050   + Initialization steps:
00051      o Initialize the SDRAM external memory
using the BSP_SDRAM_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 SDRAM memory.
00054      o It contains the SDRAM initialization
sequence to program the SDRAM external
00055        device using the function BSP_SDRAM_I
nitialization_sequence(). Note that this
00056        sequence is standard for all SDRAM de
vices, but can include some differences
00057        from a device to another. If it is th
e case, the right sequence should be
00058        implemented separately.
00059
```

```
00060   + SDRAM read/write operations
00061       o SDRAM external memory can be accessed
 with read/write operations once it is
00062         initialized.
00063         Read/write operation can be performed
 with AHB access using the functions
00064         BSP_SDRAM_ReadData()/BSP_SDRAM_WriteD
ata(), or by DMA transfer using the functions
00065         BSP_SDRAM_ReadData_DMA()/BSP_SDRAM_Wr
iteData_DMA().
00066       o The AHB access is performed with 16-b
it width transaction, the DMA transfer
00067         configuration is fixed at single (no
 burst) halfword transfer (see the
00068         BSP_SDRAM_MspInit() __weak function).
00069       o User can implement his own functions
 for read/write access with his desired
00070         configurations.
00071       o If interrupt mode is used for DMA tra
nsfer, the function BSP_SDRAM_DMA_IRQHandler()
00072         is called in IRQ handler file, to ser
ve the generated interrupt once the DMA
00073         transfer is complete.
00074       o You can send a command to the SDRAM d
evice in runtime using the function
00075         BSP_SDRAM_Sendcmd(), and giving the d
esired command as parameter chosen between
00076         the predefined commands of the "FMC_S
DRAM_CommandTypeDef" structure.
00077
00078 ------------------------------------------
-----------------------------------*/
00079
00080 /* Includes ---------------------------------
-----------------------------------*/
00081 #include "stm32446e_eval_sdram.h"
00082
```

```
00083 /** @addtogroup BSP
00084   * @{
00085   */
00086
00087 /** @addtogroup STM32446E_EVAL
00088   * @{
00089   */
00090
00091 /** @defgroup STM32446E_EVAL_SDRAM STM32446E
 EVAL SDRAM
00092   * @{
00093   */
00094
00095 /** @defgroup STM32446E_EVAL_SDRAM_Private_T
ypes_Definitions STM32446E EVAL SDRAM Private Type
s Definitions
00096   * @{
00097   */
00098 /**
00099   * @}
00100   */
00101
00102 /** @defgroup STM32446E_EVAL_SDRAM_Private_D
efines STM32446E EVAL SDRAM Private Defines
00103   * @{
00104   */
00105 /**
00106   * @}
00107   */
00108
00109 /** @defgroup STM32446E_EVAL_SDRAM_Private_M
acros STM32446E EVAL SDRAM Private Macros
00110   * @{
00111   */
00112 /**
00113   * @}
00114   */
```

```
00115
00116 /** @defgroup STM32446E_EVAL_SDRAM_Private_V
ariables STM32446E EVAL SDRAM Private Variables
00117   * @{
00118   */
00119 static SDRAM_HandleTypeDef sdramHandle;
00120 static FMC_SDRAM_TimingTypeDef Timing;
00121 static FMC_SDRAM_CommandTypeDef Command;
00122 /**
00123   * @}
00124   */
00125
00126 /** @defgroup STM32446E_EVAL_SDRAM_Private_F
unction_Prototypes STM32446E EVAL SDRAM Private Fu
nction Prototypes
00127   * @{
00128   */
00129 /**
00130   * @}
00131   */
00132
00133 /** @defgroup STM32446E_EVAL_SDRAM_Private_F
unctions STM32446E EVAL SDRAM Private Functions
00134   * @{
00135   */
00136
00137 /**
00138   * @brief  Initializes the SDRAM device.
00139   * @retval SDRAM status
00140   */
00141 uint8_t BSP_SDRAM_Init(void)
00142 {
00143   static uint8_t sdramstatus = SDRAM_ERROR;
00144   /* SDRAM device configuration */
00145   sdramHandle.Instance = FMC_SDRAM_DEVICE;
00146
00147   /* Timing configuration for 90Mhz as SD cl
```

