STM324x9I_EVAL BSP User Manual

STM324x9I EVAL LOW LEVEL Private TypesDefinitions

STM324x9I EVAL LOW LEVEL

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL LOW_LEVEL_Private_Macros

STM324x9I EVAL LOW LEVEL

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL LOW LEVEL Exported Macros

STM324x9I EVAL LOW LEVEL

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL AUDIO Private Types

STM324x9I EVAL AUDIO

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL AUDIO Private Defines

STM324x9I EVAL AUDIO

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL AUDIO Private Macros

STM324x9I EVAL AUDIO

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL AUDIO Exported Types

STM324x9I EVAL AUDIO

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL CAMERA Private TypesDefinitions

STM324x9I EVAL CAMERA

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL CAMERA Private Defines

STM324x9I EVAL CAMERA

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL CAMERA Private Macros

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL CAMERA Exported Constants

STM324x9I EVAL CAMERA

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL EEPROM Private Types

STM324x9I EVAL EEPROM

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL EEPROM Private Defines

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL EEPROM Private Macros

STM324x9I EVAL EEPROM

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL EEPROM Private Function Prototypes

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL EEPROM Exported Types

STM324x9I EVAL EEPROM

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL EEPROM Exported Macros

STM324x9I EVAL EEPROM

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL IO Private Types Definitions

STM324x9I EVAL IO

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL IO Private Defines

STM324x9I EVAL IO

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL IO Private Macros

STM324x9I EVAL IO

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL IO Private Function Prototypes

STM324x9I EVAL IO

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

IO_StateTypeDef Struct Reference

STM324x9I EVAL IO Exported Types

```
#include <stm324x9i_eval_io.h>
```

Data Fields

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

Detailed Description

Definition at line **67** of file **stm324x9i_eval_io.h**.

Field Documentation

uint16_t IO_StateTypeDef::TouchDetected

Definition at line 69 of file stm324x9i_eval_io.h.

uint16_t IO_StateTypeDef::x

Definition at line 70 of file stm324x9i_eval_io.h.

uint16_t IO_StateTypeDef::y

Definition at line 71 of file stm324x9i_eval_io.h.

uint16_t IO_StateTypeDef::z

Definition at line 72 of file stm324x9i_eval_io.h.

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

- stm324x9i_eval_io.h

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL IO Exported Macro

STM324x9I EVAL IO

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL LCD Private TypesDefinitions

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

LCD_DrawPropTypeDef Struct Reference

STM324x9I EVAL LCD Exported Types

```
#include <stm324x9i_eval_lcd.h>
```

Data Fields

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

Detailed Description

Definition at line **78** of file **stm324x9i_eval_lcd.h**.

Field Documentation

uint32_t LCD_DrawPropTypeDef::BackColor

Definition at line **81** of file **stm324x9i_eval_lcd.h**.

Referenced by **BSP_LCD_ClearStringLine()**, **BSP_LCD_GetBackColor()**, **BSP_LCD_LayerDefaultInit()**, and **BSP_LCD_SetBackColor()**.

sFONT* LCD_DrawPropTypeDef::pFont

Definition at line **82** of file **stm324x9i_eval_lcd.h**.

Referenced by **BSP_LCD_DisplayChar()**, **BSP_LCD_DisplayStringAt()**, **BSP_LCD_GetFont()**, **BSP_LCD_LayerDefaultInit()**, **BSP_LCD_SetFont()**, and **DrawChar()**.

uint32_t LCD_DrawPropTypeDef::TextColor

Definition at line **80** of file **stm324x9i_eval_lcd.h**.

Referenced by **BSP_LCD_ClearStringLine()**, **BSP_LCD_GetTextColor()**, **BSP_LCD_LayerDefaultInit()**, and **BSP_LCD_SetTextColor()**.

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

- **stm324x9i_eval_lcd.h**

STM324x9I_EVAL BSP User Manual

Point Struct Reference

STM324x9I EVAL LCD Exported Types

```
#include <stm324x9i_eval_lcd.h>
```

Data Fields

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

Detailed Description

Definition at line **85** of file **stm324x9i_eval_lcd.h**.

Field Documentation

int16_t Point::X

Definition at line **87** of file **stm324x9i_eval_lcd.h**.

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

int16_t Point::Y

Definition at line **88** of file **stm324x9i_eval_lcd.h**.

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

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

- **stm324x9i_eval_lcd.h**

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL NOR Private Types Definitions

STM324x9I EVAL NOR

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL NOR Private Defines

STM324x9I EVAL NOR

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL NOR Private Macros

STM324x9I EVAL NOR

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL NOR Private Function Prototypes

STM324x9I EVAL NOR

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL NOR Exported Types

STM324x9I EVAL NOR

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL NOR Exported Macro

STM324x9I EVAL NOR

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL SD Private TypesDefinitions

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL SD Private Defines

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL SD Private Macros

STM324x9I EVAL SD

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL SD Exported Macro

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL SDRAM Private Types Definitions

STM324x9I EVAL SDRAM

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL SDRAM Private Defines

STM324x9I EVAL SDRAM

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL SDRAM Private Macros

STM324x9I EVAL SDRAM

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL SDRAM Exported Macro

STM324x9I EVAL SDRAM

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL SRAM Private Types Definitions

STM324x9I EVAL SRAM

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL SRAM Private Defines

STM324x9I EVAL SRAM

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL SRAM Private Macros

STM324x9I EVAL SRAM

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL SRAM Exported Types

STM324x9I EVAL SRAM

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL SRAM Exported Macro

STM324x9I EVAL SRAM

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL TS Private Types Definitions

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL TS Private Defines

STM324x9I EVAL TS

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL TS Private Macros

STM324x9I EVAL TS

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL TS Private Function Prototypes

STM324x9I EVAL TS

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

TS_StateTypeDef Struct Reference

STM324x9I EVAL TS Exported Types

```
#include <stm324x9i_eval_ts.h>
```

Data Fields

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

Detailed Description

Definition at line **69** of file **stm324x9i_eval_ts.h**.

Field Documentation

uint16_t TS_StateTypeDef::TouchDetected

Definition at line 71 of file stm324x9i_eval_ts.h.

Referenced by BSP_TS_GetState().

uint16_t TS_StateTypeDef::x

Definition at line 72 of file stm324x9i_eval_ts.h.

Referenced by BSP_TS_GetState().

uint16_t TS_StateTypeDef::y

Definition at line 73 of file stm324x9i_eval_ts.h.

Referenced by BSP_TS_GetState().

uint16_t TS_StateTypeDef::z

Definition at line 74 of file stm324x9i_eval_ts.h.

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

- stm324x9i_eval_ts.h

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL TS Exported Macros

STM324x9I EVAL TS

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

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

- BackColor : **LCD_DrawPropTypeDef**
- pFont : **LCD_DrawPropTypeDef**
- TextColor : **LCD_DrawPropTypeDef**
- TouchDetected : **TS_StateTypeDef** , **IO_StateTypeDef**
- X : **Point**
- x : **TS_StateTypeDef** , **IO_StateTypeDef**
- Y : **Point**
- y : **TS_StateTypeDef** , **IO_StateTypeDef**
- z : **IO_StateTypeDef** , **TS_StateTypeDef**

STM324x9I_EVAL BSP User Manual

- BackColor : **LCD_DrawPropTypeDef**
- pFont : **LCD_DrawPropTypeDef**
- TextColor : **LCD_DrawPropTypeDef**
- TouchDetected : **TS_StateTypeDef** , **IO_StateTypeDef**
- X : **Point**
- x : **TS_StateTypeDef** , **IO_StateTypeDef**
- Y : **Point**
- y : **TS_StateTypeDef** , **IO_StateTypeDef**
- z : **IO_StateTypeDef** , **TS_StateTypeDef**

STM324x9I_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_ENABLE : **stm324x9i_eval_sdram.h**
- __DMAx_TxRx_CLK_ENABLE : **stm324x9i_eval_sd.h**
- __SRAM_DMAx_CLK_ENABLE : **stm324x9i_eval_sram.h**
- __STM324x9I_EVAL_BSP_VERSION : **stm324x9i_eval.c**
- __STM324x9I_EVAL_BSP_VERSION_MAIN : **stm324x9i_eval.c**
- __STM324x9I_EVAL_BSP_VERSION_RC : **stm324x9i_eval.c**
- __STM324x9I_EVAL_BSP_VERSION_SUB1 : **stm324x9i_eval.c**
- __STM324x9I_EVAL_BSP_VERSION_SUB2 : **stm324x9i_eval.c**

STM324x9I_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 : **stm324x9i_eval_lcd.c**
- ActiveLayer : **stm324x9i_eval_lcd.c**
- audio_drv : **stm324x9i_eval_audio.c**
- AUDIO_ERROR : **stm324x9i_eval_audio.h**
- AUDIO_I2C_ADDRESS : **stm324x9i_eval.h**
- AUDIO_I2Sx : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_CLK_ENABLE : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_DMAx_CHANNEL : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_DMAx_CLK_ENABLE : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_DMAx_IRQ : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_DMAx_IRQHandler : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_DMAx_MEM_DATA_SIZE : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_DMAx_PERIPH_DATA_SIZE : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_DMAx_STREAM : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_SCK_AF : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_SCK_GPIO_CLK_ENABLE : **stm324x9i_eval_audio.h**

- AUDIO_I2Sx_SCK_GPIO_PORT : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_SCK_PIN : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_SD_AF : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_SD_GPIO_CLK_ENABLE : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_SD_GPIO_PORT : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_SD_PIN : **stm324x9i_eval_audio.h**
- AUDIO_IN_IRQ_PREPRIO : **stm324x9i_eval_audio.h**
- AUDIO_INT_PIN : **stm324x9i_eval.h**
- AUDIO_IO_DeInit() : **stm324x9i_eval.c**
- AUDIO_IO_Delay() : **stm324x9i_eval.c**
- AUDIO_IO_Init() : **stm324x9i_eval.c**
- AUDIO_IO_Read() : **stm324x9i_eval.c**
- AUDIO_IO_Write() : **stm324x9i_eval.c**
- AUDIO_OK : **stm324x9i_eval_audio.h**
- AUDIO_OUT_IRQ_PREPRIO : **stm324x9i_eval_audio.h**
- AUDIO_SAIx : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_CLK_ENABLE : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_DMAx_CHANNEL : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_DMAx_CLK_ENABLE : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_DMAx_IRQ : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_DMAx_IRQHandler : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_DMAx_MEM_DATA_SIZE : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_DMAx_PERIPH_DATA_SIZE : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_DMAx_STREAM : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_FS_PIN : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_MCK_PIN : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_MCLK_SCK_SD_FS_AF : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_MCLK_SCK_SD_FS_ENABLE : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_MCLK_SCK_SD_FS_GPIO_PORT : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_SCK_PIN : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_SD_PIN : **stm324x9i_eval_audio.h**
- AUDIO_TIMEOUT : **stm324x9i_eval_audio.h**

- AUDIO_TIMx : **stm324x9i_eval_audio.h**
- AUDIO_TIMx_AF : **stm324x9i_eval_audio.h**
- AUDIO_TIMx_CLK_DISABLE : **stm324x9i_eval_audio.h**
- AUDIO_TIMx_CLK_ENABLE : **stm324x9i_eval_audio.h**
- AUDIO_TIMx_GPIO : **stm324x9i_eval_audio.h**
- AUDIO_TIMx_GPIO_CLK_ENABLE : **stm324x9i_eval_audio.h**
- AUDIO_TIMx_IN_CHANNEL : **stm324x9i_eval_audio.h**
- AUDIO_TIMx_IN_GPIO_PIN : **stm324x9i_eval_audio.h**
- AUDIO_TIMx_OUT_CHANNEL : **stm324x9i_eval_audio.h**
- AUDIO_TIMx_OUT_GPIO_PIN : **stm324x9i_eval_audio.h**
- AUDIODATA_SIZE : **stm324x9i_eval_audio.h**
- AudioInVolume : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**

STM324x9I_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 -

- BLOCKERASE_TIMEOUT : **stm324x9i_eval_nor.h**
- BSP_AUDIO_IN_Error_Callback() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_IN_HalfTransfer_CallBack() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_IN_Init() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_IN_Pause() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_IN_PDMToPCM() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_IN_Record() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_IN_Resume() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_IN_SetVolume() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_IN_Stop() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**

- BSP_AUDIO_IN_TransferComplete_CallBack() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_ChangeBuffer() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_Error_CallBack() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_HalfTransfer_CallBack() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_Init() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_Pause() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_Play() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_Resume() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_SetAudioFrameSlot() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_SetFrequency() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_SetMute() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_SetOutputMode() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_SetVolume() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_Stop() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_TransferComplete_CallBack() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_CAMERA_BlackWhiteConfig() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_ColorEffectConfig() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_ContinuousStart() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_ContrastBrightnessConfig() :

stm324x9i_eval_camera.c , **stm324x9i_eval_camera.h**

- BSP_CAMERA_DMA_IRQHandler() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_ErrorCallback() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_FrameEventCallback() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_Init() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_IRQHandler() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_LineEventCallback() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_Resume() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_SnapshotStart() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_Stop() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_Suspend() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_VsyncEventCallback() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_COM_Init() : **stm324x9i_eval.c** , **stm324x9i_eval.h**
- BSP_EEPROM_Init() : **stm324x9i_eval_eeprom.c** , **stm324x9i_eval_eeprom.h**
- BSP_EEPROM_ReadBuffer() : **stm324x9i_eval_eeprom.c** , **stm324x9i_eval_eeprom.h**
- BSP_EEPROM_TIMEOUT_UserCallback() : **stm324x9i_eval_eeprom.c** , **stm324x9i_eval_eeprom.h**
- BSP_EEPROM_WaitEepromStandbyState() : **stm324x9i_eval_eeprom.c** , **stm324x9i_eval_eeprom.h**
- BSP_EEPROM_WriteBuffer() : **stm324x9i_eval_eeprom.c** , **stm324x9i_eval_eeprom.h**
- BSP_EEPROM_WritePage() : **stm324x9i_eval_eeprom.c** , **stm324x9i_eval_eeprom.h**
- BSP_GetVersion() : **stm324x9i_eval.c** , **stm324x9i_eval.h**

- BSP_IO_ConfigPin() : **stm324x9i_eval_io.c** , **stm324x9i_eval_io.h**
- BSP_IO_Init() : **stm324x9i_eval_io.c** , **stm324x9i_eval_io.h**
- BSP_IO_ITClear() : **stm324x9i_eval_io.c** , **stm324x9i_eval_io.h**
- BSP_IO_ITGetStatus() : **stm324x9i_eval_io.c** , **stm324x9i_eval_io.h**
- BSP_IO_ReadPin() : **stm324x9i_eval_io.c** , **stm324x9i_eval_io.h**
- BSP_IO_TogglePin() : **stm324x9i_eval_io.c** , **stm324x9i_eval_io.h**
- BSP_IO_WritePin() : **stm324x9i_eval_io.c** , **stm324x9i_eval_io.h**
- BSP_JOY_GetState() : **stm324x9i_eval.c** , **stm324x9i_eval.h**
- BSP_JOY_Init() : **stm324x9i_eval.c** , **stm324x9i_eval.h**
- BSP_LCD_Clear() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_ClearStringLine() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_ClockConfig() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DisplayChar() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DisplayOff() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DisplayOn() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DisplayStringAt() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DisplayStringAtLine() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DrawBitmap() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DrawCircle() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DrawEllipse() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DrawHLine() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DrawLine() : **stm324x9i_eval_lcd.c** ,

stm324x9i_eval_lcd.h

- BSP_LCD_DrawPixel() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DrawPolygon() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DrawRect() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DrawVLine() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_FillCircle() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_FillEllipse() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_FillPolygon() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_FillRect() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_GetBackColor() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_GetFont() : **stm324x9i_eval_lcd.h** , **stm324x9i_eval_lcd.c**
- BSP_LCD_GetTextColor() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_GetXSize() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_GetYSize() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_Init() : **stm324x9i_eval_lcd.h** , **stm324x9i_eval_lcd.c**
- BSP_LCD_InitEx() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_LayerDefaultInit() : **stm324x9i_eval_lcd.h** , **stm324x9i_eval_lcd.c**
- BSP_LCD_ReadPixel() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_ResetColorKeying() : **stm324x9i_eval_lcd.h** , **stm324x9i_eval_lcd.c**
- BSP_LCD_SelectLayer() : **stm324x9i_eval_lcd.c** ,

stm324x9i_eval_lcd.h

- BSP_LCD_SetBackColor() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_SetColorKeying() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_SetFont() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_SetLayerAddress() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_SetLayerVisible() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_SetLayerWindow() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_SetTextColor() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_SetTransparency() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LED_Init() : **stm324x9i_eval.c** , **stm324x9i_eval.h**
- BSP_LED_Off() : **stm324x9i_eval.c** , **stm324x9i_eval.h**
- BSP_LED_On() : **stm324x9i_eval.c** , **stm324x9i_eval.h**
- BSP_LED_Toggle() : **stm324x9i_eval.c** , **stm324x9i_eval.h**
- BSP_NOR_Erase_Block() : **stm324x9i_eval_nor.c** , **stm324x9i_eval_nor.h**
- BSP_NOR_Erase_Chip() : **stm324x9i_eval_nor.c** , **stm324x9i_eval_nor.h**
- BSP_NOR_Init() : **stm324x9i_eval_nor.h** , **stm324x9i_eval_nor.c**
- BSP_NOR_ProgramData() : **stm324x9i_eval_nor.c** , **stm324x9i_eval_nor.h**
- BSP_NOR_Read_ID() : **stm324x9i_eval_nor.h** , **stm324x9i_eval_nor.c**
- BSP_NOR_ReadData() : **stm324x9i_eval_nor.h** , **stm324x9i_eval_nor.c**
- BSP_NOR_ReturnToReadMode() : **stm324x9i_eval_nor.h** , **stm324x9i_eval_nor.c**
- BSP_NOR_WriteData() : **stm324x9i_eval_nor.c** , **stm324x9i_eval_nor.h**
- BSP_PB_GetState() : **stm324x9i_eval.h** , **stm324x9i_eval.c**

- BSP_PB_Init() : **stm324x9i_eval.c** , **stm324x9i_eval.h**
- BSP_SD_DetectCallback() : **stm324x9i_eval_sd.h** , **stm324x9i_eval_sd.c**
- BSP_SD_DetectIT() : **stm324x9i_eval_sd.c** , **stm324x9i_eval_sd.h**
- BSP_SD_DMA_Rx_IRQHandler() : **stm324x9i_eval_sd.h** , **stm324x9i_eval_sd.c**
- BSP_SD_DMA_Tx_IRQHandler() : **stm324x9i_eval_sd.h** , **stm324x9i_eval_sd.c**
- BSP_SD_Erase() : **stm324x9i_eval_sd.h** , **stm324x9i_eval_sd.c**
- BSP_SD_GetCardInfo() : **stm324x9i_eval_sd.h** , **stm324x9i_eval_sd.c**
- BSP_SD_GetStatus() : **stm324x9i_eval_sd.h** , **stm324x9i_eval_sd.c**
- BSP_SD_Init() : **stm324x9i_eval_sd.h** , **stm324x9i_eval_sd.c**
- BSP_SD_IRQHandler() : **stm324x9i_eval_sd.h** , **stm324x9i_eval_sd.c**
- BSP_SD_IsDetected() : **stm324x9i_eval_sd.c** , **stm324x9i_eval_sd.h**
- BSP_SD_ITConfig() : **stm324x9i_eval_sd.h** , **stm324x9i_eval_sd.c**
- BSP_SD_ReadBlocks() : **stm324x9i_eval_sd.h** , **stm324x9i_eval_sd.c**
- BSP_SD_ReadBlocks_DMA() : **stm324x9i_eval_sd.c** , **stm324x9i_eval_sd.h**
- BSP_SD_WriteBlocks() : **stm324x9i_eval_sd.c** , **stm324x9i_eval_sd.h**
- BSP_SD_WriteBlocks_DMA() : **stm324x9i_eval_sd.h** , **stm324x9i_eval_sd.c**
- BSP_SDRAM_DMA_IRQHandler() : **stm324x9i_eval_sdram.h** , **stm324x9i_eval_sdram.c**
- BSP_SDRAM_Init() : **stm324x9i_eval_sdram.c** , **stm324x9i_eval_sdram.h**
- BSP_SDRAM_Initialization_sequence() : **stm324x9i_eval_sdram.c** , **stm324x9i_eval_sdram.h**
- BSP_SDRAM_ReadData() : **stm324x9i_eval_sdram.c** , **stm324x9i_eval_sdram.h**

- BSP_SDRAM_ReadData_DMA() : **stm324x9i_eval_sdram.c** , **stm324x9i_eval_sdram.h**
- BSP_SDRAM_Sendcmd() : **stm324x9i_eval_sdram.c** , **stm324x9i_eval_sdram.h**
- BSP_SDRAM_WriteData() : **stm324x9i_eval_sdram.c** , **stm324x9i_eval_sdram.h**
- BSP_SDRAM_WriteData_DMA() : **stm324x9i_eval_sdram.h** , **stm324x9i_eval_sdram.c**
- BSP_SRAM_DMA_IRQHandler() : **stm324x9i_eval_sram.h** , **stm324x9i_eval_sram.c**
- BSP_SRAM_Init() : **stm324x9i_eval_sram.c** , **stm324x9i_eval_sram.h**
- BSP_SRAM_ReadData() : **stm324x9i_eval_sram.h** , **stm324x9i_eval_sram.c**
- BSP_SRAM_ReadData_DMA() : **stm324x9i_eval_sram.c** , **stm324x9i_eval_sram.h**
- BSP_SRAM_WriteData() : **stm324x9i_eval_sram.c** , **stm324x9i_eval_sram.h**
- BSP_SRAM_WriteData_DMA() : **stm324x9i_eval_sram.c** , **stm324x9i_eval_sram.h**
- BSP_TS3510_IsDetected() : **stm324x9i_eval.h** , **stm324x9i_eval.c**
- BSP_TS_DeInit() : **stm324x9i_eval_ts.c** , **stm324x9i_eval_ts.h**
- BSP_TS_GetState() : **stm324x9i_eval_ts.c** , **stm324x9i_eval_ts.h**
- BSP_TS_Init() : **stm324x9i_eval_ts.c** , **stm324x9i_eval_ts.h**
- BSP_TS_ITClear() : **stm324x9i_eval_ts.c** , **stm324x9i_eval_ts.h**
- BSP_TS_ITConfig() : **stm324x9i_eval_ts.h** , **stm324x9i_eval_ts.c**
- BSP_TS_ITGetStatus() : **stm324x9i_eval_ts.c** , **stm324x9i_eval_ts.h**
- BUTTON_IRQn : **stm324x9i_eval.c**
- BUTTON_KEY : **stm324x9i_eval.h**
- BUTTON_MODE_EXTI : **stm324x9i_eval.h**
- BUTTON_MODE_GPIO : **stm324x9i_eval.h**
- BUTTON_PIN : **stm324x9i_eval.c**
- BUTTON_PORT : **stm324x9i_eval.c**

- BUTTON_TAMPER : **stm324x9i_eval.h**
- Button_TypeDef : **stm324x9i_eval.h**
- BUTTON_WAKEUP : **stm324x9i_eval.h**
- ButtonMode_TypeDef : **stm324x9i_eval.h**
- BUTTONn : **stm324x9i_eval.h**
- BUTTONx_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- BUTTONx_GPIO_CLK_ENABLE : **stm324x9i_eval.h**

STM324x9I_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 -

- CAM_PLUG_PIN : **stm324x9i_eval.h**
- CAMERA_Delay() : **stm324x9i_eval.c**
- camera_drv : **stm324x9i_eval_camera.c**
- CAMERA_ERROR : **stm324x9i_eval_camera.h**
- CAMERA_I2C_ADDRESS : **stm324x9i_eval.h**
- CAMERA_IO_Init() : **stm324x9i_eval.c**
- CAMERA_IO_Read() : **stm324x9i_eval.c**
- CAMERA_IO_Write() : **stm324x9i_eval.c**
- CAMERA_OK : **stm324x9i_eval_camera.h**
- Camera_StatusTypeDef : **stm324x9i_eval_camera.h**
- CAMERA_TIMEOUT : **stm324x9i_eval_camera.h**
- CENTER_MODE : **stm324x9i_eval_lcd.h**
- Channel_Demux : **stm324x9i_eval_audio.c**
- CHANNEL_DEMUX_MASK : **stm324x9i_eval_audio.h**
- CHIPERASE_TIMEOUT : **stm324x9i_eval_nor.h**
- CODEC_AUDIOFRAME_SLOT_0123 : **stm324x9i_eval_audio.h**
- CODEC_AUDIOFRAME_SLOT_02 : **stm324x9i_eval_audio.h**
- CODEC_AUDIOFRAME_SLOT_13 : **stm324x9i_eval_audio.h**
- CODEC_RESET_DELAY : **stm324x9i_eval_audio.h**

- COM1 : **stm324x9i_eval.h**
- COM2 : **stm324x9i_eval.h**
- COM_RX_AF : **stm324x9i_eval.c**
- COM_RX_PIN : **stm324x9i_eval.c**
- COM_RX_PORT : **stm324x9i_eval.c**
- COM_TX_AF : **stm324x9i_eval.c**
- COM_TX_PIN : **stm324x9i_eval.c**
- COM_TX_PORT : **stm324x9i_eval.c**
- COM_TypeDef : **stm324x9i_eval.h**
- COM_USART : **stm324x9i_eval.c**
- Command : **stm324x9i_eval_sdram.c**
- COMn : **stm324x9i_eval.h**
- CONTINUOUSCLOCK_FEATURE : **stm324x9i_eval_sram.h** , **stm324x9i_eval_nor.h**
- current_resolution : **stm324x9i_eval_camera.c**

STM324x9I_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 -

- DCMI_MspInit() : **stm324x9i_eval_camera.c**
- DEFAULT_AUDIO_IN_BIT_RESOLUTION : **stm324x9i_eval_audio.h**
- DEFAULT_AUDIO_IN_CHANNEL_NBR : **stm324x9i_eval_audio.h**
- DEFAULT_AUDIO_IN_FREQ : **stm324x9i_eval_audio.h**
- DEFAULT_AUDIO_IN_VOLUME : **stm324x9i_eval_audio.h**
- DMA_MAX : **stm324x9i_eval_audio.h**
- DMA_MAX_SZE : **stm324x9i_eval_audio.h**
- DrawChar() : **stm324x9i_eval_lcd.c**
- DrawProp : **stm324x9i_eval_lcd.c**

STM324x9I_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 : **stm324x9i_eval_eeprom.h**
- EEPROM_I2C_ADDRESS_A01 : **stm324x9i_eval.h**
- EEPROM_I2C_ADDRESS_A02 : **stm324x9i_eval.h**
- EEPROM_IO_Init() : **stm324x9i_eval.c** , **stm324x9i_eval_eeprom.h**
- EEPROM_IO_IsDeviceReady() : **stm324x9i_eval.c** , **stm324x9i_eval_eeprom.h**
- EEPROM_IO_ReadData() : **stm324x9i_eval.c** , **stm324x9i_eval_eeprom.h**
- EEPROM_IO_WriteData() : **stm324x9i_eval.c** , **stm324x9i_eval_eeprom.h**
- EEPROM_MAX_SIZE : **stm324x9i_eval_eeprom.h**
- EEPROM_MAX_TRIALS : **stm324x9i_eval_eeprom.h**
- EEPROM_OK : **stm324x9i_eval_eeprom.h**
- EEPROM_PAGESIZE : **stm324x9i_eval_eeprom.h**
- EEPROM_READ_TIMEOUT : **stm324x9i_eval_eeprom.h**
- EEPROM_TIMEOUT : **stm324x9i_eval_eeprom.h**
- EEPROM_WRITE_TIMEOUT : **stm324x9i_eval_eeprom.h**
- EEPROMAddress : **stm324x9i_eval_eeprom.c**

- EEPROMDataRead : **stm324x9i_eval_eeprom.c**
- EEPROMDataWrite : **stm324x9i_eval_eeprom.c**
- EEPROMTimeout : **stm324x9i_eval_eeprom.c**
- EVAL_COM1 : **stm324x9i_eval.h**
- EVAL_COM1_CLK_DISABLE : **stm324x9i_eval.h**
- EVAL_COM1_CLK_ENABLE : **stm324x9i_eval.h**
- EVAL_COM1_IRQn : **stm324x9i_eval.h**
- EVAL_COM1_RX_AF : **stm324x9i_eval.h**
- EVAL_COM1_RX_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- EVAL_COM1_RX_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- EVAL_COM1_RX_GPIO_PORT : **stm324x9i_eval.h**
- EVAL_COM1_RX_PIN : **stm324x9i_eval.h**
- EVAL_COM1_TX_AF : **stm324x9i_eval.h**
- EVAL_COM1_TX_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- EVAL_COM1_TX_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- EVAL_COM1_TX_GPIO_PORT : **stm324x9i_eval.h**
- EVAL_COM1_TX_PIN : **stm324x9i_eval.h**
- EVAL_COMx_CLK_DISABLE : **stm324x9i_eval.h**
- EVAL_COMx_CLK_ENABLE : **stm324x9i_eval.h**
- EVAL_COMx_RX_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- EVAL_COMx_RX_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- EVAL_COMx_TX_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- EVAL_COMx_TX_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- EVAL_DMAx_CLK_ENABLE : **stm324x9i_eval.h**
- EVAL_I2Cx : **stm324x9i_eval.h**
- EVAL_I2Cx_CLK_ENABLE : **stm324x9i_eval.h**
- EVAL_I2Cx_ER_IRQn : **stm324x9i_eval.h**
- EVAL_I2Cx_EV_IRQn : **stm324x9i_eval.h**
- EVAL_I2Cx_FORCE_RESET : **stm324x9i_eval.h**
- EVAL_I2Cx_RELEASE_RESET : **stm324x9i_eval.h**
- EVAL_I2Cx_SCL_PIN : **stm324x9i_eval.h**
- EVAL_I2Cx_SCL_SDA_AF : **stm324x9i_eval.h**
- EVAL_I2Cx_SCL_SDA_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- EVAL_I2Cx_SCL_SDA_GPIO_PORT : **stm324x9i_eval.h**
- EVAL_I2Cx_SDA_PIN : **stm324x9i_eval.h**
- EXC7200_I2C_ADDRESS : **stm324x9i_eval.h**

Generated on Wed Jan 13 2016 15:52:55 for STM324x9I_EVAL BSP
User Manual by  1.7.6.1

STM324x9I_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() : **stm324x9i_eval_lcd.c**
- Filter : **stm324x9i_eval_audio.c**

STM324x9I_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() : **stm324x9i_eval_camera.c**
- GPIO_PIN : **stm324x9i_eval.c**
- GPIO_PORT : **stm324x9i_eval.c**