```
ock frequency (System clock is up to 180Mhz */
00148   Timing.LoadToActiveDelay    = 2;
00149   Timing.ExitSelfRefreshDelay = 7;
00150   Timing.SelfRefreshTime      = 4;
00151   Timing.RowCycleDelay        = 7;
00152   Timing.WriteRecoveryTime    = 2;
00153   Timing.RPDelay              = 2;
00154   Timing.RCDDelay             = 2;
00155
00156   sdramHandle.Init.SDBank             = FMC_
SDRAM_BANK1;
00157   sdramHandle.Init.ColumnBitsNumber   = FMC_
SDRAM_COLUMN_BITS_NUM_8;
00158   sdramHandle.Init.RowBitsNumber      = FMC_
SDRAM_ROW_BITS_NUM_12;
00159   sdramHandle.Init.MemoryDataWidth    = SDRA
M_MEMORY_WIDTH;
00160   sdramHandle.Init.InternalBankNumber = FMC_
SDRAM_INTERN_BANKS_NUM_4;
00161   sdramHandle.Init.CASLatency         = FMC_
SDRAM_CAS_LATENCY_2;
00162   sdramHandle.Init.WriteProtection    = FMC_
SDRAM_WRITE_PROTECTION_DISABLE;
00163   sdramHandle.Init.SDClockPeriod      = SDCL
OCK_PERIOD;
00164   sdramHandle.Init.ReadBurst          = FMC_
SDRAM_RBURST_ENABLE;
00165   sdramHandle.Init.ReadPipeDelay      = FMC_
SDRAM_RPIPE_DELAY_0;
00166
00167   /* SDRAM controller initialization */
00168
00169   BSP_SDRAM_MspInit(&sdramHandle, NULL); /*
__weak function can be rewritten by the applic */
00170
00171   if(HAL_SDRAM_Init(&sdramHandle, &Timing) !
= HAL_OK)
```

```
00172   {
00173     sdramstatus = SDRAM_ERROR;
00174   }
00175   else
00176   {
00177     sdramstatus = SDRAM_OK;
00178   }
00179
00180   /* SDRAM initialization sequence */
00181   BSP_SDRAM_Initialization_sequence(REFRESH_
COUNT);
00182
00183   return sdramstatus;
00184 }
00185
00186 /**
00187   * @brief  DeInitializes the SDRAM device.
00188   * @retval SDRAM status
00189   */
00190 uint8_t BSP_SDRAM_DeInit(void)
00191 {
00192   static uint8_t sdramstatus = SDRAM_ERROR;
00193   /* SDRAM device configuration */
00194   sdramHandle.Instance = FMC_SDRAM_DEVICE;
00195
00196   if(HAL_SDRAM_DeInit(&sdramHandle) != HAL_O
K)
00197   {
00198     sdramstatus = SDRAM_ERROR;
00199   }
00200   else
00201   {
00202     sdramstatus = SDRAM_OK;
00203   }
00204
00205   /* SDRAM controller initialization */
00206   BSP_SDRAM_MspDeInit(&sdramHandle, NULL);
```

```
00207
00208   return sdramstatus;
00209 }
00210
00211 /**
00212   * @brief  Programs the SDRAM device.
00213   * @param  RefreshCount: SDRAM refresh coun
ter value
00214   */
00215 void BSP_SDRAM_Initialization_sequence(uint3
2_t RefreshCount)
00216 {
00217   __IO uint32_t tmpmrd = 0;
00218
00219   /* Step 1: Configure a clock configuration
 enable command */
00220   Command.CommandMode            = FMC_SDRAM
_CMD_CLK_ENABLE;
00221   Command.CommandTarget          = FMC_SDRAM
_CMD_TARGET_BANK1;
00222   Command.AutoRefreshNumber      = 1;
00223   Command.ModeRegisterDefinition = 0;
00224
00225   /* Send the command */
00226   HAL_SDRAM_SendCommand(&sdramHandle, &Comma
nd, SDRAM_TIMEOUT);
00227
00228   /* Step 2: Insert 100 us minimum delay */
00229   /* Inserted delay is equal to 1 ms due to
systick time base unit (ms) */
00230   HAL_Delay(1);
00231
00232   /* Step 3: Configure a PALL (precharge all
) command */
00233   Command.CommandMode            = FMC_SDRAM
_CMD_PALL;
00234   Command.CommandTarget          = FMC_SDRAM
```