STM324x9I_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() : **stm324x9i_eval_camera.c**
- HAL_DCMI_FrameEventCallback() : **stm324x9i_eval_camera.c**
- HAL_DCMI_LineEventCallback() : **stm324x9i_eval_camera.c**
- HAL_DCMI_VsyncEventCallback() : **stm324x9i_eval_camera.c**
- HAL_I2S_ErrorCallback() : **stm324x9i_eval_audio.c**
- HAL_I2S_RxCpltCallback() : **stm324x9i_eval_audio.c**
- HAL_I2S_RxHalfCpltCallback() : **stm324x9i_eval_audio.c**
- HAL_NOR_MspWait() : **stm324x9i_eval_nor.c**
- HAL_SAI_ErrorCallback() : **stm324x9i_eval_audio.c**
- HAL_SAI_TxCpltCallback() : **stm324x9i_eval_audio.c**
- HAL_SAI_TxHalfCpltCallback() : **stm324x9i_eval_audio.c**
- haudio_in_i2s : **stm324x9i_eval_audio.c**
- haudio_out_sai : **stm324x9i_eval_audio.c**
- haudio_tim : **stm324x9i_eval_audio.c**
- hdcmi_eval : **stm324x9i_eval_camera.c**
- hdma2d_eval : **stm324x9i_eval_lcd.c**
- heval_I2c : **stm324x9i_eval.c**
- hltdc_eval : **stm324x9i_eval_lcd.c**

Generated on Wed Jan 13 2016 15:52:55 for STM324x9I_EVAL BSP
User Manual by  1.7.6.1

STM324x9I_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 -

- I2C_Address : **stm324x9i_eval_ts.c**
- I2Cx_Error() : **stm324x9i_eval.c**
- I2Cx_Init() : **stm324x9i_eval.c**
- I2Cx_IsDeviceReady() : **stm324x9i_eval.c**
- I2Cx_ITConfig() : **stm324x9i_eval.c**
- I2Cx_MspInit() : **stm324x9i_eval.c**
- I2Cx_Read() : **stm324x9i_eval.c**
- I2Cx_ReadMultiple() : **stm324x9i_eval.c**
- I2Cx_Write() : **stm324x9i_eval.c**
- I2Cx_WriteMultiple() : **stm324x9i_eval.c**
- I2Sx_Init() : **stm324x9i_eval_audio.c**
- I2Sx_MspInit() : **stm324x9i_eval_audio.c**
- INTERNAL_BUFF_SIZE : **stm324x9i_eval_audio.h**
- io_driver : **stm324x9i_eval_io.c**
- IO_ERROR : **stm324x9i_eval_io.h**
- IO_I2C_ADDRESS : **stm324x9i_eval.h**
- IO_OK : **stm324x9i_eval_io.h**
- IO_PIN_0 : **stm324x9i_eval_io.h**
- IO_PIN_1 : **stm324x9i_eval_io.h**

- IO_PIN_10 : **stm324x9i_eval_io.h**
- IO_PIN_11 : **stm324x9i_eval_io.h**
- IO_PIN_12 : **stm324x9i_eval_io.h**
- IO_PIN_13 : **stm324x9i_eval_io.h**
- IO_PIN_14 : **stm324x9i_eval_io.h**
- IO_PIN_15 : **stm324x9i_eval_io.h**
- IO_PIN_2 : **stm324x9i_eval_io.h**
- IO_PIN_3 : **stm324x9i_eval_io.h**
- IO_PIN_4 : **stm324x9i_eval_io.h**
- IO_PIN_5 : **stm324x9i_eval_io.h**
- IO_PIN_6 : **stm324x9i_eval_io.h**
- IO_PIN_7 : **stm324x9i_eval_io.h**
- IO_PIN_8 : **stm324x9i_eval_io.h**
- IO_PIN_9 : **stm324x9i_eval_io.h**
- IO_PIN_ALL : **stm324x9i_eval_io.h**
- IO_StatusTypeDef : **stm324x9i_eval_io.h**
- IO_TIMEOUT : **stm324x9i_eval_io.h**
- IOE_Delay() : **stm324x9i_eval.c**
- IOE_Init() : **stm324x9i_eval.c**
- IOE_ITConfig() : **stm324x9i_eval.c**
- IOE_Read() : **stm324x9i_eval.c**
- IOE_ReadMultiple() : **stm324x9i_eval.c**
- IOE_Write() : **stm324x9i_eval.c**
- IOE_WriteMultiple() : **stm324x9i_eval.c**

STM324x9I_EVAL BSP User Manual

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

- j -

- JOY_ALL_PINS : **stm324x9i_eval.h**
- JOY_DOWN : **stm324x9i_eval.h**
- JOY_DOWN_PIN : **stm324x9i_eval.h**
- JOY_LEFT : **stm324x9i_eval.h**
- JOY_LEFT_PIN : **stm324x9i_eval.h**
- JOY_MODE_EXTI : **stm324x9i_eval.h**
- JOY_MODE_GPIO : **stm324x9i_eval.h**
- JOY_NONE : **stm324x9i_eval.h**
- JOY_NONE_PIN : **stm324x9i_eval.h**
- JOY_RIGHT : **stm324x9i_eval.h**
- JOY_RIGHT_PIN : **stm324x9i_eval.h**
- JOY_SEL : **stm324x9i_eval.h**
- JOY_SEL_PIN : **stm324x9i_eval.h**
- JOY_UP : **stm324x9i_eval.h**
- JOY_UP_PIN : **stm324x9i_eval.h**
- JOYMode_TypeDef : **stm324x9i_eval.h**
- JOYState_TypeDef : **stm324x9i_eval.h**

Generated on Wed Jan 13 2016 15:52:55 for STM324x9I_EVAL BSP
User Manual by  1.7.6.1

STM324x9I_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 : **stm324x9i_eval.h**
- KEY_BUTTON_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- KEY_BUTTON_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- KEY_BUTTON_GPIO_PORT : **stm324x9i_eval.h**
- KEY_BUTTON_PIN : **stm324x9i_eval.h**

STM324x9I_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 : **stm324x9i_eval_lcd.h**
- LCD_COLOR_BLUE : **stm324x9i_eval_lcd.h**
- LCD_COLOR_BROWN : **stm324x9i_eval_lcd.h**
- LCD_COLOR_CYAN : **stm324x9i_eval_lcd.h**
- LCD_COLOR_DARKBLUE : **stm324x9i_eval_lcd.h**
- LCD_COLOR_DARKCYAN : **stm324x9i_eval_lcd.h**
- LCD_COLOR_DARKGRAY : **stm324x9i_eval_lcd.h**
- LCD_COLOR_DARKGREEN : **stm324x9i_eval_lcd.h**
- LCD_COLOR_DARKMAGENTA : **stm324x9i_eval_lcd.h**
- LCD_COLOR_DARKRED : **stm324x9i_eval_lcd.h**
- LCD_COLOR_DARKYELLOW : **stm324x9i_eval_lcd.h**
- LCD_COLOR_GRAY : **stm324x9i_eval_lcd.h**
- LCD_COLOR_GREEN : **stm324x9i_eval_lcd.h**
- LCD_COLOR_LIGHTBLUE : **stm324x9i_eval_lcd.h**
- LCD_COLOR_LIGHTCYAN : **stm324x9i_eval_lcd.h**
- LCD_COLOR_LIGHTGRAY : **stm324x9i_eval_lcd.h**
- LCD_COLOR_LIGHTGREEN : **stm324x9i_eval_lcd.h**
- LCD_COLOR_LIGHTMAGENTA : **stm324x9i_eval_lcd.h**
- LCD_COLOR_LIGHTRED : **stm324x9i_eval_lcd.h**

- LCD_COLOR_LIGHTYELLOW : **stm324x9i_eval_lcd.h**
- LCD_COLOR_MAGENTA : **stm324x9i_eval_lcd.h**
- LCD_COLOR_ORANGE : **stm324x9i_eval_lcd.h**
- LCD_COLOR_RED : **stm324x9i_eval_lcd.h**
- LCD_COLOR_TRANSPARENT : **stm324x9i_eval_lcd.h**
- LCD_COLOR_WHITE : **stm324x9i_eval_lcd.h**
- LCD_COLOR_YELLOW : **stm324x9i_eval_lcd.h**
- LCD_DEFAULT_FONT : **stm324x9i_eval_lcd.h**
- LCD_ERROR : **stm324x9i_eval_lcd.h**
- LCD_FB_START_ADDRESS : **stm324x9i_eval_lcd.h**
- LCD_INT_PIN : **stm324x9i_eval.h**
- LCD_LayerCfgTypeDef : **stm324x9i_eval_lcd.h**
- LCD_MAX_PCLK : **stm324x9i_eval_lcd.h**
- LCD_MIN_PCLK : **stm324x9i_eval_lcd.h**
- LCD_OK : **stm324x9i_eval_lcd.h**
- LCD_TIMEOUT : **stm324x9i_eval_lcd.h**
- LED1 : **stm324x9i_eval.h**
- LED1_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- LED1_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- LED1_GPIO_PORT : **stm324x9i_eval.h**
- LED1_PIN : **stm324x9i_eval.h**
- LED2 : **stm324x9i_eval.h**
- LED2_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- LED2_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- LED2_GPIO_PORT : **stm324x9i_eval.h**
- LED2_PIN : **stm324x9i_eval.h**
- LED3 : **stm324x9i_eval.h**
- LED3_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- LED3_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- LED3_GPIO_PORT : **stm324x9i_eval.h**
- LED3_PIN : **stm324x9i_eval.h**
- LED4 : **stm324x9i_eval.h**
- LED4_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- LED4_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- LED4_GPIO_PORT : **stm324x9i_eval.h**
- LED4_PIN : **stm324x9i_eval.h**
- Led_TypeDef : **stm324x9i_eval.h**

- LEDn : **stm324x9i_eval.h**
- LEDx_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- LEDx_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- LEFT_MODE : **stm324x9i_eval_lcd.h**
- LL_ConvertLineToARGB8888() : **stm324x9i_eval_lcd.c**
- LL_FillBuffer() : **stm324x9i_eval_lcd.c**

STM324x9I_EVAL BSP User Manual

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

- m -

- MAX_LAYER_NUMBER : **stm324x9i_eval_lcd.h**
- MII_INT_PIN : **stm324x9i_eval.h**
- MSD_ERROR : **stm324x9i_eval_sd.h**
- MSD_OK : **stm324x9i_eval_sd.h**
- MspInit() : **stm324x9i_eval_lcd.c**

STM324x9I_EVAL BSP User Manual

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

- n -

- NOR_BURSTACCESS : **stm324x9i_eval_nor.h**
- NOR_BUSY_STATE : **stm324x9i_eval_nor.h**
- NOR_DEVICE_ADDR : **stm324x9i_eval_nor.h**
- NOR_MEMORY_WIDTH : **stm324x9i_eval_nor.h**
- NOR_MspInit() : **stm324x9i_eval_nor.c**
- NOR_READY_BUSY_GPIO : **stm324x9i_eval_nor.h**
- NOR_READY_BUSY_PIN : **stm324x9i_eval_nor.h**
- NOR_READY_STATE : **stm324x9i_eval_nor.h**
- NOR_STATUS_ERROR : **stm324x9i_eval_nor.h**
- NOR_STATUS_OK : **stm324x9i_eval_nor.h**
- NOR_WRITEBURST : **stm324x9i_eval_nor.h**
- norHandle : **stm324x9i_eval_nor.c**

STM324x9I_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 -

- OTG_FS1_OVER_CURRENT_PIN : **stm324x9i_eval.h**
- OTG_FS1_POWER_SWITCH_PIN : **stm324x9i_eval.h**
- OTG_FS2_OVER_CURRENT_PIN : **stm324x9i_eval.h**
- OTG_FS2_POWER_SWITCH_PIN : **stm324x9i_eval.h**

STM324x9I_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 -

- PCLK_profile : **stm324x9i_eval_lcd.c**
- PCM_OUT_SIZE : **stm324x9i_eval_audio.h**
- PDMDecoder_Init() : **stm324x9i_eval_audio.c**
- POLY_X : **stm324x9i_eval_lcd.c**
- POLY_Y : **stm324x9i_eval_lcd.c**
- pPoint : **stm324x9i_eval_lcd.h**
- PROGRAM_TIMEOUT : **stm324x9i_eval_nor.h**

STM324x9I_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 : **stm324x9i_eval_sdram.h**
- RESOLUTION_R160x120 : **stm324x9i_eval_camera.h**
- RESOLUTION_R320x240 : **stm324x9i_eval_camera.h**
- RESOLUTION_R480x272 : **stm324x9i_eval_camera.h**
- RESOLUTION_R640x480 : **stm324x9i_eval_camera.h**
- RIGHT_MODE : **stm324x9i_eval_lcd.h**
- RSTI_PIN : **stm324x9i_eval.h**

STM324x9I_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_Init() : **stm324x9i_eval_audio.c**
- SAIx_MspInit() : **stm324x9i_eval_audio.c**
- SD_CardInfo : **stm324x9i_eval_sd.h**
- SD_DATATIMEOUT : **stm324x9i_eval_sd.h**
- SD_DETECT_PIN : **stm324x9i_eval.h**
- SD_DetectIRQHandler : **stm324x9i_eval_sd.h**
- SD_DMAx_Rx_CHANNEL : **stm324x9i_eval_sd.h**
- SD_DMAx_Rx_IRQHandler : **stm324x9i_eval_sd.h**
- SD_DMAx_Rx_IRQn : **stm324x9i_eval_sd.h**
- SD_DMAx_Rx_STREAM : **stm324x9i_eval_sd.h**
- SD_DMAx_Tx_CHANNEL : **stm324x9i_eval_sd.h**
- SD_DMAx_Tx_IRQHandler : **stm324x9i_eval_sd.h**
- SD_DMAx_Tx_IRQn : **stm324x9i_eval_sd.h**
- SD_DMAx_Tx_STREAM : **stm324x9i_eval_sd.h**
- SD_MspInit() : **stm324x9i_eval_sd.c**
- SD_NOT_PRESENT : **stm324x9i_eval_sd.h**
- SD_PRESENT : **stm324x9i_eval_sd.h**
- SDCLOCK_PERIOD : **stm324x9i_eval_sdram.h**
- SDRAM_DEVICE_ADDR : **stm324x9i_eval_sdram.h**

- SDRAM_DEVICE_SIZE : **stm324x9i_eval_sdram.h**
- SDRAM_DMAx_CHANNEL : **stm324x9i_eval_sdram.h**
- SDRAM_DMAx_IRQHandler : **stm324x9i_eval_sdram.h**
- SDRAM_DMAx_IRQn : **stm324x9i_eval_sdram.h**
- SDRAM_DMAx_STREAM : **stm324x9i_eval_sdram.h**
- SDRAM_ERROR : **stm324x9i_eval_sdram.h**
- SDRAM_MEMORY_WIDTH : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_BURST_LENGTH_1 : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_BURST_LENGTH_2 : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_BURST_LENGTH_4 : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_BURST_LENGTH_8 : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_BURST_TYPE_INTERLEAVED : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_CAS_LATENCY_2 : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_CAS_LATENCY_3 : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_OPERATING_MODE_STANDARD : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_WRITEBURST_MODE_PROGRAMMED : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_WRITEBURST_MODE_SINGLE : **stm324x9i_eval_sdram.h**
- SDRAM_MspInit() : **stm324x9i_eval_sdram.c**
- SDRAM_OK : **stm324x9i_eval_sdram.h**
- SDRAM_TIMEOUT : **stm324x9i_eval_sdram.h**
- sdramHandle : **stm324x9i_eval_sdram.c**
- SRAM_BURSTACCESS : **stm324x9i_eval_sram.h**
- SRAM_DEVICE_ADDR : **stm324x9i_eval_sram.h**
- SRAM_DEVICE_SIZE : **stm324x9i_eval_sram.h**
- SRAM_DMAx_CHANNEL : **stm324x9i_eval_sram.h**

- SRAM_DMAx_IRQHandler : **stm324x9i_eval_sram.h**
- SRAM_DMAx_IRQn : **stm324x9i_eval_sram.h**
- SRAM_DMAx_STREAM : **stm324x9i_eval_sram.h**
- SRAM_ERROR : **stm324x9i_eval_sram.h**
- SRAM_MEMORY_WIDTH : **stm324x9i_eval_sram.h**
- SRAM_MspInit() : **stm324x9i_eval_sram.c**
- SRAM_OK : **stm324x9i_eval_sram.h**
- SRAM_WRITEBURST : **stm324x9i_eval_sram.h**
- sramHandle : **stm324x9i_eval_sram.c**

STM324x9I_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 : **stm324x9i_eval.h**
- TAMPER_BUTTON_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- TAMPER_BUTTON_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- TAMPER_BUTTON_GPIO_PORT : **stm324x9i_eval.h**
- TAMPER_BUTTON_PIN : **stm324x9i_eval.h**
- Text_AlignModeTypdef : **stm324x9i_eval_lcd.h**
- Timing : **stm324x9i_eval_sdram.c** , **stm324x9i_eval_sram.c** , **stm324x9i_eval_nor.c**
- TIMx_IC_MspInit() : **stm324x9i_eval_audio.c**
- TIMx_Init() : **stm324x9i_eval_audio.c**
- TS3510_I2C_ADDRESS : **stm324x9i_eval.h**
- ts_driver : **stm324x9i_eval_ts.c**
- TS_ERROR : **stm324x9i_eval_ts.h**
- TS_I2C_ADDRESS : **stm324x9i_eval.h**
- TS_INT_PIN : **stm324x9i_eval_ts.h**
- TS_OK : **stm324x9i_eval_ts.h**
- ts_orientation : **stm324x9i_eval_ts.c**
- TS_StatusTypeDef : **stm324x9i_eval_ts.h**
- TS_SWAP_NONE : **stm324x9i_eval_ts.h**

- TS_SWAP_X : **stm324x9i_eval_ts.h**
- TS_SWAP_XY : **stm324x9i_eval_ts.h**
- TS_SWAP_Y : **stm324x9i_eval_ts.h**
- TS_TIMEOUT : **stm324x9i_eval_ts.h**
- ts_x_boundary : **stm324x9i_eval_ts.c**
- ts_y_boundary : **stm324x9i_eval_ts.c**

STM324x9I_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 : **stm324x9i_eval_sd.c**
- uSdHandle : **stm324x9i_eval_sd.c**

STM324x9I_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 : **stm324x9i_eval.h**
- WAKEUP_BUTTON_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- WAKEUP_BUTTON_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- WAKEUP_BUTTON_GPIO_PORT : **stm324x9i_eval.h**
- WAKEUP_BUTTON_PIN : **stm324x9i_eval.h**

STM324x9I_EVAL BSP User Manual

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

- x -

- XSDN_PIN : **stm324x9i_eval.h**

STM324x9I_EVAL BSP User Manual

- a -

- AUDIO_IO_DeInit() : **stm324x9i_eval.c**
- AUDIO_IO_Delay() : **stm324x9i_eval.c**
- AUDIO_IO_Init() : **stm324x9i_eval.c**
- AUDIO_IO_Read() : **stm324x9i_eval.c**
- AUDIO_IO_Write() : **stm324x9i_eval.c**

STM324x9I_EVAL BSP User Manual

- b -

- BSP_AUDIO_IN_Error_Callback() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_IN_HalfTransfer_CallBack() : **stm324x9i_eval_audio.h** , **stm324x9i_eval_audio.c**
- BSP_AUDIO_IN_Init() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_IN_Pause() : **stm324x9i_eval_audio.h** , **stm324x9i_eval_audio.c**
- BSP_AUDIO_IN_PDMToPCM() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_IN_Record() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_IN_Resume() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_IN_SetVolume() : **stm324x9i_eval_audio.h** , **stm324x9i_eval_audio.c**
- BSP_AUDIO_IN_Stop() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_IN_TransferComplete_CallBack() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**

- BSP_AUDIO_OUT_ChangeBuffer() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_Error_CallBack() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_HalfTransfer_CallBack() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_Init() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_Pause() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_Play() : **stm324x9i_eval_audio.h** , **stm324x9i_eval_audio.c**
- BSP_AUDIO_OUT_Resume() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_SetAudioFrameSlot() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_SetFrequency() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_SetMute() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_SetOutputMode() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_SetVolume() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_Stop() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_AUDIO_OUT_TransferComplete_CallBack() : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**
- BSP_CAMERA_BlackWhiteConfig() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_ColorEffectConfig() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_ContinuousStart() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_ContrastBrightnessConfig() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_DMA_IRQHandler() : **stm324x9i_eval_camera.c**

, **stm324x9i_eval_camera.h**

- BSP_CAMERA_ErrorCallback() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_FrameEventCallback() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_Init() : **stm324x9i_eval_camera.h** , **stm324x9i_eval_camera.c**
- BSP_CAMERA_IRQHandler() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_LineEventCallback() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_Resume() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_SnapshotStart() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_Stop() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_Suspend() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_CAMERA_VsyncEventCallback() : **stm324x9i_eval_camera.c** , **stm324x9i_eval_camera.h**
- BSP_COM_Init() : **stm324x9i_eval.c** , **stm324x9i_eval.h**
- BSP_EEPROM_Init() : **stm324x9i_eval_eeprom.c** , **stm324x9i_eval_eeprom.h**
- BSP_EEPROM_ReadBuffer() : **stm324x9i_eval_eeprom.c** , **stm324x9i_eval_eeprom.h**
- BSP_EEPROM_TIMEOUT_UserCallback() : **stm324x9i_eval_eeprom.c** , **stm324x9i_eval_eeprom.h**
- BSP_EEPROM_WaitEepromStandbyState() : **stm324x9i_eval_eeprom.c** , **stm324x9i_eval_eeprom.h**
- BSP_EEPROM_WriteBuffer() : **stm324x9i_eval_eeprom.c** , **stm324x9i_eval_eeprom.h**
- BSP_EEPROM_WritePage() : **stm324x9i_eval_eeprom.c** , **stm324x9i_eval_eeprom.h**
- BSP_GetVersion() : **stm324x9i_eval.c** , **stm324x9i_eval.h**
- BSP_IO_ConfigPin() : **stm324x9i_eval_io.c** , **stm324x9i_eval_io.h**

- BSP_IO_Init() : **stm324x9i_eval_io.c** , **stm324x9i_eval_io.h**
- BSP_IO_ITClear() : **stm324x9i_eval_io.c** , **stm324x9i_eval_io.h**
- BSP_IO_ITGetStatus() : **stm324x9i_eval_io.c** , **stm324x9i_eval_io.h**
- BSP_IO_ReadPin() : **stm324x9i_eval_io.c** , **stm324x9i_eval_io.h**
- BSP_IO_TogglePin() : **stm324x9i_eval_io.c** , **stm324x9i_eval_io.h**
- BSP_IO_WritePin() : **stm324x9i_eval_io.c** , **stm324x9i_eval_io.h**
- BSP_JOY_GetState() : **stm324x9i_eval.c** , **stm324x9i_eval.h**
- BSP_JOY_Init() : **stm324x9i_eval.c** , **stm324x9i_eval.h**
- BSP_LCD_Clear() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_ClearStringLine() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_ClockConfig() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DisplayChar() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DisplayOff() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DisplayOn() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DisplayStringAt() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DisplayStringAtLine() : **stm324x9i_eval_lcd.h** , **stm324x9i_eval_lcd.c**
- BSP_LCD_DrawBitmap() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DrawCircle() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DrawEllipse() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DrawHLine() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DrawLine() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DrawPixel() : **stm324x9i_eval_lcd.c** ,

stm324x9i_eval_lcd.h

- BSP_LCD_DrawPolygon() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DrawRect() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_DrawVLine() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_FillCircle() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_FillEllipse() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_FillPolygon() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_FillRect() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_GetBackColor() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_GetFont() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_GetTextColor() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_GetXSize() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_GetYSize() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_Init() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_InitEx() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_LayerDefaultInit() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_ReadPixel() : **stm324x9i_eval_lcd.h** , **stm324x9i_eval_lcd.c**
- BSP_LCD_ResetColorKeying() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_SelectLayer() : **stm324x9i_eval_lcd.h** , **stm324x9i_eval_lcd.c**
- BSP_LCD_SetBackColor() : **stm324x9i_eval_lcd.c** ,

stm324x9i_eval_lcd.h

- BSP_LCD_SetColorKeying() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_SetFont() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_SetLayerAddress() : **stm324x9i_eval_lcd.h** , **stm324x9i_eval_lcd.c**
- BSP_LCD_SetLayerVisible() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_SetLayerWindow() : **stm324x9i_eval_lcd.h** , **stm324x9i_eval_lcd.c**
- BSP_LCD_SetTextColor() : **stm324x9i_eval_lcd.c** , **stm324x9i_eval_lcd.h**
- BSP_LCD_SetTransparency() : **stm324x9i_eval_lcd.h** , **stm324x9i_eval_lcd.c**
- BSP_LED_Init() : **stm324x9i_eval.c** , **stm324x9i_eval.h**
- BSP_LED_Off() : **stm324x9i_eval.h** , **stm324x9i_eval.c**
- BSP_LED_On() : **stm324x9i_eval.c** , **stm324x9i_eval.h**
- BSP_LED_Toggle() : **stm324x9i_eval.c** , **stm324x9i_eval.h**
- BSP_NOR_Erase_Block() : **stm324x9i_eval_nor.h** , **stm324x9i_eval_nor.c**
- BSP_NOR_Erase_Chip() : **stm324x9i_eval_nor.h** , **stm324x9i_eval_nor.c**
- BSP_NOR_Init() : **stm324x9i_eval_nor.h** , **stm324x9i_eval_nor.c**
- BSP_NOR_ProgramData() : **stm324x9i_eval_nor.h** , **stm324x9i_eval_nor.c**
- BSP_NOR_Read_ID() : **stm324x9i_eval_nor.h** , **stm324x9i_eval_nor.c**
- BSP_NOR_ReadData() : **stm324x9i_eval_nor.h** , **stm324x9i_eval_nor.c**
- BSP_NOR_ReturnToReadMode() : **stm324x9i_eval_nor.h** , **stm324x9i_eval_nor.c**
- BSP_NOR_WriteData() : **stm324x9i_eval_nor.c** , **stm324x9i_eval_nor.h**
- BSP_PB_GetState() : **stm324x9i_eval.c** , **stm324x9i_eval.h**
- BSP_PB_Init() : **stm324x9i_eval.c** , **stm324x9i_eval.h**
- BSP_SD_DetectCallback() : **stm324x9i_eval_sd.h** ,

stm324x9i_eval_sd.c

- BSP_SD_DetectIT() : **stm324x9i_eval_sd.c** , **stm324x9i_eval_sd.h**
- BSP_SD_DMA_Rx_IRQHandler() : **stm324x9i_eval_sd.h** , **stm324x9i_eval_sd.c**
- BSP_SD_DMA_Tx_IRQHandler() : **stm324x9i_eval_sd.c** , **stm324x9i_eval_sd.h**
- BSP_SD_Erase() : **stm324x9i_eval_sd.c** , **stm324x9i_eval_sd.h**
- BSP_SD_GetCardInfo() : **stm324x9i_eval_sd.c** , **stm324x9i_eval_sd.h**
- BSP_SD_GetStatus() : **stm324x9i_eval_sd.h** , **stm324x9i_eval_sd.c**
- BSP_SD_Init() : **stm324x9i_eval_sd.h** , **stm324x9i_eval_sd.c**
- BSP_SD_IRQHandler() : **stm324x9i_eval_sd.c** , **stm324x9i_eval_sd.h**
- BSP_SD_IsDetected() : **stm324x9i_eval_sd.c** , **stm324x9i_eval_sd.h**
- BSP_SD_ITConfig() : **stm324x9i_eval_sd.h** , **stm324x9i_eval_sd.c**
- BSP_SD_ReadBlocks() : **stm324x9i_eval_sd.c** , **stm324x9i_eval_sd.h**
- BSP_SD_ReadBlocks_DMA() : **stm324x9i_eval_sd.c** , **stm324x9i_eval_sd.h**
- BSP_SD_WriteBlocks() : **stm324x9i_eval_sd.c** , **stm324x9i_eval_sd.h**
- BSP_SD_WriteBlocks_DMA() : **stm324x9i_eval_sd.c** , **stm324x9i_eval_sd.h**
- BSP_SDRAM_DMA_IRQHandler() : **stm324x9i_eval_sdram.h** , **stm324x9i_eval_sdram.c**
- BSP_SDRAM_Init() : **stm324x9i_eval_sdram.h** , **stm324x9i_eval_sdram.c**
- BSP_SDRAM_Initialization_sequence() : **stm324x9i_eval_sdram.h** , **stm324x9i_eval_sdram.c**
- BSP_SDRAM_ReadData() : **stm324x9i_eval_sdram.c** , **stm324x9i_eval_sdram.h**
- BSP_SDRAM_ReadData_DMA() : **stm324x9i_eval_sdram.h** , **stm324x9i_eval_sdram.c**

- BSP_SDRAM_Sendcmd() : **stm324x9i_eval_sdram.c** , **stm324x9i_eval_sdram.h**
- BSP_SDRAM_WriteData() : **stm324x9i_eval_sdram.h** , **stm324x9i_eval_sdram.c**
- BSP_SDRAM_WriteData_DMA() : **stm324x9i_eval_sdram.c** , **stm324x9i_eval_sdram.h**
- BSP_SRAM_DMA_IRQHandler() : **stm324x9i_eval_sram.h** , **stm324x9i_eval_sram.c**
- BSP_SRAM_Init() : **stm324x9i_eval_sram.h** , **stm324x9i_eval_sram.c**
- BSP_SRAM_ReadData() : **stm324x9i_eval_sram.h** , **stm324x9i_eval_sram.c**
- BSP_SRAM_ReadData_DMA() : **stm324x9i_eval_sram.c** , **stm324x9i_eval_sram.h**
- BSP_SRAM_WriteData() : **stm324x9i_eval_sram.c** , **stm324x9i_eval_sram.h**
- BSP_SRAM_WriteData_DMA() : **stm324x9i_eval_sram.c** , **stm324x9i_eval_sram.h**
- BSP_TS3510_IsDetected() : **stm324x9i_eval.c** , **stm324x9i_eval.h**
- BSP_TS_DeInit() : **stm324x9i_eval_ts.h** , **stm324x9i_eval_ts.c**
- BSP_TS_GetState() : **stm324x9i_eval_ts.h** , **stm324x9i_eval_ts.c**
- BSP_TS_Init() : **stm324x9i_eval_ts.h** , **stm324x9i_eval_ts.c**
- BSP_TS_ITClear() : **stm324x9i_eval_ts.c** , **stm324x9i_eval_ts.h**
- BSP_TS_ITConfig() : **stm324x9i_eval_ts.h** , **stm324x9i_eval_ts.c**
- BSP_TS_ITGetStatus() : **stm324x9i_eval_ts.c** , **stm324x9i_eval_ts.h**

STM324x9I_EVAL BSP User Manual

- c -

- CAMERA_Delay() : **stm324x9i_eval.c**
- CAMERA_IO_Init() : **stm324x9i_eval.c**
- CAMERA_IO_Read() : **stm324x9i_eval.c**
- CAMERA_IO_Write() : **stm324x9i_eval.c**

STM324x9I_EVAL BSP User Manual

- d -

- DCMI_MspInit() : **stm324x9i_eval_camera.c**
- DrawChar() : **stm324x9i_eval_lcd.c**

STM324x9I_EVAL BSP User Manual

- e -

- EEPROM_IO_Init() : **stm324x9i_eval.c** , **stm324x9i_eval_eeprom.h**
- EEPROM_IO_IsDeviceReady() : **stm324x9i_eval_eeprom.h** , **stm324x9i_eval.c**
- EEPROM_IO_ReadData() : **stm324x9i_eval.c** , **stm324x9i_eval_eeprom.h**
- EEPROM_IO_WriteData() : **stm324x9i_eval_eeprom.h** , **stm324x9i_eval.c**

STM324x9I_EVAL BSP User Manual

- f -

- FillTriangle() : **stm324x9i_eval_lcd.c**

STM324x9I_EVAL BSP User Manual

- g -

- GetSize() : **stm324x9i_eval_camera.c**

STM324x9I_EVAL BSP User Manual

- h -

- HAL_DCMI_ErrorCallback() : **stm324x9i_eval_camera.c**
- HAL_DCMI_FrameEventCallback() : **stm324x9i_eval_camera.c**
- HAL_DCMI_LineEventCallback() : **stm324x9i_eval_camera.c**
- HAL_DCMI_VsyncEventCallback() : **stm324x9i_eval_camera.c**
- HAL_I2S_ErrorCallback() : **stm324x9i_eval_audio.c**
- HAL_I2S_RxCpltCallback() : **stm324x9i_eval_audio.c**
- HAL_I2S_RxHalfCpltCallback() : **stm324x9i_eval_audio.c**
- HAL_NOR_MspWait() : **stm324x9i_eval_nor.c**
- HAL_SAI_ErrorCallback() : **stm324x9i_eval_audio.c**
- HAL_SAI_TxCpltCallback() : **stm324x9i_eval_audio.c**
- HAL_SAI_TxHalfCpltCallback() : **stm324x9i_eval_audio.c**

STM324x9I_EVAL BSP User Manual

- i -

Generated on Wed Jan 13 2016 15:52:55 for STM324x9I_EVAL BSP
User Manual by  1.7.6.1

STM324x9I_EVAL BSP User Manual

- l -

- LL_ConvertLineToARGB8888() : **stm324x9i_eval_lcd.c**
- LL_FillBuffer() : **stm324x9i_eval_lcd.c**

STM324x9I_EVAL BSP User Manual

- m -

- MspInit() : **stm324x9i_eval_lcd.c**

STM324x9I_EVAL BSP User Manual

- n -

- NOR_MspInit() : **stm324x9i_eval_nor.c**

STM324x9I_EVAL BSP User Manual

- p -

- PDMDecoder_Init() : **stm324x9i_eval_audio.c**

STM324x9I_EVAL BSP User Manual

- s -

- SAIx_Init() : **stm324x9i_eval_audio.c**
- SAIx_MspInit() : **stm324x9i_eval_audio.c**
- SD_MspInit() : **stm324x9i_eval_sd.c**
- SDRAM_MspInit() : **stm324x9i_eval_sdram.c**
- SRAM_MspInit() : **stm324x9i_eval_sram.c**

STM324x9I_EVAL BSP User Manual

- t -

- TIMx_IC_MspInit() : **stm324x9i_eval_audio.c**
- TIMx_Init() : **stm324x9i_eval_audio.c**

STM324x9I_EVAL BSP User Manual

- a -

- ActiveLayer : **stm324x9i_eval_lcd.c**
- audio_drv : **stm324x9i_eval_audio.c**
- AudioInVolume : **stm324x9i_eval_audio.c** , **stm324x9i_eval_audio.h**

- b -

- BUTTON_IRQn : **stm324x9i_eval.c**
- BUTTON_PIN : **stm324x9i_eval.c**
- BUTTON_PORT : **stm324x9i_eval.c**

- c -

- camera_drv : **stm324x9i_eval_camera.c**
- Channel_Demux : **stm324x9i_eval_audio.c**
- COM_RX_AF : **stm324x9i_eval.c**
- COM_RX_PIN : **stm324x9i_eval.c**
- COM_RX_PORT : **stm324x9i_eval.c**
- COM_TX_AF : **stm324x9i_eval.c**
- COM_TX_PIN : **stm324x9i_eval.c**

- COM_TX_PORT : **stm324x9i_eval.c**
- COM_USART : **stm324x9i_eval.c**
- Command : **stm324x9i_eval_sdram.c**
- current_resolution : **stm324x9i_eval_camera.c**

- d -

- DrawProp : **stm324x9i_eval_lcd.c**

- e -

- EEPROMAddress : **stm324x9i_eval_eeprom.c**
- EEPROMDataRead : **stm324x9i_eval_eeprom.c**
- EEPROMDataWrite : **stm324x9i_eval_eeprom.c**
- EEPROMTimeout : **stm324x9i_eval_eeprom.c**

- f -

- Filter : **stm324x9i_eval_audio.c**

- g -

- GPIO_PIN : **stm324x9i_eval.c**
- GPIO_PORT : **stm324x9i_eval.c**

- h -

- haudio_in_i2s : **stm324x9i_eval_audio.c**
- haudio_out_sai : **stm324x9i_eval_audio.c**
- haudio_tim : **stm324x9i_eval_audio.c**
- hdcmi_eval : **stm324x9i_eval_camera.c**
- hdma2d_eval : **stm324x9i_eval_lcd.c**
- heval_I2c : **stm324x9i_eval.c**
- hltdc_eval : **stm324x9i_eval_lcd.c**

- i -

- I2C_Address : **stm324x9i_eval_ts.c**
- io_driver : **stm324x9i_eval_io.c**

- n -

- norHandle : **stm324x9i_eval_nor.c**

- p -

- PCLK_profile : **stm324x9i_eval_lcd.c**

- s -

- sdramHandle : **stm324x9i_eval_sdram.c**
- sramHandle : **stm324x9i_eval_sram.c**

- t -

- Timing : **stm324x9i_eval_nor.c** , **stm324x9i_eval_sdram.c** , **stm324x9i_eval_sram.c**
- ts_driver : **stm324x9i_eval_ts.c**
- ts_orientation : **stm324x9i_eval_ts.c**
- ts_x_boundary : **stm324x9i_eval_ts.c**
- ts_y_boundary : **stm324x9i_eval_ts.c**

- u -

- uSdCardInfo : **stm324x9i_eval_sd.c**
- uSdHandle : **stm324x9i_eval_sd.c**

STM324x9I_EVAL BSP User Manual

- pPoint : **stm324x9i_eval_lcd.h**

STM324x9I_EVAL BSP User Manual

- Button_TypeDef : **stm324x9i_eval.h**
- ButtonMode_TypeDef : **stm324x9i_eval.h**
- Camera_StatusTypeDef : **stm324x9i_eval_camera.h**
- COM_TypeDef : **stm324x9i_eval.h**
- IO_StatusTypeDef : **stm324x9i_eval_io.h**
- JOYMode_TypeDef : **stm324x9i_eval.h**
- JOYState_TypeDef : **stm324x9i_eval.h**
- Led_TypeDef : **stm324x9i_eval.h**
- Text_AlignModeTypdef : **stm324x9i_eval_lcd.h**
- TS_StatusTypeDef : **stm324x9i_eval_ts.h**

STM324x9I_EVAL BSP User Manual

- b -

- BUTTON_KEY : **stm324x9i_eval.h**
- BUTTON_MODE_EXTI : **stm324x9i_eval.h**
- BUTTON_MODE_GPIO : **stm324x9i_eval.h**
- BUTTON_TAMPER : **stm324x9i_eval.h**
- BUTTON_WAKEUP : **stm324x9i_eval.h**

- c -

- CAMERA_ERROR : **stm324x9i_eval_camera.h**
- CAMERA_OK : **stm324x9i_eval_camera.h**
- CAMERA_TIMEOUT : **stm324x9i_eval_camera.h**
- CENTER_MODE : **stm324x9i_eval_lcd.h**
- COM1 : **stm324x9i_eval.h**
- COM2 : **stm324x9i_eval.h**

- i -

- IO_ERROR : **stm324x9i_eval_io.h**
- IO_OK : **stm324x9i_eval_io.h**
- IO_TIMEOUT : **stm324x9i_eval_io.h**

- j -

- JOY_DOWN : **stm324x9i_eval.h**
- JOY_LEFT : **stm324x9i_eval.h**
- JOY_MODE_EXTI : **stm324x9i_eval.h**
- JOY_MODE_GPIO : **stm324x9i_eval.h**
- JOY_NONE : **stm324x9i_eval.h**
- JOY_RIGHT : **stm324x9i_eval.h**
- JOY_SEL : **stm324x9i_eval.h**
- JOY_UP : **stm324x9i_eval.h**

- l -

- LED1 : **stm324x9i_eval.h**
- LED2 : **stm324x9i_eval.h**
- LED3 : **stm324x9i_eval.h**
- LED4 : **stm324x9i_eval.h**
- LEFT_MODE : **stm324x9i_eval_lcd.h**

- r -

- RIGHT_MODE : **stm324x9i_eval_lcd.h**

- t -

- TS_ERROR : **stm324x9i_eval_ts.h**
- TS_OK : **stm324x9i_eval_ts.h**
- TS_TIMEOUT : **stm324x9i_eval_ts.h**

STM324x9I_EVAL BSP User Manual

- _ -

- __DMAx_CLK_ENABLE : **stm324x9i_eval_sdram.h**
- __DMAx_TxRx_CLK_ENABLE : **stm324x9i_eval_sd.h**
- __SRAM_DMAx_CLK_ENABLE : **stm324x9i_eval_sram.h**
- __STM324x9I_EVAL_BSP_VERSION : **stm324x9i_eval.c**
- __STM324x9I_EVAL_BSP_VERSION_MAIN : **stm324x9i_eval.c**
- __STM324x9I_EVAL_BSP_VERSION_RC : **stm324x9i_eval.c**
- __STM324x9I_EVAL_BSP_VERSION_SUB1 : **stm324x9i_eval.c**
- __STM324x9I_EVAL_BSP_VERSION_SUB2 : **stm324x9i_eval.c**

STM324x9I_EVAL BSP User Manual

- a -

- ABS : **stm324x9i_eval_lcd.c**
- AUDIO_ERROR : **stm324x9i_eval_audio.h**
- AUDIO_I2C_ADDRESS : **stm324x9i_eval.h**
- AUDIO_I2Sx : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_CLK_ENABLE : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_DMAx_CHANNEL : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_DMAx_CLK_ENABLE : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_DMAx_IRQ : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_DMAx_IRQHandler : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_DMAx_MEM_DATA_SIZE : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_DMAx_PERIPH_DATA_SIZE : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_DMAx_STREAM : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_SCK_AF : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_SCK_GPIO_CLK_ENABLE : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_SCK_GPIO_PORT : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_SCK_PIN : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_SD_AF : **stm324x9i_eval_audio.h**

- AUDIO_I2Sx_SD_GPIO_CLK_ENABLE : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_SD_GPIO_PORT : **stm324x9i_eval_audio.h**
- AUDIO_I2Sx_SD_PIN : **stm324x9i_eval_audio.h**
- AUDIO_IN_IRQ_PREPRIO : **stm324x9i_eval_audio.h**
- AUDIO_INT_PIN : **stm324x9i_eval.h**
- AUDIO_OK : **stm324x9i_eval_audio.h**
- AUDIO_OUT_IRQ_PREPRIO : **stm324x9i_eval_audio.h**
- AUDIO_SAIx : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_CLK_ENABLE : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_DMAx_CHANNEL : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_DMAx_CLK_ENABLE : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_DMAx_IRQ : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_DMAx_IRQHandler : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_DMAx_MEM_DATA_SIZE : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_DMAx_PERIPH_DATA_SIZE : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_DMAx_STREAM : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_FS_PIN : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_MCK_PIN : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_MCLK_SCK_SD_FS_AF : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_MCLK_SCK_SD_FS_ENABLE : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_MCLK_SCK_SD_FS_GPIO_PORT : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_SCK_PIN : **stm324x9i_eval_audio.h**
- AUDIO_SAIx_SD_PIN : **stm324x9i_eval_audio.h**
- AUDIO_TIMEOUT : **stm324x9i_eval_audio.h**
- AUDIO_TIMx : **stm324x9i_eval_audio.h**
- AUDIO_TIMx_AF : **stm324x9i_eval_audio.h**
- AUDIO_TIMx_CLK_DISABLE : **stm324x9i_eval_audio.h**
- AUDIO_TIMx_CLK_ENABLE : **stm324x9i_eval_audio.h**
- AUDIO_TIMx_GPIO : **stm324x9i_eval_audio.h**
- AUDIO_TIMx_GPIO_CLK_ENABLE : **stm324x9i_eval_audio.h**
- AUDIO_TIMx_IN_CHANNEL : **stm324x9i_eval_audio.h**
- AUDIO_TIMx_IN_GPIO_PIN : **stm324x9i_eval_audio.h**

- AUDIO_TIMx_OUT_CHANNEL : **stm324x9i_eval_audio.h**
- AUDIO_TIMx_OUT_GPIO_PIN : **stm324x9i_eval_audio.h**
- AUDIODATA_SIZE : **stm324x9i_eval_audio.h**

STM324x9I_EVAL BSP User Manual

- b -

- BLOCKERASE_TIMEOUT : **stm324x9i_eval_nor.h**
- BUTTONn : **stm324x9i_eval.h**
- BUTTONx_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- BUTTONx_GPIO_CLK_ENABLE : **stm324x9i_eval.h**

STM324x9I_EVAL BSP User Manual

- c -

- CAM_PLUG_PIN : **stm324x9i_eval.h**
- CAMERA_I2C_ADDRESS : **stm324x9i_eval.h**
- CHANNEL_DEMUX_MASK : **stm324x9i_eval_audio.h**
- CHIPERASE_TIMEOUT : **stm324x9i_eval_nor.h**
- CODEC_AUDIOFRAME_SLOT_0123 : **stm324x9i_eval_audio.h**
- CODEC_AUDIOFRAME_SLOT_02 : **stm324x9i_eval_audio.h**
- CODEC_AUDIOFRAME_SLOT_13 : **stm324x9i_eval_audio.h**
- CODEC_RESET_DELAY : **stm324x9i_eval_audio.h**
- COMn : **stm324x9i_eval.h**
- CONTINUOUSCLOCK_FEATURE : **stm324x9i_eval_nor.h** , **stm324x9i_eval_sram.h**

STM324x9I_EVAL BSP User Manual

- d -

- DEFAULT_AUDIO_IN_BIT_RESOLUTION : **stm324x9i_eval_audio.h**
- DEFAULT_AUDIO_IN_CHANNEL_NBR : **stm324x9i_eval_audio.h**
- DEFAULT_AUDIO_IN_FREQ : **stm324x9i_eval_audio.h**
- DEFAULT_AUDIO_IN_VOLUME : **stm324x9i_eval_audio.h**
- DMA_MAX : **stm324x9i_eval_audio.h**
- DMA_MAX_SZE : **stm324x9i_eval_audio.h**

STM324x9I_EVAL BSP User Manual

- e -

- EEPROM_FAIL : **stm324x9i_eval_eeprom.h**
- EEPROM_I2C_ADDRESS_A01 : **stm324x9i_eval.h**
- EEPROM_I2C_ADDRESS_A02 : **stm324x9i_eval.h**
- EEPROM_MAX_SIZE : **stm324x9i_eval_eeprom.h**
- EEPROM_MAX_TRIALS : **stm324x9i_eval_eeprom.h**
- EEPROM_OK : **stm324x9i_eval_eeprom.h**
- EEPROM_PAGESIZE : **stm324x9i_eval_eeprom.h**
- EEPROM_READ_TIMEOUT : **stm324x9i_eval_eeprom.h**
- EEPROM_TIMEOUT : **stm324x9i_eval_eeprom.h**
- EEPROM_WRITE_TIMEOUT : **stm324x9i_eval_eeprom.h**
- EVAL_COM1 : **stm324x9i_eval.h**
- EVAL_COM1_CLK_DISABLE : **stm324x9i_eval.h**
- EVAL_COM1_CLK_ENABLE : **stm324x9i_eval.h**
- EVAL_COM1_IRQn : **stm324x9i_eval.h**
- EVAL_COM1_RX_AF : **stm324x9i_eval.h**
- EVAL_COM1_RX_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- EVAL_COM1_RX_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- EVAL_COM1_RX_GPIO_PORT : **stm324x9i_eval.h**
- EVAL_COM1_RX_PIN : **stm324x9i_eval.h**
- EVAL_COM1_TX_AF : **stm324x9i_eval.h**

- EVAL_COM1_TX_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- EVAL_COM1_TX_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- EVAL_COM1_TX_GPIO_PORT : **stm324x9i_eval.h**
- EVAL_COM1_TX_PIN : **stm324x9i_eval.h**
- EVAL_COMx_CLK_DISABLE : **stm324x9i_eval.h**
- EVAL_COMx_CLK_ENABLE : **stm324x9i_eval.h**
- EVAL_COMx_RX_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- EVAL_COMx_RX_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- EVAL_COMx_TX_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- EVAL_COMx_TX_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- EVAL_DMAx_CLK_ENABLE : **stm324x9i_eval.h**
- EVAL_I2Cx : **stm324x9i_eval.h**
- EVAL_I2Cx_CLK_ENABLE : **stm324x9i_eval.h**
- EVAL_I2Cx_ER_IRQn : **stm324x9i_eval.h**
- EVAL_I2Cx_EV_IRQn : **stm324x9i_eval.h**
- EVAL_I2Cx_FORCE_RESET : **stm324x9i_eval.h**
- EVAL_I2Cx_RELEASE_RESET : **stm324x9i_eval.h**
- EVAL_I2Cx_SCL_PIN : **stm324x9i_eval.h**
- EVAL_I2Cx_SCL_SDA_AF : **stm324x9i_eval.h**
- EVAL_I2Cx_SCL_SDA_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- EVAL_I2Cx_SCL_SDA_GPIO_PORT : **stm324x9i_eval.h**
- EVAL_I2Cx_SDA_PIN : **stm324x9i_eval.h**
- EXC7200_I2C_ADDRESS : **stm324x9i_eval.h**

STM324x9I_EVAL BSP User Manual

- i -

- INTERNAL_BUFF_SIZE : **stm324x9i_eval_audio.h**
- IO_I2C_ADDRESS : **stm324x9i_eval.h**
- IO_PIN_0 : **stm324x9i_eval_io.h**
- IO_PIN_1 : **stm324x9i_eval_io.h**
- IO_PIN_10 : **stm324x9i_eval_io.h**
- IO_PIN_11 : **stm324x9i_eval_io.h**
- IO_PIN_12 : **stm324x9i_eval_io.h**
- IO_PIN_13 : **stm324x9i_eval_io.h**
- IO_PIN_14 : **stm324x9i_eval_io.h**
- IO_PIN_15 : **stm324x9i_eval_io.h**
- IO_PIN_2 : **stm324x9i_eval_io.h**
- IO_PIN_3 : **stm324x9i_eval_io.h**
- IO_PIN_4 : **stm324x9i_eval_io.h**
- IO_PIN_5 : **stm324x9i_eval_io.h**
- IO_PIN_6 : **stm324x9i_eval_io.h**
- IO_PIN_7 : **stm324x9i_eval_io.h**
- IO_PIN_8 : **stm324x9i_eval_io.h**
- IO_PIN_9 : **stm324x9i_eval_io.h**
- IO_PIN_ALL : **stm324x9i_eval_io.h**

Generated on Wed Jan 13 2016 15:52:55 for STM324x9I_EVAL BSP
User Manual by  1.7.6.1

STM324x9I_EVAL BSP User Manual

- j -

- JOY_ALL_PINS : **stm324x9i_eval.h**
- JOY_DOWN_PIN : **stm324x9i_eval.h**
- JOY_LEFT_PIN : **stm324x9i_eval.h**
- JOY_NONE_PIN : **stm324x9i_eval.h**
- JOY_RIGHT_PIN : **stm324x9i_eval.h**
- JOY_SEL_PIN : **stm324x9i_eval.h**
- JOY_UP_PIN : **stm324x9i_eval.h**

STM324x9I_EVAL BSP User Manual

- k -

- KEY_BUTTON_EXTI_IRQn : **stm324x9i_eval.h**
- KEY_BUTTON_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- KEY_BUTTON_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- KEY_BUTTON_GPIO_PORT : **stm324x9i_eval.h**
- KEY_BUTTON_PIN : **stm324x9i_eval.h**

STM324x9I_EVAL BSP User Manual

- l -

- LCD_COLOR_BLACK : **stm324x9i_eval_lcd.h**
- LCD_COLOR_BLUE : **stm324x9i_eval_lcd.h**
- LCD_COLOR_BROWN : **stm324x9i_eval_lcd.h**
- LCD_COLOR_CYAN : **stm324x9i_eval_lcd.h**
- LCD_COLOR_DARKBLUE : **stm324x9i_eval_lcd.h**
- LCD_COLOR_DARKCYAN : **stm324x9i_eval_lcd.h**
- LCD_COLOR_DARKGRAY : **stm324x9i_eval_lcd.h**
- LCD_COLOR_DARKGREEN : **stm324x9i_eval_lcd.h**
- LCD_COLOR_DARKMAGENTA : **stm324x9i_eval_lcd.h**
- LCD_COLOR_DARKRED : **stm324x9i_eval_lcd.h**
- LCD_COLOR_DARKYELLOW : **stm324x9i_eval_lcd.h**
- LCD_COLOR_GRAY : **stm324x9i_eval_lcd.h**
- LCD_COLOR_GREEN : **stm324x9i_eval_lcd.h**
- LCD_COLOR_LIGHTBLUE : **stm324x9i_eval_lcd.h**
- LCD_COLOR_LIGHTCYAN : **stm324x9i_eval_lcd.h**
- LCD_COLOR_LIGHTGRAY : **stm324x9i_eval_lcd.h**
- LCD_COLOR_LIGHTGREEN : **stm324x9i_eval_lcd.h**
- LCD_COLOR_LIGHTMAGENTA : **stm324x9i_eval_lcd.h**
- LCD_COLOR_LIGHTRED : **stm324x9i_eval_lcd.h**
- LCD_COLOR_LIGHTYELLOW : **stm324x9i_eval_lcd.h**

- LCD_COLOR_MAGENTA : **stm324x9i_eval_lcd.h**
- LCD_COLOR_ORANGE : **stm324x9i_eval_lcd.h**
- LCD_COLOR_RED : **stm324x9i_eval_lcd.h**
- LCD_COLOR_TRANSPARENT : **stm324x9i_eval_lcd.h**
- LCD_COLOR_WHITE : **stm324x9i_eval_lcd.h**
- LCD_COLOR_YELLOW : **stm324x9i_eval_lcd.h**
- LCD_DEFAULT_FONT : **stm324x9i_eval_lcd.h**
- LCD_ERROR : **stm324x9i_eval_lcd.h**
- LCD_FB_START_ADDRESS : **stm324x9i_eval_lcd.h**
- LCD_INT_PIN : **stm324x9i_eval.h**
- LCD_LayerCfgTypeDef : **stm324x9i_eval_lcd.h**
- LCD_MAX_PCLK : **stm324x9i_eval_lcd.h**
- LCD_MIN_PCLK : **stm324x9i_eval_lcd.h**
- LCD_OK : **stm324x9i_eval_lcd.h**
- LCD_TIMEOUT : **stm324x9i_eval_lcd.h**
- LED1_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- LED1_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- LED1_GPIO_PORT : **stm324x9i_eval.h**
- LED1_PIN : **stm324x9i_eval.h**
- LED2_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- LED2_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- LED2_GPIO_PORT : **stm324x9i_eval.h**
- LED2_PIN : **stm324x9i_eval.h**
- LED3_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- LED3_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- LED3_GPIO_PORT : **stm324x9i_eval.h**
- LED3_PIN : **stm324x9i_eval.h**
- LED4_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- LED4_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- LED4_GPIO_PORT : **stm324x9i_eval.h**
- LED4_PIN : **stm324x9i_eval.h**
- LEDn : **stm324x9i_eval.h**
- LEDx_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- LEDx_GPIO_CLK_ENABLE : **stm324x9i_eval.h**

User Manual by  1.7.6.1

STM324x9I_EVAL BSP User Manual

- m -

- MAX_LAYER_NUMBER : **stm324x9i_eval_lcd.h**
- MII_INT_PIN : **stm324x9i_eval.h**
- MSD_ERROR : **stm324x9i_eval_sd.h**
- MSD_OK : **stm324x9i_eval_sd.h**

STM324x9I_EVAL BSP User Manual

- n -

- NOR_BURSTACCESS : **stm324x9i_eval_nor.h**
- NOR_BUSY_STATE : **stm324x9i_eval_nor.h**
- NOR_DEVICE_ADDR : **stm324x9i_eval_nor.h**
- NOR_MEMORY_WIDTH : **stm324x9i_eval_nor.h**
- NOR_READY_BUSY_GPIO : **stm324x9i_eval_nor.h**
- NOR_READY_BUSY_PIN : **stm324x9i_eval_nor.h**
- NOR_READY_STATE : **stm324x9i_eval_nor.h**
- NOR_STATUS_ERROR : **stm324x9i_eval_nor.h**
- NOR_STATUS_OK : **stm324x9i_eval_nor.h**
- NOR_WRITEBURST : **stm324x9i_eval_nor.h**

STM324x9I_EVAL BSP User Manual

- o -

- OTG_FS1_OVER_CURRENT_PIN : **stm324x9i_eval.h**
- OTG_FS1_POWER_SWITCH_PIN : **stm324x9i_eval.h**
- OTG_FS2_OVER_CURRENT_PIN : **stm324x9i_eval.h**
- OTG_FS2_POWER_SWITCH_PIN : **stm324x9i_eval.h**

STM324x9I_EVAL BSP User Manual

- p -

- PCM_OUT_SIZE : **stm324x9i_eval_audio.h**
- POLY_X : **stm324x9i_eval_lcd.c**
- POLY_Y : **stm324x9i_eval_lcd.c**
- PROGRAM_TIMEOUT : **stm324x9i_eval_nor.h**

STM324x9I_EVAL BSP User Manual

- r -

- REFRESH_COUNT : **stm324x9i_eval_sdram.h**
- RESOLUTION_R160x120 : **stm324x9i_eval_camera.h**
- RESOLUTION_R320x240 : **stm324x9i_eval_camera.h**
- RESOLUTION_R480x272 : **stm324x9i_eval_camera.h**
- RESOLUTION_R640x480 : **stm324x9i_eval_camera.h**
- RSTI_PIN : **stm324x9i_eval.h**

STM324x9I_EVAL BSP User Manual

- s -

- SD_CardInfo : **stm324x9i_eval_sd.h**
- SD_DATATIMEOUT : **stm324x9i_eval_sd.h**
- SD_DETECT_PIN : **stm324x9i_eval.h**
- SD_DetectIRQHandler : **stm324x9i_eval_sd.h**
- SD_DMAx_Rx_CHANNEL : **stm324x9i_eval_sd.h**
- SD_DMAx_Rx_IRQHandler : **stm324x9i_eval_sd.h**
- SD_DMAx_Rx_IRQn : **stm324x9i_eval_sd.h**
- SD_DMAx_Rx_STREAM : **stm324x9i_eval_sd.h**
- SD_DMAx_Tx_CHANNEL : **stm324x9i_eval_sd.h**
- SD_DMAx_Tx_IRQHandler : **stm324x9i_eval_sd.h**
- SD_DMAx_Tx_IRQn : **stm324x9i_eval_sd.h**
- SD_DMAx_Tx_STREAM : **stm324x9i_eval_sd.h**
- SD_NOT_PRESENT : **stm324x9i_eval_sd.h**
- SD_PRESENT : **stm324x9i_eval_sd.h**
- SDCLOCK_PERIOD : **stm324x9i_eval_sdram.h**
- SDRAM_DEVICE_ADDR : **stm324x9i_eval_sdram.h**
- SDRAM_DEVICE_SIZE : **stm324x9i_eval_sdram.h**
- SDRAM_DMAx_CHANNEL : **stm324x9i_eval_sdram.h**
- SDRAM_DMAx_IRQHandler : **stm324x9i_eval_sdram.h**
- SDRAM_DMAx_IRQn : **stm324x9i_eval_sdram.h**

- SDRAM_DMAx_STREAM : **stm324x9i_eval_sdram.h**
- SDRAM_ERROR : **stm324x9i_eval_sdram.h**
- SDRAM_MEMORY_WIDTH : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_BURST_LENGTH_1 : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_BURST_LENGTH_2 : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_BURST_LENGTH_4 : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_BURST_LENGTH_8 : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_BURST_TYPE_INTERLEAVED : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_CAS_LATENCY_2 : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_CAS_LATENCY_3 : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_OPERATING_MODE_STANDARD : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_WRITEBURST_MODE_PROGRAMMED : **stm324x9i_eval_sdram.h**
- SDRAM_MODEREG_WRITEBURST_MODE_SINGLE : **stm324x9i_eval_sdram.h**
- SDRAM_OK : **stm324x9i_eval_sdram.h**
- SDRAM_TIMEOUT : **stm324x9i_eval_sdram.h**
- SRAM_BURSTACCESS : **stm324x9i_eval_sram.h**
- SRAM_DEVICE_ADDR : **stm324x9i_eval_sram.h**
- SRAM_DEVICE_SIZE : **stm324x9i_eval_sram.h**
- SRAM_DMAx_CHANNEL : **stm324x9i_eval_sram.h**
- SRAM_DMAx_IRQHandler : **stm324x9i_eval_sram.h**
- SRAM_DMAx_IRQn : **stm324x9i_eval_sram.h**
- SRAM_DMAx_STREAM : **stm324x9i_eval_sram.h**
- SRAM_ERROR : **stm324x9i_eval_sram.h**
- SRAM_MEMORY_WIDTH : **stm324x9i_eval_sram.h**
- SRAM_OK : **stm324x9i_eval_sram.h**

- SRAM_WRITEBURST : **stm324x9i_eval_sram.h**

STM324x9I_EVAL BSP User Manual

- t -

- TAMPER_BUTTON_EXTI_IRQn : **stm324x9i_eval.h**
- TAMPER_BUTTON_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- TAMPER_BUTTON_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- TAMPER_BUTTON_GPIO_PORT : **stm324x9i_eval.h**
- TAMPER_BUTTON_PIN : **stm324x9i_eval.h**
- TS3510_I2C_ADDRESS : **stm324x9i_eval.h**
- TS_I2C_ADDRESS : **stm324x9i_eval.h**
- TS_INT_PIN : **stm324x9i_eval_ts.h**
- TS_SWAP_NONE : **stm324x9i_eval_ts.h**
- TS_SWAP_X : **stm324x9i_eval_ts.h**
- TS_SWAP_XY : **stm324x9i_eval_ts.h**
- TS_SWAP_Y : **stm324x9i_eval_ts.h**

STM324x9I_EVAL BSP User Manual

- w -

- WAKEUP_BUTTON_EXTI_IRQn : **stm324x9i_eval.h**
- WAKEUP_BUTTON_GPIO_CLK_DISABLE : **stm324x9i_eval.h**
- WAKEUP_BUTTON_GPIO_CLK_ENABLE : **stm324x9i_eval.h**
- WAKEUP_BUTTON_GPIO_PORT : **stm324x9i_eval.h**
- WAKEUP_BUTTON_PIN : **stm324x9i_eval.h**

STM324x9I_EVAL BSP User Manual

- x -

- XSDN_PIN : **stm324x9i_eval.h**

STM324x9I_EVAL BSP User Manual

stm324x9i_eval.c File Reference

This file provides a set of firmware functions to manage LEDs, push-buttons and COM ports available on STM324x9I-EVAL evaluation board(MB1045) RevB from STMicroelectronics. More...

```
#include "stm324x9i_eval.h" #include "stm324x9i_eval_io.h"
```

Go to the source code of this file.