```
_CMD_TARGET_BANK1;
00235   Command.AutoRefreshNumber      = 1;
00236   Command.ModeRegisterDefinition = 0;
00237
00238   /* Send the command */
00239   HAL_SDRAM_SendCommand(&sdramHandle, &Comma
nd, SDRAM_TIMEOUT);
00240
00241   /* Step 4: Configure an Auto Refresh comma
nd */
00242   Command.CommandMode            = FMC_SDRAM
_CMD_AUTOREFRESH_MODE;
00243   Command.CommandTarget          = FMC_SDRAM
_CMD_TARGET_BANK1;
00244   Command.AutoRefreshNumber      = 8;
00245   Command.ModeRegisterDefinition = 0;
00246
00247   /* Send the command */
00248   HAL_SDRAM_SendCommand(&sdramHandle, &Comma
nd, SDRAM_TIMEOUT);
00249
00250   /* Step 5: Program the external memory mod
e register */
00251   tmpmrd = (uint32_t)SDRAM_MODEREG_BURST_LEN
GTH_1          |\
00252                      SDRAM_MODEREG_BURST_TYP
E_SEQUENTIAL   |\
00253                      SDRAM_MODEREG_CAS_LATEN
CY_2           |\
00254                      SDRAM_MODEREG_OPERATING
_MODE_STANDARD |\
00255                      SDRAM_MODEREG_WRITEBURS
T_MODE_SINGLE;
00256
00257   Command.CommandMode            = FMC_SDRAM
_CMD_LOAD_MODE;
00258   Command.CommandTarget          = FMC_SDRAM
```

```
_CMD_TARGET_BANK1;
00259   Command.AutoRefreshNumber      = 1;
00260   Command.ModeRegisterDefinition = tmpmrd;
00261
00262   /* Send the command */
00263   HAL_SDRAM_SendCommand(&sdramHandle, &Comma
nd, SDRAM_TIMEOUT);
00264
00265   /* Step 6: Set the refresh rate counter */
00266   /* Set the device refresh rate */
00267   HAL_SDRAM_ProgramRefreshRate(&sdramHandle,
 RefreshCount);
00268 }
00269
00270 /**
00271   * @brief  Reads an mount of data from the
SDRAM memory in polling mode.
00272   * @param  uwStartAddress: Read start addre
ss
00273   * @param  pData: Pointer to data to be rea
d
00274   * @param  uwDataSize: Size of read data fr
om the memory
00275   * @retval SDRAM status
00276   */
00277 uint8_t BSP_SDRAM_ReadData(uint32_t uwStartA
ddress, uint32_t *pData, uint32_t uwDataSize)
00278 {
00279   if(HAL_SDRAM_Read_32b(&sdramHandle, (uint3
2_t *)uwStartAddress, pData, uwDataSize) != HAL_OK
)
00280   {
00281     return SDRAM_ERROR;
00282   }
00283   else
00284   {
00285     return SDRAM_OK;
```