Defines

#define	**__STM324x9I_EVAL_BSP_VERSION_MAIN** (0x02) STM324x9I EVAL BSP Driver version number V2.2.2.
#define	**__STM324x9I_EVAL_BSP_VERSION_SUB1** (0x02)
#define	**__STM324x9I_EVAL_BSP_VERSION_SUB2** (0x02)
#define	**__STM324x9I_EVAL_BSP_VERSION_RC** (0x00)
#define	**__STM324x9I_EVAL_BSP_VERSION**

Functions

static void	**I2Cx_MspInit** (void) Initializes I2C MSP.
static void	**I2Cx_Init** (void) Initializes I2C HAL.
static void	**I2Cx_ITConfig** (void) Configures I2C Interrupt.
static void	**I2Cx_Write** (uint8_t Addr, uint8_t Reg, uint8_t Value) Writes a single data.
static uint8_t	**I2Cx_Read** (uint8_t Addr, uint8_t Reg) Reads a single data.
static HAL_StatusTypeDef	**I2Cx_ReadMultiple** (uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t *Buffer, uint16_t Length) Reads multiple data.
static HAL_StatusTypeDef	**I2Cx_WriteMultiple** (uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t *Buffer, uint16_t Length) Writes a value in a register of the device through BUS in using DMA mode.
static HAL_StatusTypeDef	**I2Cx_IsDeviceReady** (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.
void	**IOE_Init** (void) Initializes IOE low level.
void	**IOE_ITConfig** (void) Configures IOE low level interrupt.

void	**IOE_Delay** (uint32_t Delay) IOE delay.
void	**IOE_Write** (uint8_t Addr, uint8_t Reg, uint8_t Value) IOE writes single data.
uint8_t	**IOE_Read** (uint8_t Addr, uint8_t Reg) IOE reads single data.
uint16_t	**IOE_ReadMultiple** (uint8_t Addr, uint8_t Reg, uint8_t *Buffer, uint16_t Length) IOE reads multiple data.
void	**IOE_WriteMultiple** (uint8_t Addr, uint8_t Reg, uint8_t *Buffer, uint16_t Length) IOE writes multiple data.
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, uint8_t Reg, uint8_t Value) Camera writes single data.
uint8_t	**CAMERA_IO_Read** (uint8_t Addr, uint8_t Reg)

	Camera reads single data.
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.
uint32_t	**BSP_GetVersion** (void) This method returns the STM324x9I EVAL BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LED GPIO.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** Button_Mode) Configures button GPIO and EXTI Line.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button)

	Returns the selected button state.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) Configures COM port.
uint8_t	**BSP_JOY_Init** (**JOYMode_TypeDef** Joy_Mode) Configures joystick GPIO and EXTI modes.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the current joystick status.
uint8_t	**BSP_TS3510_IsDetected** (void) Check TS3510 touch screen presence.

Variables

GPIO_TypeDef *	**GPIO_PORT** [**LEDn**]
const uint16_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 I2C_HandleTypeDef	**heval_I2c**

Detailed Description

This file provides a set of firmware functions to manage LEDs, push-buttons and COM ports available on STM324x9I-EVAL evaluation board(MB1045) RevB from STMicroelectronics.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval.c**.

STM324x9I_EVAL BSP User Manual

Defines | Enumerations | Functions

stm324x9i_eval.h File Reference

This file contains definitions for STM324x9I_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** 4

#define **LED1_PIN** GPIO_PIN_6

#define **LED1_GPIO_PORT** GPIOG

#define **LED1_GPIO_CLK_ENABLE**() __GPIOG_CLK_ENABLE()

#define **LED1_GPIO_CLK_DISABLE**() __GPIOG_CLK_DISABLE()

#define **LED2_PIN** GPIO_PIN_7

#define **LED2_GPIO_PORT** GPIOG

#define **LED2_GPIO_CLK_ENABLE**() __GPIOG_CLK_ENABLE()

#define **LED2_GPIO_CLK_DISABLE**() __GPIOG_CLK_DISABLE()

#define **LED3_PIN** GPIO_PIN_10

#define **LED3_GPIO_PORT** GPIOG

#define **LED3_GPIO_CLK_ENABLE**() __GPIOG_CLK_ENABLE()

#define **LED3_GPIO_CLK_DISABLE**() __GPIOG_CLK_DISABLE()

#define **LED4_PIN** GPIO_PIN_12

#define **LED4_GPIO_PORT** GPIOG

#define **LED4_GPIO_CLK_ENABLE**() __GPIOG_CLK_ENABLE()

#define **LED4_GPIO_CLK_DISABLE**() __GPIOG_CLK_DISABLE()

#define **LEDx_GPIO_CLK_ENABLE**(__INDEX__)

#define **LEDx_GPIO_CLK_DISABLE**(__INDEX__)

#define **BUTTONn** 3

#define **WAKEUP_BUTTON_PIN** GPIO_PIN_0
Wakeup push-button.

#define **WAKEUP_BUTTON_GPIO_PORT** GPIOA

#define **WAKEUP_BUTTON_GPIO_CLK_ENABLE**() __GPIOA_CLK

#define **WAKEUP_BUTTON_GPIO_CLK_DISABLE**() __GPIOA_CLK

#define **WAKEUP_BUTTON_EXTI_IRQn** EXTI0_IRQn

#define **TAMPER_BUTTON_PIN** GPIO_PIN_13
Tamper push-button.

#define **TAMPER_BUTTON_GPIO_PORT** GPIOC

#define **TAMPER_BUTTON_GPIO_CLK_ENABLE**() __GPIOC_CLK_

#define **TAMPER_BUTTON_GPIO_CLK_DISABLE**() __GPIOC_CLK

#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**() __GPIOC_CLK_ENA

#define **KEY_BUTTON_GPIO_CLK_DISABLE**() __GPIOC_CLK_DIS

#define **KEY_BUTTON_EXTI_IRQn** EXTI15_10_IRQn

#define **BUTTONx_GPIO_CLK_ENABLE**(__INDEX__)

#define **BUTTONx_GPIO_CLK_DISABLE**(__INDEX__)

#define **COMn** 1

#define **EVAL_COM1** USART1
Definition for COM port1, connected to USART1.

#define **EVAL_COM1_CLK_ENABLE**() __USART1_CLK_ENABLE()

#define **EVAL_COM1_CLK_DISABLE**() __USART1_CLK_DISABLE(

#define **EVAL_COM1_TX_PIN** GPIO_PIN_9

#define **EVAL_COM1_TX_GPIO_PORT** GPIOA

#define **EVAL_COM1_TX_GPIO_CLK_ENABLE**() __GPIOA_CLK_E

#define **EVAL_COM1_TX_GPIO_CLK_DISABLE**() __GPIOA_CLK_D

#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**() __GPIOA_CLK_E

#define **EVAL_COM1_RX_GPIO_CLK_DISABLE**() __GPIOA_CLK_D

#define **EVAL_COM1_RX_AF** GPIO_AF7_USART1

#define **EVAL_COM1_IRQn** USART1_IRQn

#define **EVAL_COMx_CLK_ENABLE**(__INDEX__)

#define **EVAL_COMx_CLK_DISABLE**(__INDEX__)

#define **EVAL_COMx_TX_GPIO_CLK_ENABLE**(__INDEX__)

#define **EVAL_COMx_TX_GPIO_CLK_DISABLE**(__INDEX__)

#define **EVAL_COMx_RX_GPIO_CLK_ENABLE**(__INDEX__)

#define **EVAL_COMx_RX_GPIO_CLK_DISABLE**(__INDEX__)

#define **JOY_SEL_PIN** **IO_PIN_14**
Joystick Pins definition.

#define **JOY_DOWN_PIN** **IO_PIN_13**

#define	**JOY_LEFT_PIN** **IO_PIN_12**
#define	**JOY_RIGHT_PIN** **IO_PIN_11**
#define	**JOY_UP_PIN** **IO_PIN_10**
#define	**JOY_NONE_PIN** **JOY_ALL_PINS**
#define	**JOY_ALL_PINS** (**IO_PIN_10** | **IO_PIN_11** | **IO_PIN_12** | **IO_P** **IO_PIN_14**)
#define	**XSDN_PIN** **IO_PIN_0** Eval Pins definition.
#define	**MII_INT_PIN** **IO_PIN_1**
#define	**RSTI_PIN** **IO_PIN_2**
#define	**CAM_PLUG_PIN** **IO_PIN_3**
#define	**LCD_INT_PIN** **IO_PIN_4**
#define	**AUDIO_INT_PIN** **IO_PIN_5**
#define	**OTG_FS1_OVER_CURRENT_PIN** **IO_PIN_6**
#define	**OTG_FS1_POWER_SWITCH_PIN** **IO_PIN_7**
#define	**OTG_FS2_OVER_CURRENT_PIN** **IO_PIN_8**
#define	**OTG_FS2_POWER_SWITCH_PIN** **IO_PIN_9**
#define	**SD_DETECT_PIN** **IO_PIN_15**
#define	**IO_I2C_ADDRESS** 0x84
#define	**TS_I2C_ADDRESS** 0x82
#define	**TS3510_I2C_ADDRESS** 0x80
#define	**EXC7200_I2C_ADDRESS** 0x08
#define	**CAMERA_I2C_ADDRESS** 0x60
#define	**AUDIO_I2C_ADDRESS** 0x34
#define	**EEPROM_I2C_ADDRESS_A01** 0xA0
#define	**EEPROM_I2C_ADDRESS_A02** 0xA6
#define	**EVAL_I2Cx** I2C1
#define	**EVAL_I2Cx_CLK_ENABLE**() __I2C1_CLK_ENABLE()
#define	**EVAL_DMAx_CLK_ENABLE**() __DMA1_CLK_ENABLE()
#define	**EVAL_I2Cx_SCL_SDA_GPIO_CLK_ENABLE**() __GPIOB_C
#define	**EVAL_I2Cx_FORCE_RESET**() __I2C1_FORCE_RESET()
#define	**EVAL_I2Cx_RELEASE_RESET**() __I2C1_RELEASE_RESE
#define	**EVAL_I2Cx_SCL_PIN** GPIO_PIN_6
#define	**EVAL_I2Cx_SCL_SDA_GPIO_PORT** GPIOB

```
#define EVAL_I2Cx_SCL_SDA_AF   GPIO_AF4_I2C1
#define EVAL_I2Cx_SDA_PIN   GPIO_PIN_9
#define EVAL_I2Cx_EV_IRQn   I2C1_EV_IRQn
#define EVAL_I2Cx_ER_IRQn   I2C1_ER_IRQn
```

Enumerations

enum	**Led_TypeDef** { **LED1** = 0, **LED2** = 1, **LED3** = 2, **LED4** = 3 }
enum	**Button_TypeDef** { **BUTTON_WAKEUP** = 0, **BUTTON_TAMPER** = 1, **BUTTON_KEY** = 2 }
enum	**ButtonMode_TypeDef** { **BUTTON_MODE_GPIO** = 0, **BUTTON_MODE_EXTI** = 1 }
enum	**JOYMode_TypeDef** { **JOY_MODE_GPIO** = 0, **JOY_MODE_EXTI** = 1 }
enum	**JOYState_TypeDef** { **JOY_NONE** = 0, **JOY_SEL** = 1, **JOY_DOWN** = 2, **JOY_LEFT** = 3, **JOY_RIGHT** = 4, **JOY_UP** = 5 }
enum	**COM_TypeDef** { **COM1** = 0, **COM2** = 1 }

Functions

uint32_t	**BSP_GetVersion** (void) This method returns the STM324x9I EVAL BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LED GPIO.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** Button_Mode) Configures button GPIO and EXTI Line.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *husart) Configures COM port.
uint8_t	**BSP_JOY_Init** (**JOYMode_TypeDef** Joy_Mode) Configures joystick GPIO and EXTI modes.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the current joystick status.
uint8_t	**BSP_TS3510_IsDetected** (void) Check TS3510 touch screen presence.

Detailed Description

This file contains definitions for STM324x9I_EVAL's LEDs, push-buttons and COM ports hardware resources.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval.h**.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_audio.c File Reference

This file provides the Audio driver for the STM324x9I-EVAL evaluation board. More...

```
#include "stm324x9i_eval_audio.h"
```

Go to the source code of this file.

Functions

static void	**SAIx_MspInit** (void) Initializes SAI MSP.
static void	**SAIx_Init** (uint32_t AudioFreq) Initializes the Audio Codec audio interface (SAI).
static void	**I2Sx_MspInit** (void) AUDIO IN I2S MSP Init.
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_Init** (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	**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.
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.
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.

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 STM324x9I-EVAL evaluation board.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_audio.c**.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_audio.h File Reference

This file contains the common defines and functions prototypes for the **stm324x9i_eval_audio.c** driver. More...

```
#include "../Components/wm8994/wm8994.h" #include
"stm324x9i_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** SAI1_Block_B
#define	**AUDIO_SAIx_CLK_ENABLE**() __SAI1_CLK_ENABLE()
#define	**AUDIO_SAIx_MCLK_SCK_SD_FS_AF** GPIO_AF6_SAI1
#define	**AUDIO_SAIx_MCLK_SCK_SD_FS_ENABLE**() __GPIOF_CL
#define	**AUDIO_SAIx_FS_PIN** GPIO_PIN_9
#define	**AUDIO_SAIx_SCK_PIN** GPIO_PIN_8
#define	**AUDIO_SAIx_SD_PIN** GPIO_PIN_6
#define	**AUDIO_SAIx_MCK_PIN** GPIO_PIN_7
#define	**AUDIO_SAIx_MCLK_SCK_SD_FS_GPIO_PORT** GPIOF
#define	**AUDIO_SAIx_DMAx_CLK_ENABLE**() __DMA2_CLK_ENAB
#define	**AUDIO_SAIx_DMAx_STREAM** DMA2_Stream5
#define	**AUDIO_SAIx_DMAx_CHANNEL** DMA_CHANNEL_0
#define	**AUDIO_SAIx_DMAx_IRQ** DMA2_Stream5_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_Stream5_IRQHan
#define	**AUDIO_OUT_IRQ_PREPRIO** 5 /* Select the preemption prio
#define	**AUDIO_I2Sx** SPI3
#define	**AUDIO_I2Sx_CLK_ENABLE**() __SPI3_CLK_ENABLE()
#define	**AUDIO_I2Sx_SCK_PIN** GPIO_PIN_3
#define	**AUDIO_I2Sx_SCK_GPIO_PORT** GPIOB
#define	**AUDIO_I2Sx_SCK_GPIO_CLK_ENABLE**() __GPIOB_CLK_
#define	**AUDIO_I2Sx_SCK_AF** GPIO_AF6_SPI3
#define	**AUDIO_I2Sx_SD_PIN** GPIO_PIN_6
#define	**AUDIO_I2Sx_SD_GPIO_PORT** GPIOD
#define	**AUDIO_I2Sx_SD_GPIO_CLK_ENABLE**() __GPIOD_CLK_E

```
#define AUDIO_I2Sx_SD_AF   GPIO_AF5_I2S3ext
#define AUDIO_I2Sx_DMAx_CLK_ENABLE()   __DMA1_CLK_ENAB
#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_IRQHanc
#define AUDIO_IN_IRQ_PREPRIO   6 /* Select the preemption priority
#define AUDIO_TIMx_CLK_ENABLE()   __TIM3_CLK_ENABLE()
#define AUDIO_TIMx_CLK_DISABLE()   __TIM3_CLK_DISABLE()
#define AUDIO_TIMx   TIM3
#define AUDIO_TIMx_IN_CHANNEL   TIM_CHANNEL_1
#define AUDIO_TIMx_OUT_CHANNEL   TIM_CHANNEL_2 /* Select c
#define AUDIO_TIMx_GPIO_CLK_ENABLE()   __GPIOC_CLK_ENAE
#define AUDIO_TIMx_GPIO   GPIOC
#define AUDIO_TIMx_IN_GPIO_PIN   GPIO_PIN_6
#define AUDIO_TIMx_OUT_GPIO_PIN   GPIO_PIN_7
#define AUDIO_TIMx_AF   GPIO_AF2_TIM3
#define AUDIODATA_SIZE   2 /* 16-bits audio data size */
#define AUDIO_OK   0
#define AUDIO_ERROR   1
#define AUDIO_TIMEOUT   2
#define DEFAULT_AUDIO_IN_FREQ   I2S_AUDIOFREQ_16K
#define DEFAULT_AUDIO_IN_BIT_RESOLUTION   16
#define DEFAULT_AUDIO_IN_CHANNEL_NBR   2 /* Mono = 1, Stere
#define DEFAULT_AUDIO_IN_VOLUME   64
#define INTERNAL_BUFF_SIZE   128*DEFAULT_AUDIO_IN_FREQ/1
#define PCM_OUT_SIZE   DEFAULT_AUDIO_IN_FREQ/1000*2
#define CHANNEL_DEMUX_MASK   0x55
#define CODEC_RESET_DELAY   5
#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_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.
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.
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.

Variables

__IO uint16_t	**AudioInVolume**

Detailed Description

This file contains the common defines and functions prototypes for the **stm324x9i_eval_audio.c** driver.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_audio.h**.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_camera.c File Reference

This file includes the driver for Camera modules mounted on STM324x9I-EVAL evaluation board. More...

```
#include "stm324x9i_eval_camera.h"
```

Go to the source code of this file.

Functions

static void	**DCMI_MspInit** (void) Initializes the DCMI MSP.
static uint32_t	**GetSize** (uint32_t resolution) Get the capture size.
uint8_t	**BSP_CAMERA_Init** (uint32_t Resolution) Initializes 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_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	**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

static DCMI_HandleTypeDef	**hdcmi_eval**
CAMERA_DrvTypeDef *	**camera_drv**
uint32_t	**current_resolution**

Detailed Description

This file includes the driver for Camera modules mounted on STM324x9I-EVAL evaluation board.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_camera.c**.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_camera.h File Reference

This file contains the common defines and functions prototypes for the **stm324x9i_eval_camera.c** driver. More...

```
#include "../Components/ov2640/ov2640.h" #include
"stm324x9i_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.
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_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.

Detailed Description

This file contains the common defines and functions prototypes for the **stm324x9i_eval_camera.c** driver.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_camera.h**.

STM324x9I_EVAL BSP User Manual

stm324x9i_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 "stm324x9i_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.
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 uint32_t	**EEPROMTimeout** = **EEPROM_READ_TIMEOUT**
__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:
V2.2.2

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 STM324x9I_EVAL board. To use this driver you have to connect the ANT7-M24LR-A to CN3 connector of STM324x9I_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 **stm324x9i_eval.h** file, and the initialization is performed in **EEPROM_IO_Init()** function declared in **stm324x9i_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:

Definition in file **stm324x9i_eval_eeprom.c**.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_eeprom.h File Reference

This file contains all the functions prototypes for the **stm324x9i_eval_eeprom.c** firmware driver. More...

```
#include "stm324x9i_eval.h"
```

Go to the source code of this file.

Defines

```
#define EEPROM_PAGESIZE 4
#define EEPROM_MAX_SIZE 0x2000 /* 64Kbit */
#define EEPROM_READ_TIMEOUT ((uint32_t)(1000))
#define EEPROM_WRITE_TIMEOUT ((uint32_t)(10))
#define EEPROM_MAX_TRIALS 3000
#define EEPROM_OK 0
#define EEPROM_FAIL 1
#define EEPROM_TIMEOUT 2
```

Functions

uint32_t	**BSP_EEPROM_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
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 **stm324x9i_eval_eeprom.c** firmware driver.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_eeprom.h**.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_io.c File Reference

This file provides a set of functions needed to manage the IO pins on STM324x9I-EVAL evaluation board. More...

```
#include "stm324x9i_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 (GPIOs, clocks..).
uint8_t	**BSP_IO_ITGetStatus** (uint16_t IO_Pin) Gets the selected pins IT status.
void	**BSP_IO_ITClear** (void) Clears all the IO IT pending bits.
uint8_t	**BSP_IO_ConfigPin** (uint16_t IO_Pin, IO_ModeTypedef IO_Mode) Configures the IO pin(s) according to IO mode structure value.
void	**BSP_IO_WritePin** (uint16_t IO_Pin, uint8_t PinState) Sets the selected pins state.
uint16_t	**BSP_IO_ReadPin** (uint16_t IO_Pin) Gets the selected pins current state.
void	**BSP_IO_TogglePin** (uint16_t IO_Pin) Toggles the selected pins state.

Variables

static IO_DrvTypeDef *	**io_driver**

Detailed Description

This file provides a set of functions needed to manage the IO pins on STM324x9I-EVAL evaluation board.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_io.c**.

STM324x9I_EVAL BSP User Manual

Data Structures | Defines | Enumerations | Functions

stm324x9i_eval_io.h File Reference

This file contains the common defines and functions prototypes for the **stm324x9i_eval_io.c** driver. More...

```
#include "stm324x9i_eval.h" #include
"../Components/stmpe1600/stmpe1600.h"
```

Go to the source code of this file.

Data Structures

struct **IO_StateTypeDef**

Defines

#define	**IO_PIN_0** 0x0001
#define	**IO_PIN_1** 0x0002
#define	**IO_PIN_2** 0x0004
#define	**IO_PIN_3** 0x0008
#define	**IO_PIN_4** 0x0010
#define	**IO_PIN_5** 0x0020
#define	**IO_PIN_6** 0x0040
#define	**IO_PIN_7** 0x0080
#define	**IO_PIN_8** 0x0100
#define	**IO_PIN_9** 0x0200
#define	**IO_PIN_10** 0x0400
#define	**IO_PIN_11** 0x0800
#define	**IO_PIN_12** 0x1000
#define	**IO_PIN_13** 0x2000
#define	**IO_PIN_14** 0x4000
#define	**IO_PIN_15** 0x8000
#define	**IO_PIN_ALL** 0xFFFF

Enumerations

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 (GPIOs, clocks..).
uint8_t	**BSP_IO_ITGetStatus** (uint16_t IO_Pin) Gets the selected pins IT status.
void	**BSP_IO_ITClear** (void) Clears all the IO IT pending bits.
uint8_t	**BSP_IO_ConfigPin** (uint16_t IO_Pin, IO_ModeTypedef IO_Mode) Configures the IO pin(s) according to IO mode structure value.
void	**BSP_IO_WritePin** (uint16_t IO_Pin, uint8_t PinState) Sets the selected pins state.
uint16_t	**BSP_IO_ReadPin** (uint16_t IO_Pin) Gets the selected pins current state.
void	**BSP_IO_TogglePin** (uint16_t IO_Pin) Toggles the selected pins state.

Detailed Description

This file contains the common defines and functions prototypes for the **stm324x9i_eval_io.c** driver.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_io.h**.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_lcd.c File Reference

This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM324x9I-EVAL evaluation board. More...

```
#include "stm324x9i_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	**MspInit** (void) Initializes the LTDC MSP.
static void	**DrawChar** (uint16_t Xpos, uint16_t Ypos, const uint8_t *c) Draws a character on LCD.
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).
static void	**LL_FillBuffer** (uint32_t LayerIndex, void *pDst, uint32_t xSize, uint32_t ySize, uint32_t OffLine, uint32_t ColorIndex) Fills a buffer.
static void	**LL_ConvertLineToARGB8888** (void *pSrc, void *pDst, uint32_t xSize, uint32_t ColorMode) Converts a line to an ARGB8888 pixel format.
uint8_t	**BSP_LCD_Init** (void) Initializes the LCD.
uint8_t	**BSP_LCD_InitEx** (uint32_t PclkConfig) Initializes the LCD.
uint32_t	**BSP_LCD_GetXSize** (void) Gets the LCD X size.
uint32_t	**BSP_LCD_GetYSize** (void) Gets the LCD Y size.
void	**BSP_LCD_LayerDefaultInit** (uint16_t LayerIndex, uint32_t FB_Address) Initializes the LCD layers.
void	**BSP_LCD_SelectLayer** (uint32_t LayerIndex) Selects the LCD Layer.
void	**BSP_LCD_SetLayerVisible** (uint32_t LayerIndex, FunctionalState State) Sets an LCD Layer visible.

void	**BSP_LCD_SetTransparency** (uint32_t LayerIndex, uint8_t Transparency) Configures the transparency.
void	**BSP_LCD_SetLayerAddress** (uint32_t LayerIndex, uint32_t Address) Sets an LCD layer frame buffer address.
void	**BSP_LCD_SetLayerWindow** (uint16_t LayerIndex, uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Sets display window.
void	**BSP_LCD_SetColorKeying** (uint32_t LayerIndex, uint32_t RGBValue) Configures and sets the color keying.
void	**BSP_LCD_ResetColorKeying** (uint32_t LayerIndex) Disables the color keying.
void	**BSP_LCD_SetTextColor** (uint32_t Color) Sets the LCD text color.
uint32_t	**BSP_LCD_GetTextColor** (void) Gets the LCD text color.
void	**BSP_LCD_SetBackColor** (uint32_t Color) Sets the LCD background color.
uint32_t	**BSP_LCD_GetBackColor** (void) Gets 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.
uint32_t	**BSP_LCD_ReadPixel** (uint16_t Xpos, uint16_t Ypos) Reads an LCD pixel.
void	**BSP_LCD_Clear** (uint32_t Color) Clears the hole LCD.
void	**BSP_LCD_ClearStringLine** (uint32_t Line) Clears the selected line.
	BSP_LCD_DisplayChar (uint16_t Xpos, uint16_t Ypos,

void	uint8_t Ascii) Displays one character.
void	**BSP_LCD_DisplayStringAt** (uint16_t Xpos, uint16_t Ypos, uint8_t *Text, **Text_AlignModeTypdef** Mode) Displays characters on the LCD.
void	**BSP_LCD_DisplayStringAtLine** (uint16_t Line, uint8_t *ptr) Displays a maximum of 60 characters on the 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** (uint32_t Xpos, uint32_t Ypos, uint8_t *pbmp) Draws a bitmap picture loaded in the internal Flash (32 bpp).
void	**BSP_LCD_FillRect** (uint16_t Xpos, uint16_t Ypos,

	uint16_t Width, uint16_t Height) Draws a full rectangle.
void	**BSP_LCD_FillCircle** (uint16_t Xpos, uint16_t Ypos, uint16_t Radius) Draws a full circle.
void	**BSP_LCD_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.
__weak void	**BSP_LCD_ClockConfig** (LTDC_HandleTypeDef *hltdc, void *Params) Clock Config.
void	**BSP_LCD_DrawPixel** (uint16_t Xpos, uint16_t Ypos, uint32_t RGB_Code) Draws a pixel on LCD.

Variables

static LTDC_HandleTypeDef	**hltdc_eval**
static DMA2D_HandleTypeDef	**hdma2d_eval**
static uint32_t	**PCLK_profile** = **LCD_MAX_PCLK**
static uint32_t	**ActiveLayer** = 0
static **LCD_DrawPropTypeDef**	**DrawProp** [**MAX_LAYER_NUMBER**]

Detailed Description

This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM324x9I-EVAL evaluation board.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_lcd.c**.

STM324x9I_EVAL BSP User Manual

Data Structures | Defines | Typedefs | Enumerations | Functions

stm324x9i_eval_lcd.h File Reference

This file contains the common defines and functions prototypes for the **stm324x9i_eval_lcd.c** driver. More...

```
#include "../Components/ampire640480/ampire640480.h" #include
"../Components/ampire480272/ampire480272.h"
#include "../Components/stmpe811/stmpe811.h"
#include "stm324x9i_eval_sdram.h"
#include "stm324x9i_eval.h"
#include "../../../Utilities/Fonts/fonts.h"
```

Go to the source code of this file.

Data Structures

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

Defines

#define	**MAX_LAYER_NUMBER** 2
#define	**LCD_LayerCfgTypeDef** LTDC_LayerCfgTypeDef
#define	**LCD_OK** 0x00 LCD status structure definition.
#define	**LCD_ERROR** 0x01
#define	**LCD_TIMEOUT** 0x02
#define	**LCD_FB_START_ADDRESS** ((uint32_t)0xC0000000) LCD FB_StartAddress.
#define	**LCD_MAX_PCLK** ((uint8_t)0x00)
#define	**LCD_MIN_PCLK** ((uint8_t)0x01)
#define	**LCD_COLOR_BLUE** 0xFF0000FF LCD color.
#define	**LCD_COLOR_GREEN** 0xFF00FF00
#define	**LCD_COLOR_RED** 0xFFFF0000
#define	**LCD_COLOR_CYAN** 0xFF00FFFF
#define	**LCD_COLOR_MAGENTA** 0xFFFF00FF
#define	**LCD_COLOR_YELLOW** 0xFFFFFF00
#define	**LCD_COLOR_LIGHTBLUE** 0xFF8080FF
#define	**LCD_COLOR_LIGHTGREEN** 0xFF80FF80
#define	**LCD_COLOR_LIGHTRED** 0xFFFF8080
#define	**LCD_COLOR_LIGHTCYAN** 0xFF80FFFF
#define	**LCD_COLOR_LIGHTMAGENTA** 0xFFFF80FF
#define	**LCD_COLOR_LIGHTYELLOW** 0xFFFFFF80
#define	**LCD_COLOR_DARKBLUE** 0xFF000080
#define	**LCD_COLOR_DARKGREEN** 0xFF008000
#define	**LCD_COLOR_DARKRED** 0xFF800000
#define	**LCD_COLOR_DARKCYAN** 0xFF008080
#define	**LCD_COLOR_DARKMAGENTA** 0xFF800080
#define	**LCD_COLOR_DARKYELLOW** 0xFF808000
#define	**LCD_COLOR_WHITE** 0xFFFFFFFF
#define	**LCD_COLOR_LIGHTGRAY** 0xFFD3D3D3

```
#define LCD_COLOR_GRAY 0xFF808080
#define LCD_COLOR_DARKGRAY 0xFF404040
#define LCD_COLOR_BLACK 0xFF000000
#define LCD_COLOR_BROWN 0xFFA52A2A
#define LCD_COLOR_ORANGE 0xFFFFA500
#define LCD_COLOR_TRANSPARENT 0xFF000000
#define LCD_DEFAULT_FONT Font24
```

LCD default font.

Typedefs

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

Enumerations

enum	**Text_AlignModeTypdef** { **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_InitEx** (uint32_t PclkConfig) Initializes the LCD.
uint32_t	**BSP_LCD_GetXSize** (void) Gets the LCD X size.
uint32_t	**BSP_LCD_GetYSize** (void) Gets the LCD Y size.
void	**BSP_LCD_LayerDefaultInit** (uint16_t LayerIndex, uint32_t FrameBuffer) Initializes the LCD layers.
void	**BSP_LCD_SetTransparency** (uint32_t LayerIndex, uint8_t Transparency) Configures the transparency.
void	**BSP_LCD_SetLayerAddress** (uint32_t LayerIndex, uint32_t Address) Sets an LCD layer frame buffer address.
void	**BSP_LCD_SetColorKeying** (uint32_t LayerIndex, uint32_t RGBValue) Configures and sets the color keying.
void	**BSP_LCD_ResetColorKeying** (uint32_t LayerIndex) Disables the color keying.
void	**BSP_LCD_SetLayerWindow** (uint16_t LayerIndex, uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Sets display window.
void	**BSP_LCD_SelectLayer** (uint32_t LayerIndex) Selects the LCD Layer.
void	**BSP_LCD_SetLayerVisible** (uint32_t LayerIndex, FunctionalState State) Sets an LCD Layer visible.

void	**BSP_LCD_SetTextColor** (uint32_t Color) Sets the LCD text color.
uint32_t	**BSP_LCD_GetTextColor** (void) Gets the LCD text color.
void	**BSP_LCD_SetBackColor** (uint32_t Color) Sets the LCD background color.
uint32_t	**BSP_LCD_GetBackColor** (void) Gets 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.
uint32_t	**BSP_LCD_ReadPixel** (uint16_t Xpos, uint16_t Ypos) Reads an LCD pixel.
void	**BSP_LCD_DrawPixel** (uint16_t Xpos, uint16_t Ypos, uint32_t pixel) Draws a pixel on LCD.
void	**BSP_LCD_Clear** (uint32_t Color) Clears the hole LCD.
void	**BSP_LCD_ClearStringLine** (uint32_t Line) Clears the selected line.
void	**BSP_LCD_DisplayStringAtLine** (uint16_t Line, uint8_t *ptr) Displays a maximum of 60 characters on the LCD.
void	**BSP_LCD_DisplayStringAt** (uint16_t Xpos, uint16_t Ypos, uint8_t *Text, **Text_AlignModeTypdef** Mode) Displays characters on the LCD.
void	**BSP_LCD_DisplayChar** (uint16_t Xpos, uint16_t Ypos, uint8_t Ascii) Displays one character.
void	**BSP_LCD_DrawHLine** (uint16_t Xpos, uint16_t Ypos, uint16_t Length) Draws an horizontal line.
	BSP_LCD_DrawVLine (uint16_t Xpos, uint16_t Ypos,

void	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** (uint32_t Xpos, uint32_t Ypos, uint8_t *pbmp) Draws a bitmap picture loaded in the internal Flash (32 bpp).
void	**BSP_LCD_FillRect** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Draws a full rectangle.
void	**BSP_LCD_FillCircle** (uint16_t Xpos, uint16_t Ypos, uint16_t Radius) Draws a full circle.
void	**BSP_LCD_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.
void	**BSP_LCD_ClockConfig** (LTDC_HandleTypeDef *hltdc, void *Params) Clock Config.

Detailed Description

This file contains the common defines and functions prototypes for the **stm324x9i_eval_lcd.c** driver.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_lcd.h**.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_nor.c File Reference

This file includes a standard driver for the M29W256GL70ZA6E NOR flash memory device mounted on STM324x9I-EVAL evaluation board. More...

```
#include "stm324x9i_eval_nor.h"
```

Go to the source code of this file.

Functions

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

Variables

static NOR_HandleTypeDef	**norHandle**
static FMC_NORSRAM_TimingTypeDef	**Timing**

Detailed Description

This file includes a standard driver for the M29W256GL70ZA6E NOR flash memory device mounted on STM324x9I-EVAL evaluation board.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_nor.c**.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_nor.h File Reference

This file contains the common defines and functions prototypes for the **stm324x9i_eval_nor.c** driver. More...

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

Go to the source code of this file.

Defines

#define	**NOR_STATUS_OK** 0x00 NOR status structure definition.
#define	**NOR_STATUS_ERROR** 0x01
#define	**NOR_DEVICE_ADDR** ((uint32_t)0x60000000)
#define	**NOR_MEMORY_WIDTH** FMC_NORSRAM_MEM_BUS_WID
#define	**NOR_BURSTACCESS** FMC_BURST_ACCESS_MODE_DIS
#define	**NOR_WRITEBURST** FMC_WRITE_BURST_DISABLE
#define	**CONTINUOUSCLOCK_FEATURE** FMC_CONTINUOUS_CL
#define	**BLOCKERASE_TIMEOUT** ((uint32_t)0x00A00000) /* NOR b */
#define	**CHIPERASE_TIMEOUT** ((uint32_t)0x30000000) /* NOR chip
#define	**PROGRAM_TIMEOUT** ((uint32_t)0x00004400) /* NOR progra
#define	**NOR_READY_BUSY_PIN** GPIO_PIN_6
#define	**NOR_READY_BUSY_GPIO** GPIOD
#define	**NOR_READY_STATE** GPIO_PIN_SET
#define	**NOR_BUSY_STATE** GPIO_PIN_RESET

Functions

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

Detailed Description

This file contains the common defines and functions prototypes for the **stm324x9i_eval_nor.c** driver.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_nor.h**.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_sd.c File Reference

This file includes the uSD card driver mounted on STM324x9I-EVAL evaluation board. More...

```
#include "stm324x9i_eval_sd.h"
```

Go to the source code of this file.

Functions

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

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

Variables

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

Detailed Description

This file includes the uSD card driver mounted on STM324x9I-EVAL evaluation board.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_sd.c**.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_sd.h File Reference

This file contains the common defines and functions prototypes for the **stm324x9i_eval_sd.c** driver. More...

```
#include "stm32f4xx_hal.h" #include "stm324x9i_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** 0x00 SD status structure definition.
#define	**MSD_ERROR** 0x01
#define	**SD_PRESENT** ((uint8_t)0x01)
#define	**SD_NOT_PRESENT** ((uint8_t)0x00)
#define	**SD_DATATIMEOUT** ((uint32_t)100000000)
#define	**__DMAx_TxRx_CLK_ENABLE** __DMA2_CLK_ENABLE
#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_ITConfig** (void) Configures Interrupt mode for SD detection pin.
void	**BSP_SD_DetectIT** (void) SD detect IT treatment.
void	**BSP_SD_DetectCallback** (void) SD detect IT detection callback.
uint8_t	**BSP_SD_ReadBlocks** (uint32_t *pData, uint64_t ReadAddr, uint32_t BlockSize, uint32_t NumOfBlocks) Reads block(s) from a specified address in an SD card, in polling mode.
uint8_t	**BSP_SD_WriteBlocks** (uint32_t *pData, uint64_t WriteAddr, uint32_t BlockSize, uint32_t NumOfBlocks) Writes block(s) to a specified address in an SD card, in polling mode.
uint8_t	**BSP_SD_ReadBlocks_DMA** (uint32_t *pData, uint64_t ReadAddr, uint32_t BlockSize, uint32_t NumOfBlocks) Reads block(s) from a specified address in an SD card, in DMA mode.
uint8_t	**BSP_SD_WriteBlocks_DMA** (uint32_t *pData, uint64_t WriteAddr, uint32_t BlockSize, uint32_t NumOfBlocks)

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

Detailed Description

This file contains the common defines and functions prototypes for the **stm324x9i_eval_sd.c** driver.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_sd.h**.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_sdram.c File Reference

This file includes the SDRAM driver for the MT48LC4M32B2B5-7 memory device mounted on STM324x9I-EVAL evaluation board. More...

```
#include "stm324x9i_eval_sdram.h"
```

Go to the source code of this file.

Functions

static void	**SDRAM_MspInit** (void) Initializes SDRAM MSP.
uint8_t	**BSP_SDRAM_Init** (void) Initializes 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.

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 STM324x9I-EVAL evaluation board.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_sdram.c**.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_sdram.h File Reference

This file contains the common defines and functions prototypes for the **stm324x9i_eval_sdram.c** driver. More...

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

Go to the source code of this file.

Defines

#define	**SDRAM_OK** 0x00 SDRAM status structure definition.
#define	**SDRAM_ERROR** 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** __DMA2_CLK_ENABLE
#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.
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.

Detailed Description

This file contains the common defines and functions prototypes for the **stm324x9i_eval_sdram.c** driver.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_sdram.h**.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_sram.c File Reference

This file includes the SRAM driver for the IS61WV102416BLL-10M memory device mounted on STM324x9I-EVAL evaluation board. More...

```
#include "stm324x9i_eval_sram.h"
```

Go to the source code of this file.

Functions

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

Variables

static SRAM_HandleTypeDef	**sramHandle**
static FMC_NORSRAM_TimingTypeDef	**Timing**

Detailed Description

This file includes the SRAM driver for the IS61WV102416BLL-10M memory device mounted on STM324x9I-EVAL evaluation board.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_sram.c**.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_sram.h File Reference

This file contains the common defines and functions prototypes for the **stm324x9i_eval_sram.c** driver. More...

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

Go to the source code of this file.

Defines

#define	**SRAM_OK** 0x00 SRAM status structure definition.
#define	**SRAM_ERROR** 0x01
#define	**SRAM_DEVICE_ADDR** ((uint32_t)0x64000000)
#define	**SRAM_DEVICE_SIZE** ((uint32_t)0x200000) /* SRAM device
#define	**SRAM_MEMORY_WIDTH** FMC_NORSRAM_MEM_BUS_WI
#define	**SRAM_BURSTACCESS** FMC_BURST_ACCESS_MODE_DI
#define	**SRAM_WRITEBURST** FMC_WRITE_BURST_DISABLE
#define	**CONTINUOUSCLOCK_FEATURE** FMC_CONTINUOUS_CL
#define	**__SRAM_DMAx_CLK_ENABLE** __DMA2_CLK_ENABLE
#define	**SRAM_DMAx_CHANNEL** DMA_CHANNEL_0
#define	**SRAM_DMAx_STREAM** DMA2_Stream0
#define	**SRAM_DMAx_IRQn** DMA2_Stream0_IRQn
#define	**SRAM_DMAx_IRQHandler** DMA2_Stream0_IRQHandler

Functions

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

Detailed Description

This file contains the common defines and functions prototypes for the **stm324x9i_eval_sram.c** driver.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_sram.h**.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_ts.c File Reference

This file provides a set of functions needed to manage the Touch Screen on STM324x9I-EVAL evaluation board. More...

```
#include "stm324x9i_eval_ts.h" #include "stm324x9i_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_ITConfig** (void) Configures and enables the touch screen interrupts.
uint8_t	**BSP_TS_ITGetStatus** (void) Gets the touch screen interrupt status.
uint8_t	**BSP_TS_GetState** (**TS_StateTypeDef** *TS_State) Returns status and positions of the touch screen.
void	**BSP_TS_ITClear** (void) Clears all touch screen interrupts.

Variables

static TS_DrvTypeDef *	**ts_driver**
static uint16_t	**ts_x_boundary**
static uint16_t	**ts_y_boundary**
static uint8_t	**ts_orientation**
static uint8_t	**I2C_Address**

Detailed Description

This file provides a set of functions needed to manage the Touch Screen on STM324x9I-EVAL evaluation board.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_ts.c**.

STM324x9I_EVAL BSP User Manual

Data Structures | Defines | Enumerations | Functions

stm324x9i_eval_ts.h File Reference

This file contains the common defines and functions prototypes for the **stm324x9i_eval_ts.c** driver. More...