```
00286   }
00287 }
00288
00289 /**
00290   * @brief  Reads an mount of data from the
SDRAM memory in DMA mode.
00291   * @param  uwStartAddress: Read start addre
ss
00292   * @param  pData: Pointer to data to be rea
d
00293   * @param  uwDataSize: Size of read data fr
om the memory
00294   * @retval SDRAM status
00295   */
00296 uint8_t BSP_SDRAM_ReadData_DMA(uint32_t uwSt
artAddress, uint32_t *pData, uint32_t uwDataSize)
00297 {
00298   if(HAL_SDRAM_Read_DMA(&sdramHandle, (uint3
2_t *)uwStartAddress, pData, uwDataSize) != HAL_OK
)
00299   {
00300     return SDRAM_ERROR;
00301   }
00302   else
00303   {
00304     return SDRAM_OK;
00305   }
00306 }
00307
00308 /**
00309   * @brief  Writes an mount of data to the S
DRAM memory in polling mode.
00310   * @param  uwStartAddress: Write start addr
ess
00311   * @param  pData: Pointer to data to be wri
tten
00312   * @param  uwDataSize: Size of written data
```

```
 from the memory
00313   * @retval SDRAM status
00314   */
00315 uint8_t BSP_SDRAM_WriteData(uint32_t uwStart
Address, uint32_t *pData, uint32_t uwDataSize)
00316 {
00317   if(HAL_SDRAM_Write_32b(&sdramHandle, (uint
32_t *)uwStartAddress, pData, uwDataSize) != HAL_O
K)
00318   {
00319     return SDRAM_ERROR;
00320   }
00321   else
00322   {
00323     return SDRAM_OK;
00324   }
00325 }
00326
00327 /**
00328   * @brief  Writes an mount of data to the S
DRAM memory in DMA mode.
00329   * @param  uwStartAddress: Write start addr
ess
00330   * @param  pData: Pointer to data to be wri
tten
00331   * @param  uwDataSize: Size of written data
 from the memory
00332   * @retval SDRAM status
00333   */
00334 uint8_t BSP_SDRAM_WriteData_DMA(uint32_t uwS
tartAddress, uint32_t *pData, uint32_t uwDataSize)

00335 {
00336   if(HAL_SDRAM_Write_DMA(&sdramHandle, (uint
32_t *)uwStartAddress, pData, uwDataSize) != HAL_O
K)
00337   {
```

```
00338     return SDRAM_ERROR;
00339   }
00340   else
00341   {
00342     return SDRAM_OK;
00343   }
00344 }
00345
00346 /**
00347   * @brief  Sends command to the SDRAM bank.
00348   * @param  SdramCmd: Pointer to SDRAM comma
nd structure
00349   * @retval HAL status
00350   */
00351 uint8_t BSP_SDRAM_Sendcmd(FMC_SDRAM_CommandT
ypeDef *SdramCmd)
00352 {
00353   if(HAL_SDRAM_SendCommand(&sdramHandle, Sdr
amCmd, SDRAM_TIMEOUT) != HAL_OK)
00354   {
00355     return SDRAM_ERROR;
00356   }
00357   else
00358   {
00359     return SDRAM_OK;
00360   }
00361 }
00362
00363 /**
00364   * @brief  Handles SDRAM DMA transfer inter
rupt request.
00365   */
00366 void BSP_SDRAM_DMA_IRQHandler(void)
00367 {
00368   HAL_DMA_IRQHandler(sdramHandle.hdma);
00369 }
00370
```

```
00371 /**
00372   * @brief  Initializes SDRAM MSP.
00373   * @param  hsdram: SDRAM handle
00374   * @param  Params: pointer on additional co
nfiguration parameters, can be NULL.
00375   */
00376 __weak void BSP_SDRAM_MspInit(SDRAM_HandleTy
peDef  *hsdram, void *Params)
00377 {
00378   static DMA_HandleTypeDef dma_handle;
00379   GPIO_InitTypeDef gpio_init_structure;
00380
00381   /* Enable FMC clock */
00382   __HAL_RCC_FMC_CLK_ENABLE();
00383
00384   /* Enable chosen DMAx clock */
00385   __DMAx_CLK_ENABLE();
00386
00387   /* Enable GPIOs clock */
00388   __HAL_RCC_GPIOA_CLK_ENABLE();
00389   __HAL_RCC_GPIOC_CLK_ENABLE();
00390   __HAL_RCC_GPIOD_CLK_ENABLE();
00391   __HAL_RCC_GPIOE_CLK_ENABLE();
00392   __HAL_RCC_GPIOF_CLK_ENABLE();
00393   __HAL_RCC_GPIOG_CLK_ENABLE();
00394
00395   /* Common GPIO configuration */
00396   gpio_init_structure.Mode      = GPIO_MODE_
AF_PP;
00397   gpio_init_structure.Pull      = GPIO_PULLU
P;
00398   gpio_init_structure.Speed     = GPIO_SPEED
_FAST;
00399   gpio_init_structure.Alternate = GPIO_AF12_
FMC;
00400
00401   /* GPIOA configuration */
```