```
#include "stm324x9i_eval.h" #include
"../Components/stmpe811/stmpe811.h"
#include "../Components/ts3510/ts3510.h"
#include "../Components/exc7200/exc7200.h"
```

Go to the source code of this file.

Data Structures

struct **TS_StateTypeDef**

Defines

#define **TS_SWAP_NONE** 0x00
#define **TS_SWAP_X** 0x01
#define **TS_SWAP_Y** 0x02
#define **TS_SWAP_XY** 0x04
#define **TS_INT_PIN** 0x0010

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_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.

Detailed Description

This file contains the common defines and functions prototypes for the **stm324x9i_eval_ts.c** driver.

Author:
MCD Application Team

Version:
V2.2.2

Date:
13-January-2016

Attention:

Definition in file **stm324x9i_eval_ts.h**.

STM324x9I_EVAL BSP User Manual

Modules

Here is a list of all modules:

- **STM324x9I EVAL TS Exported Constants**
- **STM324x9I EVAL TS Exported Macros**
- **STM324x9I EVAL TS Exported Functions**

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP
User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

Data Structures

Here are the data structures with brief descriptions:

IO_StateTypeDef	
LCD_DrawPropTypeDef	
Point	
TS_StateTypeDef	

STM324x9I_EVAL BSP User Manual

File List

Here is a list of all files with brief descriptions:

stm324x9i_eval.c [code]	This file provides a set of firmware functions to manage LEDs, push-buttons and COM ports available on STM324x9I-EVAL evaluation board(MB1045) RevB from STMicroelectronics
stm324x9i_eval.h [code]	This file contains definitions for STM324x9I_EVAL's LEDs, push-buttons and COM ports hardware resources
stm324x9i_eval_audio.c [code]	This file provides the Audio driver for the STM324x9I-EVAL evaluation board
stm324x9i_eval_audio.h [code]	This file contains the common defines and functions prototypes for the **stm324x9i_eval_audio.c** driver
stm324x9i_eval_camera.c [code]	This file includes the driver for Camera modules mounted on STM324x9I-EVAL evaluation board

stm324x9i_eval_camera.h [code]	This file contains the common defines and functions prototypes for the **stm324x9i_eval_camera.c** driver
stm324x9i_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
stm324x9i_eval_eeprom.h [code]	This file contains all the functions prototypes for the **stm324x9i_eval_eeprom.c** firmware driver
stm324x9i_eval_io.c [code]	This file provides a set of functions needed to manage the IO pins on STM324x9I-EVAL evaluation board
stm324x9i_eval_io.h [code]	This file contains the common defines and functions prototypes for the **stm324x9i_eval_io.c** driver
stm324x9i_eval_lcd.c [code]	This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM324x9I-EVAL evaluation board
stm324x9i_eval_lcd.h [code]	This file contains the common defines and functions prototypes for the **stm324x9i_eval_lcd.c** driver

stm324x9i_eval_nor.c [code]	This file includes a standard driver for the M29W256GL70ZA6E NOR flash memory device mounted on STM324x9I-EVAL evaluation board
stm324x9i_eval_nor.h [code]	This file contains the common defines and functions prototypes for the **stm324x9i_eval_nor.c** driver
stm324x9i_eval_sd.c [code]	This file includes the uSD card driver mounted on STM324x9I-EVAL evaluation board
stm324x9i_eval_sd.h [code]	This file contains the common defines and functions prototypes for the **stm324x9i_eval_sd.c** driver
stm324x9i_eval_sdram.c [code]	This file includes the SDRAM driver for the MT48LC4M32B2B5-7 memory device mounted on STM324x9I-EVAL evaluation board
stm324x9i_eval_sdram.h [code]	This file contains the common defines and functions prototypes for the **stm324x9i_eval_sdram.c** driver
stm324x9i_eval_sram.c [code]	This file includes the SRAM driver for the IS61WV102416BLL-10M memory device mounted on STM324x9I-EVAL evaluation board
stm324x9i_eval_sram.h [code]	This file contains the common

	defines and functions prototypes for the **stm324x9i_eval_sram.c** driver
stm324x9i_eval_ts.c [code]	This file provides a set of functions needed to manage the Touch Screen on STM324x9I-EVAL evaluation board
stm324x9i_eval_ts.h [code]	This file contains the common defines and functions prototypes for the **stm324x9i_eval_ts.c** driver

STM324x9I_EVAL BSP User Manual

Directories

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

- Drivers
  - BSP
    - STM324x9I_EVAL

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL LOW LEVEL

STM324x9I EVAL

Modules

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL AUDIO

This file includes the low layer driver for wm8994 Audio Codec available on STM324x9I-EVAL evaluation board(MB1045). More...

Modules

- STM324x9I EVAL AUDIO Private Types
- STM324x9I EVAL AUDIO Private Defines
- STM324x9I EVAL AUDIO Private Macros
- STM324x9I EVAL AUDIO Private Variables
- STM324x9I EVAL AUDIO Private Function Prototypes
- STM324x9I EVAL AUDIO OUT Private Functions
- STM324x9I EVAL AUDIO Exported Types
- STM324x9I EVAL AUDIO Exported Constants
- STM324x9I EVAL AUDIO Exported Variables
- STM324x9I EVAL AUDIO Exported Macros
- STM324x9I EVAL AUDIO OUT Exported Functions
- STM324x9I EVAL AUDIO IN Exported Functions

Detailed Description

This file includes the low layer driver for wm8994 Audio Codec available on STM324x9I-EVAL evaluation board(MB1045).

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL CAMERA

STM324x9I EVAL

Modules

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL EEPROM

STM324x9I EVAL

This file includes the I2C EEPROM driver of STM324x9I-EVAL evaluation board. More...

Modules

Detailed Description

This file includes the I2C EEPROM driver of STM324x9I-EVAL evaluation board.

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL IO

STM324x9I EVAL

Modules

STM324x9I_EVAL BSP User Manual

Data Structure Index

I | L | P | T

I

IO_StateTypeDef

L

LCD_DrawPropTypeDef

P

Point

T

TS_StateTypeI

I | L | P | T

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL IO Exported Types

STM324x9I EVAL IO

Data Structures

struct **IO_StateTypeDef**

Enumerations

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

Enumeration Type Documentation

enum IO_StatusTypeDef

Enumerator:

IO_OK

IO_ERROR

IO_TIMEOUT

Definition at line **75** of file **stm324x9i_eval_io.h**.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_io.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval_io.h
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm324x9i_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 __STM324x9I_EVAL_IO_H
00041 #define __STM324x9I_EVAL_IO_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------------------------------------------*/
00048 #include "stm324x9i_eval.h"
00049 /* Include IO component driver */
00050 #include "../Components/stmpe1600/stmpe1600.h"
00051
00052 /** @addtogroup BSP
00053   * @{
00054   */
00055
00056 /** @addtogroup STM324x9I_EVAL
00057   * @{
00058   */
00059
00060 /** @addtogroup STM324x9I_EVAL_IO
00061   * @{
00062   */
00063
00064 /** @defgroup STM324x9I_EVAL_IO_Exported_Types STM324x9I EVAL IO 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 }IO_StateTypeDef;
00074
00075 typedef enum
00076 {
00077   IO_OK       = 0,
00078   IO_ERROR    = 1,
00079   IO_TIMEOUT  = 2
00080 }IO_StatusTypeDef;
00081 /**
00082   * @}
00083   */
00084
00085 /** @defgroup STM324x9I_EVAL_IO_Exported_Con
stants STM324x9I EVAL IO Exported Constants
00086   * @{
00087   */
00088 #define IO_PIN_0                    0x0001
00089 #define IO_PIN_1                    0x0002
00090 #define IO_PIN_2                    0x0004
00091 #define IO_PIN_3                    0x0008
00092 #define IO_PIN_4                    0x0010
00093 #define IO_PIN_5                    0x0020
00094 #define IO_PIN_6                    0x0040
00095 #define IO_PIN_7                    0x0080
00096 #define IO_PIN_8                    0x0100
00097 #define IO_PIN_9                    0x0200
00098 #define IO_PIN_10                   0x0400
00099 #define IO_PIN_11                   0x0800
00100 #define IO_PIN_12                   0x1000
00101 #define IO_PIN_13                   0x2000
00102 #define IO_PIN_14                   0x4000
00103 #define IO_PIN_15                   0x8000
```

```
00104 #define IO_PIN_ALL                    0xFFFF
00105 /**
00106   * @}
00107   */
00108
00109 /** @defgroup STM324x9I_EVAL_IO_Exported_Mac
ro STM324x9I EVAL IO Exported Macro
00110   * @{
00111   */
00112 /**
00113   * @}
00114   */
00115
00116 /** @defgroup STM324x9I_EVAL_IO_Exported_Fun
ctions STM324x9I EVAL IO Exported Functions
00117   * @{
00118   */
00119 uint8_t  BSP_IO_Init(void);
00120 uint8_t  BSP_IO_ITGetStatus(uint16_t IO_Pin)
;
00121 void     BSP_IO_ITClear(void);
00122 uint8_t  BSP_IO_ConfigPin(uint16_t IO_Pin, I
O_ModeTypedef IO_Mode);
00123 void     BSP_IO_WritePin(uint16_t IO_Pin, ui
nt8_t PinState);
00124 uint16_t BSP_IO_ReadPin(uint16_t IO_Pin);
00125 void     BSP_IO_TogglePin(uint16_t IO_Pin);
00126
00127 /**
00128   * @}
00129   */
00130
00131 /**
00132   * @}
00133   */
00134
00135 /**
```

```
00136   * @}
00137   */
00138
00139 /**
00140   * @}
00141   */
00142
00143 #ifdef __cplusplus
00144 }
00145 #endif
00146
00147 #endif /* __STM324x9I_EVAL_IO_H */
00148
00149 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL LCD

STM324x9I EVAL

Modules

STM324x9I_EVAL BSP User Manual

Data Structures | Typedefs | Enumerations

STM324x9I EVAL LCD Exported Types

STM324x9I EVAL LCD

Data Structures

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

Typedefs

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

Enumerations

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

Typedef Documentation

typedef struct Point * pPoint

Enumeration Type Documentation

enum Text_AlignModeTypdef

Line mode structures definition.

Enumerator:

CENTER_MODE

RIGHT_MODE

LEFT_MODE

Definition at line **94** of file **stm324x9i_eval_lcd.h**.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_lcd.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval_lcd.h
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm324x9i_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 __STM324x9I_EVAL_LCD_H
00041 #define __STM324x9I_EVAL_LCD_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------
-----------------------------------*/
00048 /* Include LCD component Driver */
00049 /* LCD integrated within MB1063 */
00050 #include "../Components/ampire640480/ampire6
40480.h"
00051 /* LCD integrated within MB1046 */
00052 #include "../Components/ampire480272/ampire4
80272.h"
00053
00054 /* Include IOExpander(STMPE811) component Dr
iver */
00055 #include "../Components/stmpe811/stmpe811.h"
00056
00057 /* Include SDRAM Driver */
00058 #include "stm324x9i_eval_sdram.h"
00059
00060 #include "stm324x9i_eval.h"
00061 #include "../../../Utilities/Fonts/fonts.h"
00062
00063 /** @addtogroup BSP
00064   * @{
00065   */
```

```
00066
00067 /** @addtogroup STM324x9I_EVAL
00068   * @{
00069   */
00070
00071 /** @addtogroup STM324x9I_EVAL_LCD
00072   * @{
00073   */
00074
00075 /** @defgroup STM324x9I_EVAL_LCD_Exported_Ty
pes STM324x9I EVAL LCD Exported Types
00076   * @{
00077   */
00078 typedef struct
00079 {
00080   uint32_t TextColor;
00081   uint32_t BackColor;
00082   sFONT    *pFont;
00083 }LCD_DrawPropTypeDef;
00084
00085 typedef struct
00086 {
00087   int16_t X;
00088   int16_t Y;
00089 }Point, * pPoint;
00090
00091 /**
00092   * @brief  Line mode structures definition

00093   */
00094 typedef enum
00095 {
00096   CENTER_MODE             = 0x01,    /* Cent
er mode */
00097   RIGHT_MODE              = 0x02,    /* Righ
t mode  */
00098   LEFT_MODE               = 0x03     /* Left
```

```
 mode   */
00099 }Text_AlignModeTypdef;
00100
00101 /**
00102   * @}
00103   */
00104
00105 /** @defgroup STM324x9I_EVAL_LCD_Exported_Co
nstants STM324x9I EVAL LCD Exported Constants
00106   * @{
00107   */
00108 #define MAX_LAYER_NUMBER       2
00109
00110 #define LCD_LayerCfgTypeDef    LTDC_LayerCfg
TypeDef
00111
00112 /**
00113   * @brief  LCD status structure definition

00114   */
00115 #define   LCD_OK         0x00
00116 #define   LCD_ERROR      0x01
00117 #define   LCD_TIMEOUT    0x02
00118
00119 /**
00120   * @brief  LCD FB_StartAddress
00121   */
00122 #define LCD_FB_START_ADDRESS       ((uint32_
t)0xC0000000)
00123
00124 /* The programmed LTDC pixel clock depends o
n the vertical refresh rate of the panel 60Hz => 2
5.16MHz and
00125    the LCD/SDRAM bandwidth affected by the s
everal access on the bus and the number of used la
yers.
00126    when only one layer is enabled "LCD_MAX_P
```

```
CLK" can be used and when two layers are enabled s
imultaneously
00127    or/and there is several access on the bus
 "LCD_MIN_PCLK" parameter is recommended */
00128 #define LCD_MAX_PCLK       ((uint8_t)0x00)
00129 #define LCD_MIN_PCLK       ((uint8_t)0x01)
00130
00131 /**
00132   * @brief  LCD color
00133   */
00134 #define LCD_COLOR_BLUE          0xFF0000FF
00135 #define LCD_COLOR_GREEN         0xFF00FF00
00136 #define LCD_COLOR_RED           0xFFFF0000
00137 #define LCD_COLOR_CYAN          0xFF00FFFF
00138 #define LCD_COLOR_MAGENTA       0xFFFF00FF
00139 #define LCD_COLOR_YELLOW        0xFFFFFF00
00140 #define LCD_COLOR_LIGHTBLUE     0xFF8080FF
00141 #define LCD_COLOR_LIGHTGREEN    0xFF80FF80
00142 #define LCD_COLOR_LIGHTRED      0xFFFF8080
00143 #define LCD_COLOR_LIGHTCYAN     0xFF80FFFF
00144 #define LCD_COLOR_LIGHTMAGENTA  0xFFFF80FF
00145 #define LCD_COLOR_LIGHTYELLOW   0xFFFFFF80
00146 #define LCD_COLOR_DARKBLUE      0xFF000080
00147 #define LCD_COLOR_DARKGREEN     0xFF008000
00148 #define LCD_COLOR_DARKRED       0xFF800000
00149 #define LCD_COLOR_DARKCYAN      0xFF008080
00150 #define LCD_COLOR_DARKMAGENTA   0xFF800080
00151 #define LCD_COLOR_DARKYELLOW    0xFF808000
00152 #define LCD_COLOR_WHITE         0xFFFFFFFF
00153 #define LCD_COLOR_LIGHTGRAY     0xFFD3D3D3
00154 #define LCD_COLOR_GRAY          0xFF808080
00155 #define LCD_COLOR_DARKGRAY      0xFF404040
00156 #define LCD_COLOR_BLACK         0xFF000000
00157 #define LCD_COLOR_BROWN         0xFFA52A2A
00158 #define LCD_COLOR_ORANGE        0xFFFFA500
00159 #define LCD_COLOR_TRANSPARENT   0xFF000000
00160
```

```
00161 /**
00162   * @brief LCD default font
00163   */
00164 #define LCD_DEFAULT_FONT         Font24
00165 /**
00166   * @}
00167   */
00168
00169 /** @defgroup STM324x9I_EVAL_LCD_Exported_Fu
nctions STM324x9I EVAL LCD Exported Functions
00170   * @{
00171   */
00172 uint8_t  BSP_LCD_Init(void);
00173 uint8_t  BSP_LCD_InitEx(uint32_t PclkConfig)
;
00174
00175 uint32_t BSP_LCD_GetXSize(void);
00176 uint32_t BSP_LCD_GetYSize(void);
00177
00178 /* Functions using the LTDC controller */
00179 void     BSP_LCD_LayerDefaultInit(uint16_t L
ayerIndex, uint32_t FrameBuffer);
00180 void     BSP_LCD_SetTransparency(uint32_t La
yerIndex, uint8_t Transparency);
00181 void     BSP_LCD_SetLayerAddress(uint32_t La
yerIndex, uint32_t Address);
00182 void     BSP_LCD_SetColorKeying(uint32_t Lay
erIndex, uint32_t RGBValue);
00183 void     BSP_LCD_ResetColorKeying(uint32_t L
ayerIndex);
00184 void     BSP_LCD_SetLayerWindow(uint16_t Lay
erIndex, uint16_t Xpos, uint16_t Ypos, uint16_t Wi
dth, uint16_t Height);
00185
00186 void     BSP_LCD_SelectLayer(uint32_t LayerI
ndex);
00187 void     BSP_LCD_SetLayerVisible(uint32_t La
```

```
yerIndex, FunctionalState State);
00188
00189 void     BSP_LCD_SetTextColor(uint32_t Color
);
00190 uint32_t BSP_LCD_GetTextColor(void);
00191 void     BSP_LCD_SetBackColor(uint32_t Color
);
00192 uint32_t BSP_LCD_GetBackColor(void);
00193 void     BSP_LCD_SetFont(sFONT *fonts);
00194 sFONT    *BSP_LCD_GetFont(void);
00195
00196 uint32_t BSP_LCD_ReadPixel(uint16_t Xpos, ui
nt16_t Ypos);
00197 void     BSP_LCD_DrawPixel(uint16_t Xpos, ui
nt16_t Ypos, uint32_t pixel);
00198 void     BSP_LCD_Clear(uint32_t Color);
00199 void     BSP_LCD_ClearStringLine(uint32_t Li
ne);
00200 void     BSP_LCD_DisplayStringAtLine(uint16_
t Line, uint8_t *ptr);
00201 void     BSP_LCD_DisplayStringAt(uint16_t Xp
os, uint16_t Ypos, uint8_t *Text, Text_AlignModeTy
pdef Mode);
00202 void     BSP_LCD_DisplayChar(uint16_t Xpos,
uint16_t Ypos, uint8_t Ascii);
00203
00204 void     BSP_LCD_DrawHLine(uint16_t Xpos, ui
nt16_t Ypos, uint16_t Length);
00205 void     BSP_LCD_DrawVLine(uint16_t Xpos, ui
nt16_t Ypos, uint16_t Length);
00206 void     BSP_LCD_DrawLine(uint16_t x1, uint1
6_t y1, uint16_t x2, uint16_t y2);
00207 void     BSP_LCD_DrawRect(uint16_t Xpos, uin
t16_t Ypos, uint16_t Width, uint16_t Height);
00208 void     BSP_LCD_DrawCircle(uint16_t Xpos, u
int16_t Ypos, uint16_t Radius);
00209 void     BSP_LCD_DrawPolygon(pPoint Points,
```

```
uint16_t PointCount);
00210 void     BSP_LCD_DrawEllipse(int Xpos, int Y
pos, int XRadius, int YRadius);
00211 void     BSP_LCD_DrawBitmap(uint32_t Xpos, u
int32_t Ypos, uint8_t *pbmp);
00212
00213 void     BSP_LCD_FillRect(uint16_t Xpos, uin
t16_t Ypos, uint16_t Width, uint16_t Height);
00214 void     BSP_LCD_FillCircle(uint16_t Xpos, u
int16_t Ypos, uint16_t Radius);
00215 void     BSP_LCD_FillPolygon(pPoint Points,
uint16_t PointCount);
00216 void     BSP_LCD_FillEllipse(int Xpos, int Y
pos, int XRadius, int YRadius);
00217
00218 void     BSP_LCD_DisplayOff(void);
00219 void     BSP_LCD_DisplayOn(void);
00220
00221 void     BSP_LCD_ClockConfig(LTDC_HandleType
Def *hltdc, void *Params);
00222 /**
00223   * @}
00224   */
00225
00226 /**
00227   * @}
00228   */
00229
00230 /**
00231   * @}
00232   */
00233
00234 /**
00235   * @}
00236   */
00237
00238 #ifdef __cplusplus
```

```
00239 }
00240 #endif
00241
00242 #endif /* __STM324x9I_EVAL_LCD_H */
00243
00244 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_lcd.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval_lcd.c
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file includes the driver for Liquid Crystal Display (LCD) module
00008   *          mounted on STM324x9I-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 directly a
n LCD TFT using the LTDC controller.
00044    - This driver selects dynamically the mou
nted LCD, AMPIRE 640x480 LCD mounted
00045      on MB1063 or AMPIRE 480x272 LCD mounted
 on MB1046 daughter board,
00046      and uses the adequate timing and settin
g for the specified LCD using
00047      device ID of the STMPE811 mounted on MB
1046 daughter board.
00048
00049 2. Driver description:
00050 ---------------------
00051   + Initialization steps:
00052      o Initialize the LCD using the BSP_LCD_
Init() function.
00053      o Apply the Layer configuration using t
he BSP_LCD_LayerDefaultInit() function.
00054      o Select the LCD layer to be used using
 the BSP_LCD_SelectLayer() function.
00055      o Enable the LCD display using the BSP_
LCD_DisplayOn() function.
00056
00057   + Options
00058      o Configure and enable the color keying
 functionality using the
```

```
00059         BSP_LCD_SetColorKeying() function.
00060       o Modify in the fly the transparency an
d/or the frame buffer address
00061         using the following functions:
00062         - BSP_LCD_SetTransparency()
00063         - BSP_LCD_SetLayerAddress()
00064
00065    + Display on LCD
00066       o Clear the hole LCD using BSP_LCD_Clea
r() function or only one specified string
00067         line using the BSP_LCD_ClearStringLin
e() function.
00068       o Display a character on the specified
line and column using the BSP_LCD_DisplayChar()
00069         function or a complete string line us
ing the BSP_LCD_DisplayStringAtLine() function.
00070       o Display a string line on the specifie
d position (x,y in pixel) and align mode
00071         using the BSP_LCD_DisplayStringAtLine
() function.
00072       o Draw and fill a basic shapes (dot, li
ne, rectangle, circle, ellipse, .. bitmap)
00073         on LCD using the available set of fun
ctions.
00074
00075 ----------------------------------------
---------------------------------*/
00076
00077 /* Includes ------------------------------
-----------------------------------*/
00078 #include "stm324x9i_eval_lcd.h"
00079 #include "../../../Utilities/Fonts/fonts.h"
00080 #include "../../../Utilities/Fonts/font24.c"
00081 #include "../../../Utilities/Fonts/font20.c"
00082 #include "../../../Utilities/Fonts/font16.c"
00083 #include "../../../Utilities/Fonts/font12.c"
00084 #include "../../../Utilities/Fonts/font8.c"
```

```
00085
00086 /** @addtogroup BSP
00087   * @{
00088   */
00089
00090 /** @addtogroup STM324x9I_EVAL
00091   * @{
00092   */
00093
00094 /** @defgroup STM324x9I_EVAL_LCD STM324x9I E
VAL LCD
00095   * @{
00096   */
00097
00098 /** @defgroup STM324x9I_EVAL_LCD_Private_Typ
esDefinitions STM324x9I EVAL LCD Private TypesDefi
nitions
00099   * @{
00100   */
00101 /**
00102   * @}
00103   */
00104
00105 /** @defgroup STM324x9I_EVAL_LCD_Private_Def
ines STM324x9I EVAL LCD Private Defines
00106   * @{
00107   */
00108 #define POLY_X(Z)              ((int32_t)((P
oints + Z)->X))
00109 #define POLY_Y(Z)              ((int32_t)((P
oints + Z)->Y))
00110 /**
00111   * @}
00112   */
00113
00114 /** @defgroup STM324x9I_EVAL_LCD_Private_Mac
ros STM324x9I EVAL LCD Private Macros
```

```
00115   * @{
00116   */
00117 #define ABS(X)  ((X) > 0 ? (X) : -(X))
00118 /**
00119   * @}
00120   */
00121
00122 /** @defgroup STM324x9I_EVAL_LCD_Private_Var
iables STM324x9I EVAL LCD Private Variables
00123   * @{
00124   */
00125 static LTDC_HandleTypeDef  hltdc_eval;
00126 static DMA2D_HandleTypeDef hdma2d_eval;
00127 static uint32_t            PCLK_profile = LC
D_MAX_PCLK;
00128
00129 /* Default LCD configuration with LCD Layer
1 */
00130 static uint32_t            ActiveLayer = 0;
00131 static LCD_DrawPropTypeDef DrawProp[MAX_LAYE
R_NUMBER];
00132 /**
00133   * @}
00134   */
00135
00136 /** @defgroup STM324x9I_EVAL_LCD_Private_Fun
ctionPrototypes STM324x9I EVAL LCD Private Functio
nPrototypes
00137   * @{
00138   */
00139 static void MspInit(void);
00140 static void DrawChar(uint16_t Xpos, uint16_t
 Ypos, const uint8_t *c);
00141 static void FillTriangle(uint16_t x1, uint16
_t x2, uint16_t x3, uint16_t y1, uint16_t y2, uint
16_t y3);
00142 static void LL_FillBuffer(uint32_t LayerInde
```

```
x, void *pDst, uint32_t xSize, uint32_t ySize, uin
t32_t OffLine, uint32_t ColorIndex);
00143 static void LL_ConvertLineToARGB8888(void *
pSrc, void *pDst, uint32_t xSize, uint32_t ColorMo
de);
00144 /**
00145   * @}
00146   */
00147
00148 /** @defgroup STM324x9I_EVAL_LCD_Private_Fun
ctions STM324x9I EVAL LCD Private Functions
00149   * @{
00150   */
00151 /**
00152   * @brief  Initializes the LCD.
00153   * @retval LCD state
00154   */
00155 uint8_t BSP_LCD_Init(void)
00156 {
00157   return (BSP_LCD_InitEx(LCD_MAX_PCLK));
00158 }
00159
00160 /**
00161   * @brief  Initializes the LCD.
00162   * @param  PclkConfig : pixel clock profile
00163   * @retval LCD state
00164   */
00165 uint8_t BSP_LCD_InitEx(uint32_t PclkConfig)
00166 {
00167   PCLK_profile = PclkConfig;
00168
00169   /* Select the used LCD */
00170   /* The AMPIRE 480x272 does not contain an
ID register then we check the availability
00171      of AMPIRE 480x640 LCD using device ID o
f the STMPE811 mounted on MB1046 daughter board */
```

```
00172   if(stmpe811_ts_drv.ReadID(TS_I2C_ADDRESS)
== STMPE811_ID)
00173   {
00174     /* The AMPIRE LCD 480x272 is selected */
00175     /* Timing Configuration */
00176     hltdc_eval.Init.HorizontalSync = (AMPIRE
480272_HSYNC - 1);
00177     hltdc_eval.Init.VerticalSync = (AMPIRE48
0272_VSYNC - 1);
00178     hltdc_eval.Init.AccumulatedHBP = (AMPIRE
480272_HSYNC + AMPIRE480272_HBP - 1);
00179     hltdc_eval.Init.AccumulatedVBP = (AMPIRE
480272_VSYNC + AMPIRE480272_VBP - 1);
00180     hltdc_eval.Init.AccumulatedActiveH = (AM
PIRE480272_HEIGHT + AMPIRE480272_VSYNC + AMPIRE480
272_VBP - 1);
00181     hltdc_eval.Init.AccumulatedActiveW = (AM
PIRE480272_WIDTH + AMPIRE480272_HSYNC + AMPIRE4802
72_HBP - 1);
00182     hltdc_eval.Init.TotalHeigh = (AMPIRE4802
72_HEIGHT + AMPIRE480272_VSYNC + AMPIRE480272_VBP
+ AMPIRE480272_VFP - 1);
00183     hltdc_eval.Init.TotalWidth = (AMPIRE4802
72_WIDTH + AMPIRE480272_HSYNC + AMPIRE480272_HBP +
 AMPIRE480272_HFP - 1);
00184
00185     /* Initialize the LCD pixel width and pi
xel height */
00186     hltdc_eval.LayerCfg->ImageWidth  = AMPIR
E480272_WIDTH;
00187     hltdc_eval.LayerCfg->ImageHeight = AMPIR
E480272_HEIGHT;
00188   }
00189   else
00190   {
00191     /* The LCD AMPIRE 640x480 is selected */
00192     /* Timing configuration */
```

```
00193     hltdc_eval.Init.HorizontalSync = (AMPIRE
640480_HSYNC - 1);
00194     hltdc_eval.Init.VerticalSync = (AMPIRE64
0480_VSYNC - 1);
00195     hltdc_eval.Init.AccumulatedHBP = (AMPIRE
640480_HSYNC + AMPIRE640480_HBP - 1);
00196     hltdc_eval.Init.AccumulatedVBP = (AMPIRE
640480_VSYNC + AMPIRE640480_VBP - 1);
00197     hltdc_eval.Init.AccumulatedActiveH = (AM
PIRE640480_HEIGHT + AMPIRE640480_VSYNC + AMPIRE640
480_VBP - 1);
00198     hltdc_eval.Init.AccumulatedActiveW = (AM
PIRE640480_WIDTH + AMPIRE640480_HSYNC + AMPIRE6404
80_HBP - 1);
00199     hltdc_eval.Init.TotalHeigh = (AMPIRE6404
80_HEIGHT + AMPIRE640480_VSYNC + AMPIRE640480_VBP
+ AMPIRE640480_VFP - 1);
00200     hltdc_eval.Init.TotalWidth = (AMPIRE6404
80_WIDTH + AMPIRE640480_HSYNC + AMPIRE640480_HBP +
 AMPIRE640480_HFP - 1);
00201
00202     /* Initialize the LCD pixel width and pi
xel height */
00203     hltdc_eval.LayerCfg->ImageWidth  = AMPIR
E640480_WIDTH;
00204     hltdc_eval.LayerCfg->ImageHeight = AMPIR
E640480_HEIGHT;
00205    }
00206
00207   /* Background value */
00208   hltdc_eval.Init.Backcolor.Blue = 0;
00209   hltdc_eval.Init.Backcolor.Green = 0;
00210   hltdc_eval.Init.Backcolor.Red = 0;
00211
00212   /* Polarity */
00213   hltdc_eval.Init.HSPolarity = LTDC_HSPOLARI
TY_AL;
```

```
00214   hltdc_eval.Init.VSPolarity = LTDC_VSPOLARI
TY_AL;
00215   hltdc_eval.Init.DEPolarity = LTDC_DEPOLARI
TY_AL;
00216   hltdc_eval.Init.PCPolarity = LTDC_PCPOLARI
TY_IPC;
00217   hltdc_eval.Instance = LTDC;
00218
00219   /* LCD clock configuration */
00220   BSP_LCD_ClockConfig(&hltdc_eval, &PCLK_pro
file);
00221
00222   MspInit();
00223   HAL_LTDC_Init(&hltdc_eval);
00224
00225 #if !defined(DATA_IN_ExtSDRAM)
00226   /* Initialize the SDRAM */
00227   BSP_SDRAM_Init();
00228 #endif /* DATA_IN_ExtSDRAM */
00229
00230   /* Initialize the font */
00231   BSP_LCD_SetFont(&LCD_DEFAULT_FONT);
00232
00233   return LCD_OK;
00234 }
00235
00236 /**
00237   * @brief  Gets the LCD X size.
00238   * @retval Used LCD X size
00239   */
00240 uint32_t BSP_LCD_GetXSize(void)
00241 {
00242   return hltdc_eval.LayerCfg[ActiveLayer].Im
ageWidth;
00243 }
00244
00245 /**
```

```
00246   * @brief  Gets the LCD Y size.
00247   * @retval Used LCD Y size
00248   */
00249 uint32_t BSP_LCD_GetYSize(void)
00250 {
00251   return hltdc_eval.LayerCfg[ActiveLayer].Im
ageHeight;
00252 }
00253
00254 /**
00255   * @brief  Initializes the LCD layers.
00256   * @param  LayerIndex: Layer foreground or
background
00257   * @param  FB_Address: Layer frame buffer
00258   */
00259 void BSP_LCD_LayerDefaultInit(uint16_t Layer
Index, uint32_t FB_Address)
00260 {
00261   LCD_LayerCfgTypeDef  Layercfg;
00262
00263   /* Layer Init */
00264   Layercfg.WindowX0 = 0;
00265   Layercfg.WindowX1 = BSP_LCD_GetXSize();
00266   Layercfg.WindowY0 = 0;
00267   Layercfg.WindowY1 = BSP_LCD_GetYSize();
00268   Layercfg.PixelFormat = LTDC_PIXEL_FORMAT_A
RGB8888;
00269   Layercfg.FBStartAdress = FB_Address;
00270   Layercfg.Alpha = 255;
00271   Layercfg.Alpha0 = 0;
00272   Layercfg.Backcolor.Blue = 0;
00273   Layercfg.Backcolor.Green = 0;
00274   Layercfg.Backcolor.Red = 0;
00275   Layercfg.BlendingFactor1 = LTDC_BLENDING_F
ACTOR1_PAxCA;
00276   Layercfg.BlendingFactor2 = LTDC_BLENDING_F
ACTOR2_PAxCA;
```

```
00277   Layercfg.ImageWidth = BSP_LCD_GetXSize();
00278   Layercfg.ImageHeight = BSP_LCD_GetYSize();
00279
00280   HAL_LTDC_ConfigLayer(&hltdc_eval, &Layercf
g, LayerIndex);
00281
00282   DrawProp[LayerIndex].BackColor = LCD_COLOR
_WHITE;
00283   DrawProp[LayerIndex].pFont     = &Font24;
00284   DrawProp[LayerIndex].TextColor = LCD_COLOR
_BLACK;
00285 }
00286
00287 /**
00288   * @brief  Selects the LCD Layer.
00289   * @param  LayerIndex: Layer foreground or
background
00290   */
00291 void BSP_LCD_SelectLayer(uint32_t LayerIndex
)
00292 {
00293   ActiveLayer = LayerIndex;
00294 }
00295
00296 /**
00297   * @brief  Sets an LCD Layer visible
00298   * @param  LayerIndex: Visible Layer
00299   * @param  State: New state of the specifie
d layer
00300   *          This parameter can be one of th
e following values:
00301   *            @arg  ENABLE
00302   *            @arg  DISABLE
00303   */
00304 void BSP_LCD_SetLayerVisible(uint32_t LayerI
ndex, FunctionalState State)
00305 {
```

```
00306   if(State == ENABLE)
00307   {
00308     __HAL_LTDC_LAYER_ENABLE(&hltdc_eval, Lay
erIndex);
00309   }
00310   else
00311   {
00312     __HAL_LTDC_LAYER_DISABLE(&hltdc_eval, La
yerIndex);
00313   }
00314   __HAL_LTDC_RELOAD_CONFIG(&hltdc_eval);
00315 }
00316
00317 /**
00318   * @brief  Configures the transparency.
00319   * @param  LayerIndex: Layer foreground or
background.
00320   * @param  Transparency: Transparency
00321   *           This parameter must be a numbe
r between Min_Data = 0x00 and Max_Data = 0xFF
00322   */
00323 void BSP_LCD_SetTransparency(uint32_t LayerI
ndex, uint8_t Transparency)
00324 {
00325   HAL_LTDC_SetAlpha(&hltdc_eval, Transparenc
y, LayerIndex);
00326 }
00327
00328 /**
00329   * @brief  Sets an LCD layer frame buffer a
ddress.
00330   * @param  LayerIndex: Layer foreground or
background
00331   * @param  Address: New LCD frame buffer va
lue
00332   */
00333 void BSP_LCD_SetLayerAddress(uint32_t LayerI
```

```
ndex, uint32_t Address)
00334 {
00335   HAL_LTDC_SetAddress(&hltdc_eval, Address,
LayerIndex);
00336 }
00337
00338 /**
00339   * @brief  Sets display window.
00340   * @param  LayerIndex: Layer index
00341   * @param  Xpos: LCD X position
00342   * @param  Ypos: LCD Y position
00343   * @param  Width: LCD window width
00344   * @param  Height: LCD window height
00345   */
00346 void BSP_LCD_SetLayerWindow(uint16_t LayerIn
dex, uint16_t Xpos, uint16_t Ypos, uint16_t Width,
 uint16_t Height)
00347 {
00348   /* Reconfigure the layer size */
00349   HAL_LTDC_SetWindowSize(&hltdc_eval, Width,
 Height, LayerIndex);
00350
00351   /* Reconfigure the layer position */
00352   HAL_LTDC_SetWindowPosition(&hltdc_eval, Xp
os, Ypos, LayerIndex);
00353 }
00354
00355 /**
00356   * @brief  Configures and sets the color ke
ying.
00357   * @param  LayerIndex: Layer foreground or
background
00358   * @param  RGBValue: Color reference
00359   */
00360 void BSP_LCD_SetColorKeying(uint32_t LayerIn
dex, uint32_t RGBValue)
00361 {
```

```
00362   /* Configure and Enable the color Keying f
or LCD Layer */
00363   HAL_LTDC_ConfigColorKeying(&hltdc_eval, RG
BValue, LayerIndex);
00364   HAL_LTDC_EnableColorKeying(&hltdc_eval, La
yerIndex);
00365 }
00366
00367 /**
00368   * @brief  Disables the color keying.
00369   * @param  LayerIndex: Layer foreground or
background
00370   */
00371 void BSP_LCD_ResetColorKeying(uint32_t Layer
Index)
00372 {
00373   /* Disable the color Keying for LCD Layer
*/
00374   HAL_LTDC_DisableColorKeying(&hltdc_eval, L
ayerIndex);
00375 }
00376
00377 /**
00378   * @brief  Sets the LCD text color.
00379   * @param  Color: Text color code ARGB(8-8-
8-8)
00380   */
00381 void BSP_LCD_SetTextColor(uint32_t Color)
00382 {
00383   DrawProp[ActiveLayer].TextColor = Color;
00384 }
00385
00386 /**
00387   * @brief  Gets the LCD text color.
00388   * @retval Used text color.
00389   */
00390 uint32_t BSP_LCD_GetTextColor(void)
```

```
00391 {
00392   return DrawProp[ActiveLayer].TextColor;
00393 }
00394
00395 /**
00396   * @brief  Sets the LCD background color.
00397   * @param  Color: Layer background color co
de ARGB(8-8-8-8)
00398   */
00399 void BSP_LCD_SetBackColor(uint32_t Color)
00400 {
00401   DrawProp[ActiveLayer].BackColor = Color;
00402 }
00403
00404 /**
00405   * @brief  Gets the LCD background color.
00406   * @retval Used background color
00407   */
00408 uint32_t BSP_LCD_GetBackColor(void)
00409 {
00410   return DrawProp[ActiveLayer].BackColor;
00411 }
00412
00413 /**
00414   * @brief  Sets the LCD text font.
00415   * @param  fonts: Layer font to be used
00416   */
00417 void BSP_LCD_SetFont(sFONT *fonts)
00418 {
00419   DrawProp[ActiveLayer].pFont = fonts;
00420 }
00421
00422 /**
00423   * @brief  Gets the LCD text font.
00424   * @retval Used layer font
00425   */
00426 sFONT *BSP_LCD_GetFont(void)
```

```
00427 {
00428   return DrawProp[ActiveLayer].pFont;
00429 }
00430
00431 /**
00432   * @brief  Reads an LCD pixel.
00433   * @param  Xpos: X position
00434   * @param  Ypos: Y position
00435   * @retval RGB pixel color
00436   */
00437 uint32_t BSP_LCD_ReadPixel(uint16_t Xpos, ui
nt16_t Ypos)
00438 {
00439   uint32_t ret = 0;
00440
00441   if(hltdc_eval.LayerCfg[ActiveLayer].PixelF
ormat == LTDC_PIXEL_FORMAT_ARGB8888)
00442   {
00443     /* Read data value from SDRAM memory */
00444     ret = *(__IO uint32_t*) (hltdc_eval.Laye
rCfg[ActiveLayer].FBStartAdress + (2*(Ypos*BSP_LCD
_GetXSize() + Xpos)));
00445   }
00446   else if(hltdc_eval.LayerCfg[ActiveLayer].P
ixelFormat == LTDC_PIXEL_FORMAT_RGB888)
00447   {
00448     /* Read data value from SDRAM memory */
00449     ret = (*(__IO uint32_t*) (hltdc_eval.Lay
erCfg[ActiveLayer].FBStartAdress + (2*(Ypos*BSP_LC
D_GetXSize() + Xpos))) & 0x00FFFFFF);
00450   }
00451   else if((hltdc_eval.LayerCfg[ActiveLayer].
PixelFormat == LTDC_PIXEL_FORMAT_RGB565) || \
00452           (hltdc_eval.LayerCfg[ActiveLayer].
PixelFormat == LTDC_PIXEL_FORMAT_ARGB4444) || \
00453           (hltdc_eval.LayerCfg[ActiveLayer].
PixelFormat == LTDC_PIXEL_FORMAT_AL88))
```

```
00454   {
00455     /* Read data value from SDRAM memory */
00456     ret = *(__IO uint16_t*) (hltdc_eval.Laye
rCfg[ActiveLayer].FBStartAdress + (2*(Ypos*BSP_LCD
_GetXSize() + Xpos)));
00457   }
00458   else
00459   {
00460     /* Read data value from SDRAM memory */
00461     ret = *(__IO uint8_t*) (hltdc_eval.Layer
Cfg[ActiveLayer].FBStartAdress + (2*(Ypos*BSP_LCD_
GetXSize() + Xpos)));
00462   }
00463
00464   return ret;
00465 }
00466
00467 /**
00468   * @brief  Clears the hole LCD.
00469   * @param  Color: Color of the background
00470   */
00471 void BSP_LCD_Clear(uint32_t Color)
00472 {
00473   /* Clear the LCD */
00474   LL_FillBuffer(ActiveLayer, (uint32_t *)(hl
tdc_eval.LayerCfg[ActiveLayer].FBStartAdress), BSP
_LCD_GetXSize(), BSP_LCD_GetYSize(), 0, Color);
00475 }
00476
00477 /**
00478   * @brief  Clears the selected line.
00479   * @param  Line: Line to be cleared
00480   */
00481 void BSP_LCD_ClearStringLine(uint32_t Line)
00482 {
00483   uint32_t color_backup = DrawProp[ActiveLay
er].TextColor;
```

```
00484   DrawProp[ActiveLayer].TextColor = DrawProp[
ActiveLayer].BackColor;
00485
00486   /* Draw rectangle with background color */
00487   BSP_LCD_FillRect(0, (Line * DrawProp[Activ
eLayer].pFont->Height), BSP_LCD_GetXSize(), DrawPr
op[ActiveLayer].pFont->Height);
00488
00489   DrawProp[ActiveLayer].TextColor = color_ba
ckup;
00490   BSP_LCD_SetTextColor(DrawProp[ActiveLayer]
.TextColor);
00491 }
00492
00493 /**
00494   * @brief  Displays one character.
00495   * @param  Xpos: Start column address
00496   * @param  Ypos: Line where to display the
character shape.
00497   * @param  Ascii: Character ascii code
00498   *           This parameter must be a numbe
r between Min_Data = 0x20 and Max_Data = 0x7E
00499   */
00500 void BSP_LCD_DisplayChar(uint16_t Xpos, uint
16_t Ypos, uint8_t Ascii)
00501 {
00502   DrawChar(Xpos, Ypos, &DrawProp[ActiveLayer
].pFont->table[(Ascii-' ') *\
00503     DrawProp[ActiveLayer].pFont->Height * ((
DrawProp[ActiveLayer].pFont->Width + 7) / 8)]);
00504 }
00505
00506 /**
00507   * @brief  Displays characters on the LCD.
00508   * @param  Xpos: X position (in pixel)
00509   * @param  Ypos: Y position (in pixel)
00510   * @param  Text: Pointer to string to displ
```

```
ay on LCD
00511   * @param  Mode: Display mode
00512   *           This parameter can be one of th
e following values:
00513   *             @arg  CENTER_MODE
00514   *             @arg  RIGHT_MODE
00515   *             @arg  LEFT_MODE
00516   */
00517 void BSP_LCD_DisplayStringAt(uint16_t Xpos,
uint16_t Ypos, uint8_t *Text, Text_AlignModeTypdef
 Mode)
00518 {
00519   uint16_t refcolumn = 1, i = 0;
00520   uint32_t size = 0, xsize = 0;
00521   uint8_t  *ptr = Text;
00522
00523   /* Get the text size */
00524   while (*ptr++) size ++ ;
00525
00526   /* Characters number per line */
00527   xsize = (BSP_LCD_GetXSize()/DrawProp[Activ
eLayer].pFont->Width);
00528
00529   switch (Mode)
00530   {
00531   case CENTER_MODE:
00532     {
00533       refcolumn = Xpos + ((xsize - size)* Dr
awProp[ActiveLayer].pFont->Width) / 2;
00534       break;
00535     }
00536   case LEFT_MODE:
00537     {
00538       refcolumn = Xpos;
00539       break;
00540     }
00541   case RIGHT_MODE:
```

```
00542     {
00543       refcolumn = - Xpos + ((xsize - size)*D
rawProp[ActiveLayer].pFont->Width);
00544       break;
00545     }
00546   default:
00547     {
00548       refcolumn = Xpos;
00549       break;
00550     }
00551   }
00552
00553   /* Send the string character by character
on LCD */
00554   while ((*Text != 0) & (((BSP_LCD_GetXSize(
) - (i*DrawProp[ActiveLayer].pFont->Width)) & 0xFF
FF) >= DrawProp[ActiveLayer].pFont->Width))
00555   {
00556     /* Display one character on LCD */
00557     BSP_LCD_DisplayChar(refcolumn, Ypos, *Te
xt);
00558     /* Decrement the column position by 16 */

00559     refcolumn += DrawProp[ActiveLayer].pFont
->Width;
00560     /* Point on the next character */
00561     Text++;
00562     i++;
00563   }
00564 }
00565
00566 /**
00567   * @brief  Displays a maximum of 60 charact
ers on the LCD.
00568   * @param  Line: Line where to display the
character shape
00569   * @param  ptr: Pointer to string to displa
```

```
y on LCD
00570   */
00571 void BSP_LCD_DisplayStringAtLine(uint16_t Li
ne, uint8_t *ptr)
00572 {
00573   BSP_LCD_DisplayStringAt(0, LINE(Line), ptr
, LEFT_MODE);
00574 }
00575
00576 /**
00577   * @brief  Draws an horizontal line.
00578   * @param  Xpos: X position
00579   * @param  Ypos: Y position
00580   * @param  Length: Line length
00581   */
00582 void BSP_LCD_DrawHLine(uint16_t Xpos, uint16
_t Ypos, uint16_t Length)
00583 {
00584   uint32_t  Xaddress = 0;
00585
00586   /* Get the line address */
00587   Xaddress = (hltdc_eval.LayerCfg[ActiveLayer
].FBStartAdress) + 4*(BSP_LCD_GetXSize()*Ypos + Xp
os);
00588
00589   /* Write line */
00590   LL_FillBuffer(ActiveLayer, (uint32_t *)Xad
dress, Length, 1, 0, DrawProp[ActiveLayer].TextCol
or);
00591 }
00592
00593 /**
00594   * @brief  Draws a vertical line.
00595   * @param  Xpos: X position
00596   * @param  Ypos: Y position
00597   * @param  Length: Line length
00598   */
```

```
00599 void BSP_LCD_DrawVLine(uint16_t Xpos, uint16
_t Ypos, uint16_t Length)
00600 {
00601   uint32_t  Xaddress = 0;
00602
00603   /* Get the line address */
00604   Xaddress = (hltdc_eval.LayerCfg[ActiveLayer
].FBStartAdress) + 4*(BSP_LCD_GetXSize()*Ypos + Xp
os);
00605
00606   /* Write line */
00607   LL_FillBuffer(ActiveLayer, (uint32_t *)Xad
dress, 1, Length, (BSP_LCD_GetXSize() - 1), DrawPr
op[ActiveLayer].TextColor);
00608 }
00609
00610 /**
00611   * @brief  Draws an uni-line (between two p
oints).
00612   * @param  x1: Point 1 X position
00613   * @param  y1: Point 1 Y position
00614   * @param  x2: Point 2 X position
00615   * @param  y2: Point 2 Y position
00616   */
00617 void BSP_LCD_DrawLine(uint16_t x1, uint16_t
y1, uint16_t x2, uint16_t y2)
00618 {
00619   int16_t deltax = 0, deltay = 0, x = 0, y =
 0, xinc1 = 0, xinc2 = 0,
00620   yinc1 = 0, yinc2 = 0, den = 0, num = 0, nu
madd = 0, numpixels = 0,
00621   curpixel = 0;
00622
00623   deltax = ABS(x2 - x1);        /* The diffe
rence between the x's */
00624   deltay = ABS(y2 - y1);        /* The diffe
rence between the y's */
```

```
00625   x = x1;                         /* Start x o
ff at the first pixel */
00626   y = y1;                         /* Start y o
ff at the first pixel */
00627
00628   if (x2 >= x1)                   /* The x-val
ues are increasing */
00629   {
00630     xinc1 = 1;
00631     xinc2 = 1;
00632   }
00633   else                            /* The x-val
ues are decreasing */
00634   {
00635     xinc1 = -1;
00636     xinc2 = -1;
00637   }
00638
00639   if (y2 >= y1)                   /* The y-val
ues are increasing */
00640   {
00641     yinc1 = 1;
00642     yinc2 = 1;
00643   }
00644   else                            /* The y-val
ues are decreasing */
00645   {
00646     yinc1 = -1;
00647     yinc2 = -1;
00648   }
00649
00650   if (deltax >= deltay)           /* There is
at least one x-value for every y-value */
00651   {
00652     xinc1 = 0;                    /* Don't cha
nge the x when numerator >= denominator */
00653     yinc2 = 0;                    /* Don't cha
```

```
nge the y for every iteration */
00654     den = deltax;
00655     num = deltax / 2;
00656     numadd = deltay;
00657     numpixels = deltax;         /* There are
 more x-values than y-values */
00658   }
00659   else                          /* There is
at least one y-value for every x-value */
00660   {
00661     xinc2 = 0;                  /* Don't cha
nge the x for every iteration */
00662     yinc1 = 0;                  /* Don't cha
nge the y when numerator >= denominator */
00663     den = deltay;
00664     num = deltay / 2;
00665     numadd = deltax;
00666     numpixels = deltay;         /* There are
 more y-values than x-values */
00667   }
00668
00669   for (curpixel = 0; curpixel <= numpixels;
curpixel++)
00670   {
00671     BSP_LCD_DrawPixel(x, y, DrawProp[ActiveL
ayer].TextColor);   /* Draw the current pixel */
00672     num += numadd;
  /* Increase the numerator by the top of the frac
tion */
00673     if (num >= den)
  /* Check if numerator >= denominator */
00674     {
00675       num -= den;
  /* Calculate the new numerator value */
00676       x += xinc1;
  /* Change the x as appropriate */
00677       y += yinc1;
```

```
  /* Change the y as appropriate */
00678     }
00679     x += xinc2;
  /* Change the x as appropriate */
00680     y += yinc2;
  /* Change the y as appropriate */
00681   }
00682 }
00683
00684 /**
00685   * @brief  Draws a rectangle.
00686   * @param  Xpos: X position
00687   * @param  Ypos: Y position
00688   * @param  Width: Rectangle width
00689   * @param  Height: Rectangle height
00690   */
00691 void BSP_LCD_DrawRect(uint16_t Xpos, uint16_
t Ypos, uint16_t Width, uint16_t Height)
00692 {
00693   /* Draw horizontal lines */
00694   BSP_LCD_DrawHLine(Xpos, Ypos, Width);
00695   BSP_LCD_DrawHLine(Xpos, (Ypos+ Height), Wi
dth);
00696
00697   /* Draw vertical lines */
00698   BSP_LCD_DrawVLine(Xpos, Ypos, Height);
00699   BSP_LCD_DrawVLine((Xpos + Width), Ypos, He
ight);
00700 }
00701
00702 /**
00703   * @brief  Draws a circle.
00704   * @param  Xpos: X position
00705   * @param  Ypos: Y position
00706   * @param  Radius: Circle radius
00707   */
00708 void BSP_LCD_DrawCircle(uint16_t Xpos, uint1
```

```
6_t Ypos, uint16_t Radius)
00709 {
00710   int32_t   D;    /* Decision Variable */
00711   uint32_t  CurX; /* Current X Value */
00712   uint32_t  CurY; /* Current Y Value */
00713
00714   D = 3 - (Radius << 1);
00715   CurX = 0;
00716   CurY = Radius;
00717
00718   while (CurX <= CurY)
00719   {
00720     BSP_LCD_DrawPixel((Xpos + CurX), (Ypos -
 CurY), DrawProp[ActiveLayer].TextColor);
00721
00722     BSP_LCD_DrawPixel((Xpos - CurX), (Ypos -
 CurY), DrawProp[ActiveLayer].TextColor);
00723
00724     BSP_LCD_DrawPixel((Xpos + CurY), (Ypos -
 CurX), DrawProp[ActiveLayer].TextColor);
00725
00726     BSP_LCD_DrawPixel((Xpos - CurY), (Ypos -
 CurX), DrawProp[ActiveLayer].TextColor);
00727
00728     BSP_LCD_DrawPixel((Xpos + CurX), (Ypos +
 CurY), DrawProp[ActiveLayer].TextColor);
00729
00730     BSP_LCD_DrawPixel((Xpos - CurX), (Ypos +
 CurY), DrawProp[ActiveLayer].TextColor);
00731
00732     BSP_LCD_DrawPixel((Xpos + CurY), (Ypos +
 CurX), DrawProp[ActiveLayer].TextColor);
00733
00734     BSP_LCD_DrawPixel((Xpos - CurY), (Ypos +
 CurX), DrawProp[ActiveLayer].TextColor);
00735
00736     if (D < 0)
```

```
00737     {
00738       D += (CurX << 2) + 6;
00739     }
00740     else
00741     {
00742       D += ((CurX - CurY) << 2) + 10;
00743       CurY--;
00744     }
00745     CurX++;
00746   }
00747 }
00748
00749 /**
00750   * @brief  Draws an poly-line (between many
 points).
00751   * @param  Points: Pointer to the points ar
ray
00752   * @param  PointCount: Number of points
00753   */
00754 void BSP_LCD_DrawPolygon(pPoint Points, uint
16_t PointCount)
00755 {
00756   int16_t X = 0, Y = 0;
00757
00758   if(PointCount < 2)
00759   {
00760     return;
00761   }
00762
00763   BSP_LCD_DrawLine(Points->X, Points->Y, (Po
ints+PointCount-1)->X, (Points+PointCount-1)->Y);
00764
00765   while(--PointCount)
00766   {
00767     X = Points->X;
00768     Y = Points->Y;
00769     Points++;
```

```
00770     BSP_LCD_DrawLine(X, Y, Points->X, Points
->Y);
00771   }
00772 }
00773
00774 /**
00775   * @brief  Draws an ellipse on LCD.
00776   * @param  Xpos: X position
00777   * @param  Ypos: Y position
00778   * @param  XRadius: Ellipse X radius
00779   * @param  YRadius: Ellipse Y radius
00780   */
00781 void BSP_LCD_DrawEllipse(int Xpos, int Ypos,
int XRadius, int YRadius)
00782 {
00783   int x = 0, y = -YRadius, err = 2-2*XRadius
, e2;
00784   float K = 0, rad1 = 0, rad2 = 0;