```
00402   gpio_init_structure.Pin   = GPIO_PIN_7;
00403   HAL_GPIO_Init(GPIOA, &gpio_init_structure)
;
00404
00405   /* GPIOC configuration */
00406   gpio_init_structure.Pin   = GPIO_PIN_5 | G
PIO_PIN_4;
00407   HAL_GPIO_Init(GPIOC, &gpio_init_structure)
;
00408
00409   /* GPIOD configuration */
00410   gpio_init_structure.Pin   = GPIO_PIN_0 | G
PIO_PIN_1 | GPIO_PIN_8| GPIO_PIN_9 | GPIO_PIN_10 |
\
00411                               GPIO_PIN_14 |
GPIO_PIN_15;
00412
00413
00414   HAL_GPIO_Init(GPIOD, &gpio_init_structure)
;
00415
00416   /* GPIOE configuration */
00417   gpio_init_structure.Pin   = GPIO_PIN_0 | G
PIO_PIN_1 | GPIO_PIN_7| GPIO_PIN_8 | GPIO_PIN_9 |\
00418                               GPIO_PIN_10 |
GPIO_PIN_11 | GPIO_PIN_12 | GPIO_PIN_13 | GPIO_PIN
_14 |\
00419                               GPIO_PIN_15;
00420
00421   HAL_GPIO_Init(GPIOE, &gpio_init_structure)
;
00422
00423   /* GPIOF configuration */
00424   gpio_init_structure.Pin   = GPIO_PIN_0 | G
PIO_PIN_1 | GPIO_PIN_2| GPIO_PIN_3 | GPIO_PIN_4 |\
00425                               GPIO_PIN_5 | G
PIO_PIN_11 | GPIO_PIN_12 | GPIO_PIN_13 | GPIO_PIN_
```

```
14 |\
00426                                   GPIO_PIN_15;
00427
00428   HAL_GPIO_Init(GPIOF, &gpio_init_structure)
;
00429
00430   /* GPIOG configuration */
00431   gpio_init_structure.Pin   = GPIO_PIN_0 | G
PIO_PIN_1 | GPIO_PIN_4| GPIO_PIN_5 | GPIO_PIN_8 |\
00432                               GPIO_PIN_15 |
GPIO_PIN_2;
00433   HAL_GPIO_Init(GPIOG, &gpio_init_structure)
;
00434
00435   /* Configure common DMA parameters */
00436   dma_handle.Init.Channel             = SDRA
M_DMAx_CHANNEL;
00437   dma_handle.Init.Direction           = DMA_
MEMORY_TO_MEMORY;
00438   dma_handle.Init.PeriphInc           = DMA_
PINC_ENABLE;
00439   dma_handle.Init.MemInc              = DMA_
MINC_ENABLE;
00440   dma_handle.Init.PeriphDataAlignment = DMA_
PDATAALIGN_WORD;
00441   dma_handle.Init.MemDataAlignment    = DMA_
MDATAALIGN_WORD;
00442   dma_handle.Init.Mode                = DMA_
NORMAL;
00443   dma_handle.Init.Priority            = DMA_
PRIORITY_HIGH;
00444   dma_handle.Init.FIFOMode            = DMA_
FIFOMODE_DISABLE;
00445   dma_handle.Init.FIFOThreshold       = DMA_
FIFO_THRESHOLD_FULL;
00446   dma_handle.Init.MemBurst            = DMA_
MBURST_SINGLE;
```

```
00447   dma_handle.Init.PeriphBurst         = DMA_
PBURST_SINGLE;
00448
00449   dma_handle.Instance = SDRAM_DMAx_STREAM;
00450
00451    /* Associate the DMA handle */
00452   __HAL_LINKDMA(hsdram, hdma, dma_handle);
00453
00454   /* Deinitialize the stream for new transfe
r */
00455   HAL_DMA_DeInit(&dma_handle);
00456
00457   /* Configure the DMA stream */
00458   HAL_DMA_Init(&dma_handle);
00459
00460   /* NVIC configuration for DMA transfer com
plete interrupt */
00461   HAL_NVIC_SetPriority(SDRAM_DMAx_IRQn, 5, 0
);
00462   HAL_NVIC_EnableIRQ(SDRAM_DMAx_IRQn);
00463 }
00464
00465
00466 /**
00467   * @brief  DeInitializes SDRAM MSP.
00468   * @param  hsdram: SDRAM handle
00469   * @param  Params: pointer on additional co
nfiguration parameters, can be NULL.
00470   */
00471 __weak void BSP_SDRAM_MspDeInit(SDRAM_Handle
TypeDef  *hsdram, void *Params)
00472 {
00473     static DMA_HandleTypeDef dma_handle;
00474
00475     /* Disable NVIC configuration for DMA in
terrupt */
00476     HAL_NVIC_DisableIRQ(SDRAM_DMAx_IRQn);
```

```
00477
00478     /* Deinitialize the stream for new trans
fer */
00479     dma_handle.Instance = SDRAM_DMAx_STREAM;
00480     HAL_DMA_DeInit(&dma_handle);
00481
00482     /* DeInit GPIO pins can be done in the a
pplication
00483        (by surcharging this __weak function)
 */
00484
00485     /* GPIO pins clock, FMC clock and DMA cl
ock can be shut down in the application
00486        by surcharging this __weak function */

00487 }
00488
00489 /**
00490   * @}
00491   */
00492
00493 /**
00494   * @}
00495   */
00496
00497 /**
00498   * @}
00499   */
00500
00501 /**
00502   * @}
00503   */
00504
00505 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

Generated on Fri Jan 15 2016 10:06:21 for STM32446E_EVAL BSP
User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

Modules

BSP

Modules

STM32446E EVAL

Generated on Fri Jan 15 2016 10:06:21 for STM32446E_EVAL BSP
User Manual by doxygen 1.7.6.1

STM32446E_EVAL BSP User Manual

STM32446E EVAL

BSP

Modules

STM32446E EVAL LOW LEVEL

STM32446E EVAL AUDIO

This file includes the low layer driver for wm8994 Audio Codec available on STM32446E-EVAL evaluation board(MB1045).

STM32446E-EVAL CAMERA

STM32446E EVAL EEPROM

This file includes the I2C EEPROM driver of STM32446E-EVAL evaluation board.

STM32446E EVAL IO

STM32446E EVAL LCD

STM32446E EVAL QSPI

STM32446E EVAL SD

STM32446E EVAL SDRAM

STM32446E EVAL TS

STM32446E_EVAL BSP User Manual

STM32446E EVAL LOW LEVEL Exported Constants

STM32446E EVAL LOW LEVEL

Modules

STM32446E EVAL LOW LEVEL LED

Define for STM32446E_EVAL board.

STM32446E EVAL LOW LEVEL BUTTON

STM32446E EVAL LOW LEVEL COM

STM32446E_EVAL BSP User Manual

STM32446E EVAL QSPI

STM32446E EVAL

Modules

- STM32446E EVAL QSPI Private Variables
- STM32446E EVAL QSPI Private Functions
- STM32446E EVAL QSPI Exported Functions
- STM32446E EVAL QSPI Exported Constants
- STM32446E EVAL QSPI Exported Types