00785
00786   rad1 = XRadius;
00787   rad2 = YRadius;
00788
00789   K = (float)(rad2/rad1);
00790
00791   do {
00792     BSP_LCD_DrawPixel((Xpos-(uint16_t)(x/K))
, (Ypos+y), DrawProp[ActiveLayer].TextColor);
00793     BSP_LCD_DrawPixel((Xpos+(uint16_t)(x/K))
, (Ypos+y), DrawProp[ActiveLayer].TextColor);
00794     BSP_LCD_DrawPixel((Xpos+(uint16_t)(x/K))
, (Ypos-y), DrawProp[ActiveLayer].TextColor);
00795     BSP_LCD_DrawPixel((Xpos-(uint16_t)(x/K))
, (Ypos-y), DrawProp[ActiveLayer].TextColor);

00796
00797     e2 = err;
00798     if (e2 <= x) {
```

```
00799       err += ++x*2+1;
00800       if (-y == x && e2 <= y) e2 = 0;
00801     }
00802     if (e2 > y) err += ++y*2+1;
00803   }
00804   while (y <= 0);
00805 }
00806
00807 /**
00808   * @brief  Draws a bitmap picture loaded in
 the internal Flash (32 bpp).
00809   * @param  Xpos: Bmp X position in the LCD
00810   * @param  Ypos: Bmp Y position in the LCD
00811   * @param  pbmp: Pointer to Bmp picture add
ress in the internal Flash
00812   */
00813 void BSP_LCD_DrawBitmap(uint32_t Xpos, uint3
2_t Ypos, uint8_t *pbmp)
00814 {
00815   uint32_t index = 0, width = 0, height = 0,
 bit_pixel = 0;
00816   uint32_t Address;
00817   uint32_t InputColorMode = 0;
00818
00819   /* Get bitmap data address offset */
00820   index = *(__IO uint16_t *) (pbmp + 10);
00821   index |= (*(__IO uint16_t *) (pbmp + 12))
<< 16;
00822
00823   /* Read bitmap width */
00824   width = *(uint16_t *) (pbmp + 18);
00825   width |= (*(uint16_t *) (pbmp + 20)) << 16
;
00826
00827   /* Read bitmap height */
00828   height = *(uint16_t *) (pbmp + 22);
00829   height |= (*(uint16_t *) (pbmp + 24)) << 1
```

```
6;
00830
00831   /* Read bit/pixel */
00832   bit_pixel = *(uint16_t *) (pbmp + 28);
00833
00834   /* Set the address */
00835   Address = hltdc_eval.LayerCfg[ActiveLayer]
.FBStartAdress + (((BSP_LCD_GetXSize()*Ypos) + Xpo
s)*(4));
00836
00837   /* Get the layer pixel format */
00838   if ((bit_pixel/8) == 4)
00839   {
00840     InputColorMode = CM_ARGB8888;
00841   }
00842   else if ((bit_pixel/8) == 2)
00843   {
00844     InputColorMode = CM_RGB565;
00845   }
00846   else
00847   {
00848     InputColorMode = CM_RGB888;
00849   }
00850
00851   /* Bypass the bitmap header */
00852   pbmp += (index + (width * (height - 1) * (
bit_pixel/8)));
00853
00854   /* Convert picture to ARGB8888 pixel forma
t */
00855   for(index=0; index < height; index++)
00856   {
00857     /* Pixel format conversion */
00858     LL_ConvertLineToARGB8888((uint32_t *)pbm
p, (uint32_t *)Address, width, InputColorMode);
00859
00860     /* Increment the source and destination
```

```
buffers */
00861     Address+=  (BSP_LCD_GetXSize()*4);
00862     pbmp -= width*(bit_pixel/8);
00863   }
00864 }
00865
00866 /**
00867   * @brief  Draws a full rectangle.
00868   * @param  Xpos: X position
00869   * @param  Ypos: Y position
00870   * @param  Width: Rectangle width
00871   * @param  Height: Rectangle height
00872   */
00873 void BSP_LCD_FillRect(uint16_t Xpos, uint16_
t Ypos, uint16_t Width, uint16_t Height)
00874 {
00875   uint32_t  Xaddress = 0;
00876
00877   /* Set the text color */
00878   BSP_LCD_SetTextColor(DrawProp[ActiveLayer]
.TextColor);
00879
00880   /* Get the rectangle start address */
00881   Xaddress = (hltdc_eval.LayerCfg[ActiveLayer
].FBStartAdress) + 4*(BSP_LCD_GetXSize()*Ypos + Xp
os);
00882
00883   /* Fill the rectangle */
00884   LL_FillBuffer(ActiveLayer, (uint32_t *)Xad
dress, Width, Height, (BSP_LCD_GetXSize() - Width)
, DrawProp[ActiveLayer].TextColor);
00885 }
00886
00887 /**
00888   * @brief  Draws a full circle.
00889   * @param  Xpos: X position
00890   * @param  Ypos: Y position
```

```
00891   * @param  Radius: Circle radius
00892   */
00893 void BSP_LCD_FillCircle(uint16_t Xpos, uint1
6_t Ypos, uint16_t Radius)
00894 {
00895   int32_t  D;     /* Decision Variable */
00896   uint32_t  CurX; /* Current X Value */
00897   uint32_t  CurY; /* Current Y Value */
00898
00899   D = 3 - (Radius << 1);
00900
00901   CurX = 0;
00902   CurY = Radius;
00903
00904   BSP_LCD_SetTextColor(DrawProp[ActiveLayer]
.TextColor);
00905
00906   while (CurX <= CurY)
00907   {
00908     if(CurY > 0)
00909     {
00910       BSP_LCD_DrawHLine(Xpos - CurY, Ypos +
CurX, 2*CurY);
00911       BSP_LCD_DrawHLine(Xpos - CurY, Ypos -
CurX, 2*CurY);
00912     }
00913
00914     if(CurX > 0)
00915     {
00916       BSP_LCD_DrawHLine(Xpos - CurX, Ypos -
CurY, 2*CurX);
00917       BSP_LCD_DrawHLine(Xpos - CurX, Ypos +
CurY, 2*CurX);
00918     }
00919     if (D < 0)
00920     {
00921       D += (CurX << 2) + 6;
```

```
00922     }
00923     else
00924     {
00925       D += ((CurX - CurY) << 2) + 10;
00926       CurY--;
00927     }
00928     CurX++;
00929   }
00930
00931   BSP_LCD_SetTextColor(DrawProp[ActiveLayer]
.TextColor);
00932   BSP_LCD_DrawCircle(Xpos, Ypos, Radius);
00933 }
00934
00935 /**
00936   * @brief  Draws a full poly-line (between
many points).
00937   * @param  Points: Pointer to the points ar
ray
00938   * @param  PointCount: Number of points
00939   */
00940 void BSP_LCD_FillPolygon(pPoint Points, uint
16_t PointCount)
00941 {
00942   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;
00943   uint16_t  IMAGE_LEFT = 0, IMAGE_RIGHT = 0,
 IMAGE_TOP = 0, IMAGE_BOTTOM = 0;
00944
00945   IMAGE_LEFT = IMAGE_RIGHT = Points->X;
00946   IMAGE_TOP= IMAGE_BOTTOM = Points->Y;
00947
00948   for(counter = 1; counter < PointCount; cou
nter++)
00949   {
00950     pixelX = POLY_X(counter);
```

```
00951     if(pixelX < IMAGE_LEFT)
00952     {
00953       IMAGE_LEFT = pixelX;
00954     }
00955     if(pixelX > IMAGE_RIGHT)
00956     {
00957       IMAGE_RIGHT = pixelX;
00958     }
00959
00960     pixelY = POLY_Y(counter);
00961     if(pixelY < IMAGE_TOP)
00962     {
00963       IMAGE_TOP = pixelY;
00964     }
00965     if(pixelY > IMAGE_BOTTOM)
00966     {
00967       IMAGE_BOTTOM = pixelY;
00968     }
00969   }
00970
00971   if(PointCount < 2)
00972   {
00973     return;
00974   }
00975
00976   X_center = (IMAGE_LEFT + IMAGE_RIGHT)/2;
00977   Y_center = (IMAGE_BOTTOM + IMAGE_TOP)/2;
00978
00979   X_first = Points->X;
00980   Y_first = Points->Y;
00981
00982   while(--PointCount)
00983   {
00984     X = Points->X;
00985     Y = Points->Y;
00986     Points++;
00987     X2 = Points->X;
```

```
00988     Y2 = Points->Y;
00989
00990     FillTriangle(X, X2, X_center, Y, Y2, Y_c
enter);
00991     FillTriangle(X, X_center, X2, Y, Y_cente
r, Y2);
00992     FillTriangle(X_center, X2, X, Y_center,
Y2, Y);
00993   }
00994
00995   FillTriangle(X_first, X2, X_center, Y_firs
t, Y2, Y_center);
00996   FillTriangle(X_first, X_center, X2, Y_firs
t, Y_center, Y2);
00997   FillTriangle(X_center, X2, X_first, Y_cent
er, Y2, Y_first);
00998 }
00999
01000 /**
01001   * @brief  Draws a full ellipse.
01002   * @param  Xpos: X position
01003   * @param  Ypos: Y position
01004   * @param  XRadius: Ellipse X radius
01005   * @param  YRadius: Ellipse Y radius
01006   */
01007 void BSP_LCD_FillEllipse(int Xpos, int Ypos,
int XRadius, int YRadius)
01008 {
01009   int x = 0, y = -YRadius, err = 2-2*XRadius
, e2;
01010   float K = 0, rad1 = 0, rad2 = 0;
01011
01012   rad1 = XRadius;
01013   rad2 = YRadius;
01014
01015   K = (float)(rad2/rad1);
01016
```

```
01017   do
01018   {
01019     BSP_LCD_DrawHLine((Xpos-(uint16_t)(x/K))
, (Ypos+y), (2*(uint16_t)(x/K) + 1));
01020     BSP_LCD_DrawHLine((Xpos-(uint16_t)(x/K))
, (Ypos-y), (2*(uint16_t)(x/K) + 1));
01021
01022     e2 = err;
01023     if (e2 <= x)
01024     {
01025       err += ++x*2+1;
01026       if (-y == x && e2 <= y) e2 = 0;
01027     }
01028     if (e2 > y) err += ++y*2+1;
01029   }
01030   while (y <= 0);
01031 }
01032
01033 /**
01034   * @brief  Enables the display.
01035   */
01036 void BSP_LCD_DisplayOn(void)
01037 {
01038   /* Display On */
01039   __HAL_LTDC_ENABLE(&hltdc_eval);
01040 }
01041
01042 /**
01043   * @brief  Disables the display.
01044   */
01045 void BSP_LCD_DisplayOff(void)
01046 {
01047   /* Display Off */
01048   __HAL_LTDC_DISABLE(&hltdc_eval);
01049 }
01050
01051 /*******************************************
```

```
**********************************
01052                            LTDC and DMA2D BSP Ro
utines
01053 ***********************************************
**********************************/
01054
01055 /**
01056   * @brief  Initializes the LTDC MSP.
01057   */
01058 static void MspInit(void)
01059 {
01060   GPIO_InitTypeDef GPIO_Init_Structure;
01061
01062   /* Enable the LTDC and DMA2D clocks */
01063   __LTDC_CLK_ENABLE();
01064   __DMA2D_CLK_ENABLE();
01065
01066   /* Enable GPIOs clock */
01067   __GPIOI_CLK_ENABLE();
01068   __GPIOJ_CLK_ENABLE();
01069   __GPIOK_CLK_ENABLE();
01070
01071   /*** LTDC Pins configuration ***/
01072   /* GPIOI configuration */
01073   GPIO_Init_Structure.Pin       = GPIO_PIN_1
2 | GPIO_PIN_13 | GPIO_PIN_14 | GPIO_PIN_15;
01074   GPIO_Init_Structure.Mode      = GPIO_MODE_
AF_PP;
01075   GPIO_Init_Structure.Pull      = GPIO_NOPUL
L;
01076   GPIO_Init_Structure.Speed     = GPIO_SPEED
_FAST;
01077   GPIO_Init_Structure.Alternate = GPIO_AF14_
LTDC;
01078   HAL_GPIO_Init(GPIOI, &GPIO_Init_Structure)
;
01079
```

```
01080   /* GPIOJ configuration */
01081   GPIO_Init_Structure.Pin       = GPIO_PIN_0
 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 | \
01082                                   GPIO_PIN_4
 | GPIO_PIN_5 | GPIO_PIN_6 | GPIO_PIN_7 | \
01083                                   GPIO_PIN_8
 | GPIO_PIN_9 | GPIO_PIN_10 | GPIO_PIN_11 | \
01084                                   GPIO_PIN_1
2 | GPIO_PIN_13 | GPIO_PIN_14 | GPIO_PIN_15;
01085   GPIO_Init_Structure.Mode      = GPIO_MODE_
AF_PP;
01086   GPIO_Init_Structure.Pull      = GPIO_NOPUL
L;
01087   GPIO_Init_Structure.Speed     = GPIO_SPEED
_FAST;
01088   GPIO_Init_Structure.Alternate = GPIO_AF14_
LTDC;
01089   HAL_GPIO_Init(GPIOJ, &GPIO_Init_Structure)
;
01090
01091   /* GPIOK configuration */
01092   GPIO_Init_Structure.Pin       = GPIO_PIN_0
 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 | \
01093                                   GPIO_PIN_4
 | GPIO_PIN_5 | GPIO_PIN_6 | GPIO_PIN_7;
01094   GPIO_Init_Structure.Mode      = GPIO_MODE_
AF_PP;
01095   GPIO_Init_Structure.Pull      = GPIO_NOPUL
L;
01096   GPIO_Init_Structure.Speed     = GPIO_SPEED
_FAST;
01097   GPIO_Init_Structure.Alternate = GPIO_AF14_
LTDC;
01098   HAL_GPIO_Init(GPIOK, &GPIO_Init_Structure)
;
01099 }
01100
```

```
01101 /**
01102   * @brief  Clock Config.
01103   * @param  hltdc: LTDC handle
01104   * @param  Params: LTDC pixel clock
01105   * @note   This API is called by BSP_LCD_In
it()
01106   *         Being __weak it can be overwritt
en by the application
01107   */
01108 __weak void BSP_LCD_ClockConfig(LTDC_HandleT
ypeDef *hltdc, void *Params)
01109 {
01110   static RCC_PeriphCLKInitTypeDef  periph_cl
k_init_struct;
01111
01112   if(stmpe811_ts_drv.ReadID(TS_I2C_ADDRESS)
== STMPE811_ID)
01113   {
01114     /* AMPIRE480272 LCD clock configuration
*/
01115     /* PLLSAI_VCO Input = HSE_VALUE/PLL_M =
1 Mhz */
01116     /* PLLSAI_VCO Output = PLLSAI_VCO Input
* PLLSAIN = 192 Mhz */
01117     /* PLLLCDCLK = PLLSAI_VCO Output/PLLSAIR
 = 192/5 = 38.4 Mhz */
01118     /* LTDC clock frequency = PLLLCDCLK / LT
DC_PLLSAI_DIVR_4 = 38.4/4 = 9.6Mhz */
01119     periph_clk_init_struct.PeriphClockSelect
ion = RCC_PERIPHCLK_LTDC;
01120     periph_clk_init_struct.PLLSAI.PLLSAIN =
192;
01121     periph_clk_init_struct.PLLSAI.PLLSAIR =
AMPIRE480272_FREQUENCY_DIVIDER;
01122     periph_clk_init_struct.PLLSAIDivR = RCC_
PLLSAIDIVR_4;
01123     HAL_RCCEx_PeriphCLKConfig(&periph_clk_in
```

```
it_struct);
01124   }
01125   else
01126   {
01127     /* The programmed LTDC pixel clock depen
ds on the vertical refresh rate of the panel 60Hz
=> 25.16MHz and
01128        the LCD/SDRAM bandwidth affected by t
he several access on the bus and the number of use
d layers.
01129     */
01130     if(*(uint32_t *)Params == LCD_MAX_PCLK)
01131     {
01132       /* In case of single layer the bandwid
th is arround 160MBytesPerSec ==> theorical PCLK o
f 40MHz */
01133       /* AMPIRE640480 typical PCLK is 25.16
MHz so the PLLSAI is configured to provide this cl
ock */
01134       /* AMPIRE640480 LCD clock configuratio
n */
01135       /* PLLSAI_VCO Input = HSE_VALUE/PLL_M
= 1 Mhz */
01136       /* PLLSAI_VCO Output = PLLSAI_VCO Inpu
t * PLLSAIN = 151 Mhz */
01137       /* PLLLCDCLK = PLLSAI_VCO Output/PLLSA
IR = 151/3 = 50.3 Mhz */
01138       /* LTDC clock frequency = PLLLCDCLK /
LTDC_PLLSAI_DIVR_2 = 50.3/2 = 25.16 Mhz */
01139       periph_clk_init_struct.PLLSAI.PLLSAIN
= 151;
01140     }
01141     else
01142     {
01143       /* In case of double layers the bandwi
dth is arround 72MBytesPerSec => 18MHz (<25,16MHz)
 */
```

```
01144       /* so the PLLSAI is configured to prov
ide this clock */
01145       /* AMPIRE640480 LCD clock configuratio
n */
01146       /* PLLSAI_VCO Input = HSE_VALUE/PLL_M
= 1 Mhz */
01147       /* PLLSAI_VCO Output = PLLSAI_VCO Inpu
t * PLLSAIN = 108 Mhz */
01148       /* PLLLCDCLK = PLLSAI_VCO Output/PLLSA
IR = 108/3 = 36 Mhz */
01149       /* LTDC clock frequency = PLLLCDCLK /
LTDC_PLLSAI_DIVR_2 = 36/2 = 18 Mhz */
01150       periph_clk_init_struct.PLLSAI.PLLSAIN
= 108;
01151     }
01152     periph_clk_init_struct.PeriphClockSelect
ion = RCC_PERIPHCLK_LTDC;
01153     periph_clk_init_struct.PLLSAI.PLLSAIR =
3;
01154     periph_clk_init_struct.PLLSAIDivR = RCC_
PLLSAIDIVR_2;
01155     HAL_RCCEx_PeriphCLKConfig(&periph_clk_in
it_struct);
01156   }
01157 }
01158 /*************************************************
**********************************
01159                                       Static Functions
01160 *************************************************
**********************************/
01161
01162 /**
01163   * @brief  Draws a pixel on LCD.
01164   * @param  Xpos: X position
01165   * @param  Ypos: Y position
01166   * @param  RGB_Code: Pixel color in ARGB mo
de (8-8-8-8)
```

```
01167   */
01168 void BSP_LCD_DrawPixel(uint16_t Xpos, uint16
_t Ypos, uint32_t RGB_Code)
01169 {
01170   /* Write data value to all SDRAM memory */
01171   *(__IO uint32_t*) (hltdc_eval.LayerCfg[Act
iveLayer].FBStartAdress + (4*(Ypos*BSP_LCD_GetXSize
() + Xpos))) = RGB_Code;
01172 }
01173
01174 /**
01175   * @brief  Draws a character on LCD.
01176   * @param  Xpos: Line where to display the
character shape
01177   * @param  Ypos: Start column address
01178   * @param  c: Pointer to the character data
01179   */
01180 static void DrawChar(uint16_t Xpos, uint16_t
 Ypos, const uint8_t *c)
01181 {
01182   uint32_t i = 0, j = 0;
01183   uint16_t height, width;
01184   uint8_t  offset;
01185   uint8_t  *pchar;
01186   uint32_t line;
01187
01188   height = DrawProp[ActiveLayer].pFont->Heig
ht;
01189   width  = DrawProp[ActiveLayer].pFont->Widt
h;
01190
01191   offset =  8 *((width + 7)/8) -  width ;
01192
01193   for(i = 0; i < height; i++)
01194   {
01195     pchar = ((uint8_t *)c + (width + 7)/8 *
i);
```

```
01196
01197     switch(((width + 7)/8))
01198     {
01199
01200     case 1:
01201       line =  pchar[0];
01202       break;
01203
01204     case 2:
01205       line =  (pchar[0]<< 8) | pchar[1];

01206       break;
01207
01208     case 3:
01209     default:
01210       line =  (pchar[0]<< 16) | (pchar[1]<<
8) | pchar[2];
01211       break;
01212     }
01213
01214     for (j = 0; j < width; j++)
01215     {
01216       if(line & (1 << (width- j + offset- 1)
))
01217       {
01218         BSP_LCD_DrawPixel((Xpos + j), Ypos,
DrawProp[ActiveLayer].TextColor);
01219       }
01220       else
01221       {
01222         BSP_LCD_DrawPixel((Xpos + j), Ypos,
DrawProp[ActiveLayer].BackColor);
01223       }
01224     }
01225     Ypos++;
01226   }
01227 }
```

```
01228
01229 /**
01230   * @brief  Fills a triangle (between 3 poin
ts).
01231   * @param  x1: Point 1 X position
01232   * @param  y1: Point 1 Y position
01233   * @param  x2: Point 2 X position
01234   * @param  y2: Point 2 Y position
01235   * @param  x3: Point 3 X position
01236   * @param  y3: Point 3 Y position
01237   */
01238 static void FillTriangle(uint16_t x1, uint16
_t x2, uint16_t x3, uint16_t y1, uint16_t y2, uint
16_t y3)
01239 {
01240   int16_t deltax = 0, deltay = 0, x = 0, y =
 0, xinc1 = 0, xinc2 = 0,
01241   yinc1 = 0, yinc2 = 0, den = 0, num = 0, nu
madd = 0, numpixels = 0,
01242   curpixel = 0;
01243
01244   deltax = ABS(x2 - x1);        /* The diffe
rence between the x's */
01245   deltay = ABS(y2 - y1);        /* The diffe
rence between the y's */
01246   x = x1;                       /* Start x o
ff at the first pixel */
01247   y = y1;                       /* Start y o
ff at the first pixel */
01248
01249   if (x2 >= x1)                 /* The x-val
ues are increasing */
01250   {
01251     xinc1 = 1;
01252     xinc2 = 1;
01253   }
01254   else                          /* The x-val
```

```
ues are decreasing */
01255   {
01256     xinc1 = -1;
01257     xinc2 = -1;
01258   }
01259
01260   if (y2 >= y1)                 /* The y-val
ues are increasing */
01261   {
01262     yinc1 = 1;
01263     yinc2 = 1;
01264   }
01265   else                          /* The y-val
ues are decreasing */
01266   {
01267     yinc1 = -1;
01268     yinc2 = -1;
01269   }
01270
01271   if (deltax >= deltay)         /* There is
at least one x-value for every y-value */
01272   {
01273     xinc1 = 0;                  /* Don't cha
nge the x when numerator >= denominator */
01274     yinc2 = 0;                  /* Don't cha
nge the y for every iteration */
01275     den = deltax;
01276     num = deltax / 2;
01277     numadd = deltay;
01278     numpixels = deltax;         /* There are
 more x-values than y-values */
01279   }
01280   else                          /* There is
at least one y-value for every x-value */
01281   {
01282     xinc2 = 0;                  /* Don't cha
nge the x for every iteration */
```

```
01283     yinc1 = 0;                    /* Don't cha
nge the y when numerator >= denominator */
01284     den = deltay;
01285     num = deltay / 2;
01286     numadd = deltax;
01287     numpixels = deltay;           /* There are
 more y-values than x-values */
01288   }
01289
01290   for (curpixel = 0; curpixel <= numpixels;
curpixel++)
01291   {
01292     BSP_LCD_DrawLine(x, y, x3, y3);
01293
01294     num += numadd;                /* Increase
the numerator by the top of the fraction */
01295     if (num >= den)               /* Check if
numerator >= denominator */
01296     {
01297       num -= den;                 /* Calculate
 the new numerator value */
01298       x += xinc1;                 /* Change th
e x as appropriate */
01299       y += yinc1;                 /* Change th
e y as appropriate */
01300     }
01301     x += xinc2;                   /* Change th
e x as appropriate */
01302     y += yinc2;                   /* Change th
e y as appropriate */
01303   }
01304 }
01305
01306 /**
01307   * @brief  Fills a buffer.
01308   * @param  LayerIndex: Layer index
01309   * @param  pDst: Pointer to destination buf
```

```
fer
01310   * @param  xSize: Buffer width
01311   * @param  ySize: Buffer height
01312   * @param  OffLine: Offset
01313   * @param  ColorIndex: Color index
01314   */
01315 static void LL_FillBuffer(uint32_t LayerInde
x, void *pDst, uint32_t xSize, uint32_t ySize, uin
t32_t OffLine, uint32_t ColorIndex)
01316 {
01317   /* Register to memory mode with ARGB8888 a
s color Mode */
01318   hdma2d_eval.Init.Mode         = DMA2D_R2M;
01319   hdma2d_eval.Init.ColorMode    = DMA2D_ARGB
8888;
01320   hdma2d_eval.Init.OutputOffset = OffLine;

01321
01322   hdma2d_eval.Instance = DMA2D;
01323
01324   /* DMA2D Initialization */
01325   if(HAL_DMA2D_Init(&hdma2d_eval) == HAL_OK)

01326   {
01327     if(HAL_DMA2D_ConfigLayer(&hdma2d_eval, L
ayerIndex) == HAL_OK)
01328     {
01329       if (HAL_DMA2D_Start(&hdma2d_eval, Colo
rIndex, (uint32_t)pDst, xSize, ySize) == HAL_OK)
01330       {
01331         /* Polling For DMA transfer */
01332         HAL_DMA2D_PollForTransfer(&hdma2d_ev
al, 10);
01333       }
01334     }
01335   }
01336 }
```

```
01337
01338 /**
01339   * @brief  Converts a line to an ARGB8888 p
ixel format.
01340   * @param  pSrc: Pointer to source buffer
01341   * @param  pDst: Output color
01342   * @param  xSize: Buffer width
01343   * @param  ColorMode: Input color mode
01344   */
01345 static void LL_ConvertLineToARGB8888(void *p
Src, void *pDst, uint32_t xSize, uint32_t ColorMod
e)
01346 {
01347   /* Configure the DMA2D Mode, Color Mode an
d output offset */
01348   hdma2d_eval.Init.Mode         = DMA2D_M2M_
PFC;
01349   hdma2d_eval.Init.ColorMode    = DMA2D_ARGB
8888;
01350   hdma2d_eval.Init.OutputOffset = 0;
01351
01352   /* Foreground Configuration */
01353   hdma2d_eval.LayerCfg[1].AlphaMode = DMA2D_
NO_MODIF_ALPHA;
01354   hdma2d_eval.LayerCfg[1].InputAlpha = 0xFF;
01355   hdma2d_eval.LayerCfg[1].InputColorMode = C
olorMode;
01356   hdma2d_eval.LayerCfg[1].InputOffset = 0;
01357
01358   hdma2d_eval.Instance = DMA2D;
01359
01360   /* DMA2D Initialization */
01361   if(HAL_DMA2D_Init(&hdma2d_eval) == HAL_OK)

01362   {
01363     if(HAL_DMA2D_ConfigLayer(&hdma2d_eval, 1
) == HAL_OK)
```

```
01364     {
01365       if (HAL_DMA2D_Start(&hdma2d_eval, (uin
t32_t)pSrc, (uint32_t)pDst, xSize, 1) == HAL_OK)
01366       {
01367         /* Polling For DMA transfer */
01368         HAL_DMA2D_PollForTransfer(&hdma2d_ev
al, 10);
01369       }
01370     }
01371   }
01372 }
01373
01374 /**
01375   * @}
01376   */
01377
01378 /**
01379   * @}
01380   */
01381
01382 /**
01383   * @}
01384   */
01385
01386 /**
01387   * @}
01388   */
01389
01390 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL NOR

STM324x9I EVAL

Modules

Defines

#define	**NOR_STATUS_OK** 0x00 NOR status structure definition.
#define	**NOR_STATUS_ERROR** 0x01

Functions

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

Define Documentation

#define NOR_STATUS_ERROR 0x01

Definition at line 73 of file stm324x9i_eval_nor.h.

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

#define NOR_STATUS_OK 0x00

NOR status structure definition.

Definition at line 72 of file stm324x9i_eval_nor.h.

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

Function Documentation

uint8_t BSP_NOR_Erase_Block (uint32_t BlockAddress)

Erases the specified block of the NOR device.

Parameters:

BlockAddress,: Block address to erase

Return values:

NOR memory status

Definition at line **282** of file **stm324x9i_eval_nor.c**.

References **BLOCKERASE_TIMEOUT**, **NOR_DEVICE_ADDR**, **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, and **norHandle**.

uint8_t BSP_NOR_Erase_Chip (void)

Erases the entire NOR chip.

Return values:

NOR memory status

Definition at line **302** of file **stm324x9i_eval_nor.c**.

References **CHIPERASE_TIMEOUT**, **NOR_DEVICE_ADDR**, **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, and **norHandle**.

uint8_t BSP_NOR_Init (void)

Initializes the NOR device.

Return values:

NOR memory status

Definition at line **154** of file **stm324x9i_eval_nor.c**.

References **CONTINUOUSCLOCK_FEATURE**, **NOR_BURSTACCESS**, **NOR_MEMORY_WIDTH**, **NOR_MspInit()**, **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, **NOR_WRITEBURST**, **norHandle**, and **Timing**.

uint8_t BSP_NOR_ProgramData (uint32_t uwStartAddress,
uint16_t * pData,
uint32_t uwDataSize
)

Programs an amount of data to the NOR device.

Parameters:

uwStartAddress,: Write start address
pData,: Pointer to data to be written
uwDataSize,: Size of data to write

Return values:

NOR memory status

Definition at line **261** of file **stm324x9i_eval_nor.c**.

References **NOR_DEVICE_ADDR**, **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, **norHandle**, and **PROGRAM_TIMEOUT**.

uint8_t BSP_NOR_Read_ID (NOR_IDTypeDef * pNOR_ID)

Reads NOR flash IDs.

Parameters:

pNOR_ID : Pointer to NOR ID structure

Return values:

NOR memory status

Definition at line **323** of file **stm324x9i_eval_nor.c**.

References **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, and **norHandle**.

uint8_t BSP_NOR_ReadData (uint32_t uwStartAddress, uint16_t * pData, uint32_t uwDataSize)

Reads an amount of data from the NOR device.

Parameters:

uwStartAddress,:	Read start address
pData,:	Pointer to data to be read
uwDataSize,:	Size of data to read

Return values:

NOR memory status

Definition at line **203** of file **stm324x9i_eval_nor.c**.

References **NOR_DEVICE_ADDR**, **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, and **norHandle**.

void BSP_NOR_ReturnToReadMode (void)

Returns the NOR memory to read mode.

Definition at line 218 of file stm324x9i_eval_nor.c.

References norHandle.

uint8_t BSP_NOR_WriteData (uint32_t uwStartAddress,
uint16_t * pData,
uint32_t uwDataSize
)

Writes an amount of data to the NOR device.

Parameters:

uwStartAddress,:	Write start address
pData,:	Pointer to data to be written
uwDataSize,:	Size of data to write

Return values:

NOR memory status

Definition at line 230 of file stm324x9i_eval_nor.c.

References NOR_DEVICE_ADDR, NOR_STATUS_ERROR, NOR_STATUS_OK, norHandle, and PROGRAM_TIMEOUT.

void HAL_NOR_MspWait (NOR_HandleTypeDef * hnor,
uint32_t Timeout
)

NOR BSP Wait for Ready/Busy signal.

Parameters:

hnor,:	Pointer to NOR handle
Timeout,:	Timeout duration

Definition at line **385** of file **stm324x9i_eval_nor.c**.

References **NOR_BUSY_STATE**, **NOR_READY_BUSY_GPIO**, **NOR_READY_BUSY_PIN**, and **NOR_READY_STATE**.

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL SD

STM324x9I EVAL

Modules

Defines

#define	**MSD_OK** 0x00 SD status structure definition.
#define	**MSD_ERROR** 0x01

Define Documentation

#define MSD_ERROR 0x01

Definition at line 79 of file stm324x9i_eval_sd.h.

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

#define MSD_OK 0x00

SD status structure definition.

Definition at line 78 of file stm324x9i_eval_sd.h.

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

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL SDRAM

STM324x9I EVAL

Modules

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL SRAM

STM324x9I EVAL

Modules

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL TS

STM324x9I EVAL

Modules

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL TS Exported Types

STM324x9I EVAL TS

Data Structures

struct	**TS_StateTypeDef**

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_ts.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval_ts.h
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm324x9i_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 __STM324x9I_EVAL_TS_H
00041 #define __STM324x9I_EVAL_TS_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------------------------------------------*/
00048 #include "stm324x9i_eval.h"
00049 /* Include IOExpander(STMPE811) component Driver */
00050 #include "../Components/stmpe811/stmpe811.h"
00051 /* Include TouchScreen component drivers */
00052 #include "../Components/ts3510/ts3510.h"
00053 #include "../Components/exc7200/exc7200.h"

00054 /** @addtogroup BSP
00055   * @{
00056   */
00057
00058 /** @addtogroup STM324x9I_EVAL
00059   * @{
00060   */
00061
00062 /** @defgroup STM324x9I_EVAL_TS STM324x9I EVAL TS
00063   * @{
00064   */
00065
00066 /** @defgroup STM324x9I_EVAL_TS_Exported_Typ
```

```
es STM324x9I EVAL TS Exported Types
00067   * @{
00068   */
00069 typedef struct
00070 {
00071   uint16_t TouchDetected;
00072   uint16_t x;
00073   uint16_t y;
00074   uint16_t z;
00075 }TS_StateTypeDef;
00076 /**
00077   * @}
00078   */
00079
00080 /** @defgroup STM324x9I_EVAL_TS_Exported_Con
stants STM324x9I EVAL TS Exported Constants
00081   * @{
00082   */
00083 #define TS_SWAP_NONE                    0x00
00084 #define TS_SWAP_X                       0x01
00085 #define TS_SWAP_Y                       0x02
00086 #define TS_SWAP_XY                      0x04
00087
00088 typedef enum
00089 {
00090   TS_OK       = 0x00,
00091   TS_ERROR    = 0x01,
00092   TS_TIMEOUT  = 0x02
00093 }TS_StatusTypeDef;
00094
00095 /* Interrupt sources pins definition */
00096 #define TS_INT_PIN                      0x00
10
00097 /**
00098   * @}
00099   */
00100
```

```
00101 /** @defgroup STM324x9I_EVAL_TS_Exported_Mac
ros STM324x9I EVAL TS Exported Macros
00102   * @{
00103   */
00104 /**
00105   * @}
00106   */
00107
00108 /** @defgroup STM324x9I_EVAL_TS_Exported_Fun
ctions STM324x9I EVAL TS Exported Functions
00109   * @{
00110   */
00111 uint8_t BSP_TS_Init(uint16_t xSize, uint16_t
 ySize);
00112 uint8_t BSP_TS_DeInit(void);
00113 uint8_t BSP_TS_GetState(TS_StateTypeDef *TS_
State);
00114 uint8_t BSP_TS_ITConfig(void);
00115 uint8_t BSP_TS_ITGetStatus(void);
00116 void    BSP_TS_ITClear(void);
00117
00118 /**
00119   * @}
00120   */
00121
00122 /**
00123   * @}
00124   */
00125
00126 /**
00127   * @}
00128   */
00129
00130 /**
00131   * @}
00132   */
00133
```

```
00134
00135 #ifdef __cplusplus
00136 }
00137 #endif
00138
00139 #endif /* __STM324x9I_EVAL_TS_H */
00140
00141 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_ts.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval_ts.c
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file provides a set of functions needed to manage the Touch
00008   *          Screen on STM324x9I-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 STM324x9I-EVAL
00044      evaluation board on the AMPIRE 640x480
LCD mounted on MB1063 or AMPIRE
00045      480x272 LCD mounted on MB1046 daughter
board.
00046    - If the AMPIRE 640x480 LCD is used, the
TS3510 or EXC7200 component driver
00047      must be included according to the touch
 screen driver present on this board.
00048    - If the AMPIRE 480x272 LCD is used, the
STMPE811 IO expander device component
00049      driver must be included in order to run
 the TS module commanded by the IO
00050      expander device, the STMPE1600 IO expan
der device component driver must be
00051      also included in case of interrupt mode
 use of the TS.
00052
00053 2. Driver description:
00054 ---------------------
00055   + Initialization steps:
00056      o Initialize the TS module using the BS
P_TS_Init() function. This
00057        function includes the MSP layer hardw
are resources initialization and the
```

```
00058         communication layer configuration to
start the TS use. The LCD size properties
00059         (x and y) are passed as parameters.
00060       o If TS interrupt mode is desired, you
must configure the TS interrupt mode
00061         by calling the function BSP_TS_ITConf
ig(). The TS interrupt mode is generated
00062         as an external interrupt whenever a t
ouch is detected.
00063         The interrupt mode internally uses th
e IO functionalities driver driven by
00064         the IO expander, to configure the IT
line.
00065
00066    + Touch screen use
00067       o The touch screen state is captured wh
enever the function BSP_TS_GetState() is
00068         used. This function returns informati
on about the last LCD touch occurred
00069         in the TS_StateTypeDef structure.
00070       o If TS interrupt mode is used, the fun
ction BSP_TS_ITGetStatus() is needed to get
00071         the interrupt status. To clear the IT
 pending bits, you should call the
00072         function BSP_TS_ITClear().
00073       o The IT is handled using the correspon
ding external interrupt IRQ handler,
00074         the user IT callback treatment is imp
lemented on the same external interrupt
00075         callback.
00076
00077 ----------------------------------------
---------------------------------*/
00078
00079 /* Includes --------------------------------
---------------------------------*/
00080 #include "stm324x9i_eval_ts.h"
```

```
00081 #include "stm324x9i_eval_io.h"
00082
00083 /** @addtogroup BSP
00084   * @{
00085   */
00086
00087 /** @addtogroup STM324x9I_EVAL
00088   * @{
00089   */
00090
00091 /** @defgroup STM324x9I_EVAL_TS STM324x9I EV
AL TS
00092   * @{
00093   */
00094
00095 /** @defgroup STM324x9I_EVAL_TS_Private_Type
s_Definitions STM324x9I EVAL TS Private Types Defi
nitions
00096   * @{
00097   */
00098 /**
00099   * @}
00100   */
00101
00102 /** @defgroup STM324x9I_EVAL_TS_Private_Defi
nes STM324x9I EVAL TS Private Defines
00103   * @{
00104   */
00105 /**
00106   * @}
00107   */
00108
00109 /** @defgroup STM324x9I_EVAL_TS_Private_Macr
os STM324x9I EVAL TS Private Macros
00110   * @{
00111   */
00112 /**
```

```
00113   * @}
00114   */
00115
00116 /** @defgroup STM324x9I_EVAL_TS_Private_Vari
ables STM324x9I EVAL TS Private Variables
00117   * @{
00118   */
00119 static TS_DrvTypeDef *ts_driver;
00120 static uint16_t ts_x_boundary, ts_y_boundary
;
00121 static uint8_t  ts_orientation;
00122 static uint8_t  I2C_Address;
00123 /**
00124   * @}
00125   */
00126
00127 /** @defgroup STM324x9I_EVAL_TS_Private_Func
tion_Prototypes STM324x9I EVAL TS Private Function
 Prototypes
00128   * @{
00129   */
00130 /**
00131   * @}
00132   */
00133
00134 /** @defgroup STM324x9I_EVAL_TS_Private_Func
tions STM324x9I EVAL TS Private Functions
00135   * @{
00136   */
00137
00138 /**
00139   * @brief  Initializes and configures the t
ouch screen functionalities and
00140   *         configures all necessary hardwar
e resources (GPIOs, clocks..).
00141   * @param  xSize: Maximum X size of the TS
area on LCD
```

```
00142   * @param  ySize: Maximum Y size of the TS
area on LCD
00143   * @retval TS_OK if all initializations are
 OK. Other value if error.
00144   */
00145 uint8_t BSP_TS_Init(uint16_t xSize, uint16_t
 ySize)
00146 {
00147   uint8_t status = TS_OK;
00148   ts_x_boundary = xSize;
00149   ts_y_boundary = ySize;
00150
00151   /* Read ID and verify if the IO expander i
s ready */
00152   if(stmpe811_ts_drv.ReadID(TS_I2C_ADDRESS)
== STMPE811_ID)
00153   {
00154     /* Initialize the TS driver structure */
00155     ts_driver = &stmpe811_ts_drv;
00156     I2C_Address = TS_I2C_ADDRESS;
00157     ts_orientation = TS_SWAP_Y;
00158   }
00159   else
00160   {
00161     IOE_Init();
00162
00163     /* Check TS3510 touch screen driver pres
ence to determine if TS3510 or
00164      * EXC7200 driver will be used */
00165     if(BSP_TS3510_IsDetected() == 0)
00166     {
00167       /* Initialize the TS driver structure
*/
00168       ts_driver = &ts3510_ts_drv;
00169       I2C_Address = TS3510_I2C_ADDRESS;
00170     }
00171     else
```

```
00172     {
00173       /* Initialize the TS driver structure
*/
00174       ts_driver = &exc7200_ts_drv;
00175       I2C_Address = EXC7200_I2C_ADDRESS;
00176     }
00177     ts_orientation = TS_SWAP_NONE;
00178   }
00179
00180   /* Initialize the TS driver */
00181   ts_driver->Init(I2C_Address);
00182   ts_driver->Start(I2C_Address);
00183
00184   return status;
00185 }
00186
00187 /**
00188   * @brief  DeInitializes the TouchScreen.
00189   * @retval TS state
00190   */
00191 uint8_t BSP_TS_DeInit(void)
00192 {
00193   /* Actually ts_driver does not provide a D
eInit function */
00194   return TS_OK;
00195 }
00196
00197 /**
00198   * @brief  Configures and enables the touch
 screen interrupts.
00199   * @retval TS_OK if all initializations are
 OK. Other value if error.
00200   */
00201 uint8_t BSP_TS_ITConfig(void)
00202 {
00203   /* Initialize the IO */
00204   BSP_IO_Init();
```

```
00205
00206   /* Configure TS IT line IO */
00207   BSP_IO_ConfigPin(TS_INT_PIN, IO_MODE_IT_FA
LLING_EDGE);
00208
00209   /* Enable the TS ITs */
00210   ts_driver->EnableIT(I2C_Address);
00211
00212   return TS_OK;
00213 }
00214
00215 /**
00216   * @brief  Gets the touch screen interrupt
status.
00217   * @retval TS_OK if all initializations are
 OK. Other value if error.
00218   */
00219 uint8_t BSP_TS_ITGetStatus(void)
00220 {
00221   /* Return the TS IT status */
00222   return (ts_driver->GetITStatus(I2C_Address
));
00223 }
00224
00225 /**
00226   * @brief  Returns status and positions of
the touch screen.
00227   * @param  TS_State: Pointer to touch scree
n current state structure
00228   * @retval TS_OK if all initializations are
 OK. Other value if error.
00229   */
00230 uint8_t BSP_TS_GetState(TS_StateTypeDef *TS_
State)
00231 {
00232   static uint32_t _x = 0, _y = 0;
00233   uint16_t xDiff, yDiff , x , y;
```

```
00234   uint16_t swap;
00235
00236   TS_State->TouchDetected = ts_driver->Detec
tTouch(I2C_Address);
00237
00238   if(TS_State->TouchDetected)
00239   {
00240     ts_driver->GetXY(I2C_Address, &x, &y);
00241
00242     if(ts_orientation & TS_SWAP_X)
00243     {
00244       x = 4096 - x;
00245     }
00246
00247     if(ts_orientation & TS_SWAP_Y)
00248     {
00249       y = 4096 - y;
00250     }
00251
00252     if(ts_orientation & TS_SWAP_XY)
00253     {
00254       swap = y;
00255       y = x;
00256       x = swap;
00257     }
00258
00259     xDiff = x > _x? (x - _x): (_x - x);
00260     yDiff = y > _y? (y - _y): (_y - y);
00261
00262     if (xDiff + yDiff > 5)
00263     {
00264       _x = x;
00265       _y = y;
00266     }
00267
00268       TS_State->x = (ts_x_boundary * _x) >>
12;
```

```
00269       TS_State->y = (ts_y_boundary * _y) >>
12;
00270   }
00271   return TS_OK;
00272 }
00273
00274 /**
00275   * @brief  Clears all touch screen interrup
ts.
00276   */
00277 void BSP_TS_ITClear(void)
00278 {
00279   /* Clear all IO IT pin */
00280   BSP_IO_ITClear();
00281
00282   /* Clear TS IT pending bits */
00283   ts_driver->ClearIT(I2C_Address);
00284 }
00285
00286 /**
00287   * @}
00288   */
00289
00290 /**
00291   * @}
00292   */
00293
00294 /**
00295   * @}
00296   */
00297
00298 /**
00299   * @}
00300   */
00301
00302 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP
User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

Defines

STM324x9I EVAL
SDRAM Exported Constants

STM324x9I 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** __DMA2_CLK_ENABLE
#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_ENABLE __DMA2_CLK_ENABLE

Definition at line 94 of file stm324x9i_eval_sdram.h.

Referenced by SDRAM_MspInit().

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

Definition at line 89 of file stm324x9i_eval_sdram.h.

Referenced by BSP_SDRAM_Init().

#define SDCLOCK_PERIOD FMC_SDRAM_CLOCK_PERIOD_2

Definition at line 86 of file stm324x9i_eval_sdram.h.

Referenced by BSP_SDRAM_Init().

#define SDRAM_DEVICE_ADDR ((uint32_t)0xC0000000)

Definition at line 79 of file stm324x9i_eval_sdram.h.

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

Definition at line 80 of file stm324x9i_eval_sdram.h.

#define SDRAM_DMAx_CHANNEL DMA_CHANNEL_0

Definition at line 95 of file stm324x9i_eval_sdram.h.

Referenced by **SDRAM_MspInit()**.

#define SDRAM_DMAx_IRQHandler DMA2_Stream0_IRQHandler

Definition at line **98** of file **stm324x9i_eval_sdram.h**.

#define SDRAM_DMAx_IRQn DMA2_Stream0_IRQn

Definition at line **97** of file **stm324x9i_eval_sdram.h**.

Referenced by **SDRAM_MspInit()**.

#define SDRAM_DMAx_STREAM DMA2_Stream0

Definition at line **96** of file **stm324x9i_eval_sdram.h**.

Referenced by **SDRAM_MspInit()**.

#define SDRAM_MEMORY_WIDTH FMC_SDRAM_MEM_BUS_WID

Definition at line **84** of file **stm324x9i_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 **103** of file **stm324x9i_eval_sdram.h**.

Referenced by **BSP_SDRAM_Initialization_sequence()**.

#define SDRAM_MODEREG_BURST_LENGTH_2 ((uint16_t)0x0001

Definition at line 104 of file stm324x9i_eval_sdram.h.

#define SDRAM_MODEREG_BURST_LENGTH_4 ((uint16_t)0x0002

Definition at line 105 of file stm324x9i_eval_sdram.h.

#define SDRAM_MODEREG_BURST_LENGTH_8 ((uint16_t)0x0004

Definition at line 106 of file stm324x9i_eval_sdram.h.

#define SDRAM_MODEREG_BURST_TYPE_INTERLEAVED ((uint1

Definition at line 108 of file stm324x9i_eval_sdram.h.

#define SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL ((uint16

Definition at line 107 of file stm324x9i_eval_sdram.h.

Referenced by BSP_SDRAM_Initialization_sequence().

#define SDRAM_MODEREG_CAS_LATENCY_2 ((uint16_t)0x0020)

Definition at line 109 of file stm324x9i_eval_sdram.h.

#define SDRAM_MODEREG_CAS_LATENCY_3 ((uint16_t)0x0030)

Definition at line 110 of file stm324x9i_eval_sdram.h.

Referenced by BSP_SDRAM_Initialization_sequence().

#define SDRAM_MODEREG_OPERATING_MODE_STANDARD ((ui

Definition at line 111 of file stm324x9i_eval_sdram.h.

Referenced by BSP_SDRAM_Initialization_sequence().

#define SDRAM_MODEREG_WRITEBURST_MODE_PROGRAMMED

Definition at line 112 of file stm324x9i_eval_sdram.h.

#define SDRAM_MODEREG_WRITEBURST_MODE_SINGLE ((uint

Definition at line 113 of file stm324x9i_eval_sdram.h.

Referenced by BSP_SDRAM_Initialization_sequence().

#define SDRAM_TIMEOUT ((uint32_t)0xFFFF)

Definition at line 91 of file stm324x9i_eval_sdram.h.

Referenced by BSP_SDRAM_Initialization_sequence(), and BSP_SDRAM_Sendcmd().

STM324x9I_EVAL BSP User Manual

Defines

STM324x9I EVAL SD Exported Constants

STM324x9I 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** __DMA2_CLK_ENABLE
#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 __DMA2_CLK_ENABLE

Definition at line 90 of file stm324x9i_eval_sd.h.

Referenced by SD_MspInit().

#define SD_DATATIMEOUT ((uint32_t)100000000)

Definition at line 87 of file stm324x9i_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 99 of file stm324x9i_eval_sd.h.

#define SD_DMAx_Rx_CHANNEL DMA_CHANNEL_4

Definition at line 92 of file stm324x9i_eval_sd.h.

Referenced by SD_MspInit().

#define SD_DMAx_Rx_IRQHandler DMA2_Stream3_IRQHandler

Definition at line 98 of file stm324x9i_eval_sd.h.

#define SD_DMAx_Rx_IRQn DMA2_Stream3_IRQn

Definition at line **96** of file **stm324x9i_eval_sd.h**.

Referenced by **SD_MspInit()**.

#define SD_DMAx_Rx_STREAM DMA2_Stream3

Definition at line **94** of file **stm324x9i_eval_sd.h**.

Referenced by **SD_MspInit()**.

#define SD_DMAx_Tx_CHANNEL DMA_CHANNEL_4

Definition at line **91** of file **stm324x9i_eval_sd.h**.

Referenced by **SD_MspInit()**.

#define SD_DMAx_Tx_IRQHandler DMA2_Stream6_IRQHandler

Definition at line **97** of file **stm324x9i_eval_sd.h**.

#define SD_DMAx_Tx_IRQn DMA2_Stream6_IRQn

Definition at line **95** of file **stm324x9i_eval_sd.h**.

Referenced by **SD_MspInit()**.

#define SD_DMAx_Tx_STREAM DMA2_Stream6

Definition at line **93** of file **stm324x9i_eval_sd.h**.

Referenced by **SD_MspInit()**.

#define SD_NOT_PRESENT ((uint8_t)0x00)

Definition at line 85 of file stm324x9i_eval_sd.h.

Referenced by BSP_SD_IsDetected().

#define SD_PRESENT ((uint8_t)0x01)

Definition at line 84 of file stm324x9i_eval_sd.h.

Referenced by BSP_SD_Init(), and BSP_SD_IsDetected().

STM324x9I_EVAL BSP User Manual

Defines

STM324x9I EVAL
SRAM Exported Constants

STM324x9I EVAL SRAM

Defines

#define	**SRAM_OK** 0x00 SRAM status structure definition.
#define	**SRAM_ERROR** 0x01
#define	**SRAM_DEVICE_ADDR** ((uint32_t)0x64000000)
#define	**SRAM_DEVICE_SIZE** ((uint32_t)0x200000) /* SRAM device
#define	**SRAM_MEMORY_WIDTH** FMC_NORSRAM_MEM_BUS_WI
#define	**SRAM_BURSTACCESS** FMC_BURST_ACCESS_MODE_DI
#define	**SRAM_WRITEBURST** FMC_WRITE_BURST_DISABLE
#define	**CONTINUOUSCLOCK_FEATURE** FMC_CONTINUOUS_CL
#define	**__SRAM_DMAx_CLK_ENABLE** __DMA2_CLK_ENABLE
#define	**SRAM_DMAx_CHANNEL** DMA_CHANNEL_0
#define	**SRAM_DMAx_STREAM** DMA2_Stream0
#define	**SRAM_DMAx_IRQn** DMA2_Stream0_IRQn
#define	**SRAM_DMAx_IRQHandler** DMA2_Stream0_IRQHandler

Define Documentation

#define __SRAM_DMAx_CLK_ENABLE __DMA2_CLK_ENABLE

Definition at line **95** of file **stm324x9i_eval_sram.h**.

Referenced by **SRAM_MspInit()**.

#define CONTINUOUSCLOCK_FEATURE FMC_CONTINUOUS_CL(

Definition at line **91** of file **stm324x9i_eval_sram.h**.

#define SRAM_BURSTACCESS FMC_BURST_ACCESS_MODE_DI

Definition at line **85** of file **stm324x9i_eval_sram.h**.

Referenced by **BSP_SRAM_Init()**.

#define SRAM_DEVICE_ADDR ((uint32_t)0x64000000)

Definition at line **79** of file **stm324x9i_eval_sram.h**.

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

Definition at line **80** of file **stm324x9i_eval_sram.h**.

#define SRAM_DMAx_CHANNEL DMA_CHANNEL_0

Definition at line **96** of file **stm324x9i_eval_sram.h**.

Referenced by **SRAM_MspInit()**.

#define SRAM_DMAx_IRQHandler DMA2_Stream0_IRQHandler

Definition at line 99 of file stm324x9i_eval_sram.h.

#define SRAM_DMAx_IRQn DMA2_Stream0_IRQn

Definition at line 98 of file stm324x9i_eval_sram.h.

Referenced by SRAM_MspInit().

#define SRAM_DMAx_STREAM DMA2_Stream0

Definition at line 97 of file stm324x9i_eval_sram.h.

Referenced by SRAM_MspInit().

#define SRAM_ERROR 0x01

Definition at line 77 of file stm324x9i_eval_sram.h.

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

#define SRAM_MEMORY_WIDTH FMC_NORSRAM_MEM_BUS_WI

Definition at line 83 of file stm324x9i_eval_sram.h.

Referenced by BSP_SRAM_Init().

#define SRAM_OK 0x00

SRAM status structure definition.

Definition at line **76** of file **stm324x9i_eval_sram.h**.

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

#define SRAM_WRITEBURST FMC_WRITE_BURST_DISABLE

Definition at line **88** of file **stm324x9i_eval_sram.h**.

Referenced by **BSP_SRAM_Init()**.

STM324x9I_EVAL BSP User Manual

Defines

STM324x9I EVAL LOW LEVEL Private Defines

STM324x9I EVAL LOW LEVEL

Defines

#define	**__STM324x9I_EVAL_BSP_VERSION_MAIN** (0x02) STM324x9I EVAL BSP Driver version number V2.2.2.
#define	**__STM324x9I_EVAL_BSP_VERSION_SUB1** (0x02)
#define	**__STM324x9I_EVAL_BSP_VERSION_SUB2** (0x02)
#define	**__STM324x9I_EVAL_BSP_VERSION_RC** (0x00)
#define	**__STM324x9I_EVAL_BSP_VERSION**

Define Documentation

#define __STM324x9I_EVAL_BSP_VERSION

Value:

```
((__STM324x9I_EVAL_BSP_VERSION_MAIN << 24)\
                                                |(__
STM324x9I_EVAL_BSP_VERSION_SUB1 << 16)\
                                                |(__
STM324x9I_EVAL_BSP_VERSION_SUB2 << 8 )\
                                                |(__
STM324x9I_EVAL_BSP_VERSION_RC))
```

Definition at line **80** of file **stm324x9i_eval.c**.

Referenced by **BSP_GetVersion()**.

#define __STM324x9I_EVAL_BSP_VERSION_MAIN (0x02)

STM324x9I EVAL BSP Driver version number V2.2.2.

[31:24] main version

Definition at line **76** of file **stm324x9i_eval.c**.

#define __STM324x9I_EVAL_BSP_VERSION_RC (0x00)

[7:0] release candidate

Definition at line **79** of file **stm324x9i_eval.c**.

#define __STM324x9I_EVAL_BSP_VERSION_SUB1 (0x02)

[23:16] sub1 version

Definition at line **77** of file **stm324x9i_eval.c**.

#define __STM324x9I_EVAL_BSP_VERSION_SUB2 (0x02)

[15:8] sub2 version

Definition at line **78** of file **stm324x9i_eval.c**.

STM324x9I_EVAL BSP User Manual

Defines

STM324x9I EVAL LCD Private Macros

STM324x9I EVAL LCD

Defines

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

Define Documentation

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

Definition at line **117** of file **stm324x9i_eval_lcd.c**.

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

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL LCD
Private Variables

STM324x9I EVAL LCD

Variables

static LTDC_HandleTypeDef	**hltdc_eval**
static DMA2D_HandleTypeDef	**hdma2d_eval**
static uint32_t	**PCLK_profile** = **LCD_MAX_PCLK**
static uint32_t	**ActiveLayer** = 0
static **LCD_DrawPropTypeDef**	**DrawProp** [**MAX_LAYER_NUMBER**]

Variable Documentation

uint32_t ActiveLayer = 0 [static]

Definition at line 130 of file stm324x9i_eval_lcd.c.

Referenced by BSP_LCD_Clear(), BSP_LCD_ClearStringLine(), BSP_LCD_DisplayChar(), BSP_LCD_DisplayStringAt(), BSP_LCD_DrawBitmap(), BSP_LCD_DrawCircle(), BSP_LCD_DrawEllipse(), BSP_LCD_DrawHLine(), BSP_LCD_DrawLine(), BSP_LCD_DrawPixel(), BSP_LCD_DrawVLine(), BSP_LCD_FillCircle(), BSP_LCD_FillRect(), BSP_LCD_GetBackColor(), BSP_LCD_GetFont(), BSP_LCD_GetTextColor(), BSP_LCD_GetXSize(), BSP_LCD_GetYSize(), BSP_LCD_ReadPixel(), BSP_LCD_SelectLayer(), BSP_LCD_SetBackColor(), BSP_LCD_SetFont(), BSP_LCD_SetTextColor(), and DrawChar().

LCD_DrawPropTypeDef DrawProp[MAX_LAYER_NUMBER] [static

Definition at line 131 of file stm324x9i_eval_lcd.c.

DMA2D_HandleTypeDef hdma2d_eval [static]

Definition at line 126 of file stm324x9i_eval_lcd.c.

Referenced by LL_ConvertLineToARGB8888(), and LL_FillBuffer().

LTDC_HandleTypeDef hltdc_eval [static]

Definition at line 125 of file stm324x9i_eval_lcd.c.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_DisplayOff()**, **BSP_LCD_DisplayOn()**, **BSP_LCD_DrawBitmap()**, **BSP_LCD_DrawHLine()**, **BSP_LCD_DrawPixel()**, **BSP_LCD_DrawVLine()**, **BSP_LCD_FillRect()**, **BSP_LCD_GetXSize()**, **BSP_LCD_GetYSize()**, **BSP_LCD_InitEx()**, **BSP_LCD_LayerDefaultInit()**, **BSP_LCD_ReadPixel()**, **BSP_LCD_ResetColorKeying()**, **BSP_LCD_SetColorKeying()**, **BSP_LCD_SetLayerAddress()**, **BSP_LCD_SetLayerVisible()**, **BSP_LCD_SetLayerWindow()**, and **BSP_LCD_SetTransparency()**.

uint32_t PCLK_profile = LCD_MAX_PCLK `[static]`

Definition at line **127** of file **stm324x9i_eval_lcd.c**.

Referenced by **BSP_LCD_InitEx()**.

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL AUDIO Private Variables

STM324x9I 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 137 of file stm324x9i_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 162 of file stm324x9i_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 **143** of file **stm324x9i_eval_audio.c**.

Referenced by **BSP_AUDIO_IN_PDMToPCM()**.

PDMFilter_InitStruct Filter[2]

Definition at line **142** of file **stm324x9i_eval_audio.c**.

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

I2S_HandleTypeDef haudio_in_i2s

Definition at line **139** of file **stm324x9i_eval_audio.c**.

Referenced by **BSP_AUDIO_IN_Pause()**, **BSP_AUDIO_IN_Record()**, **BSP_AUDIO_IN_Resume()**, **BSP_AUDIO_IN_Stop()**, **I2Sx_Init()**, and **I2Sx_MspInit()**.

SAI_HandleTypeDef haudio_out_sai

Definition at line **138** of file **stm324x9i_eval_audio.c**.

Referenced by **BSP_AUDIO_OUT_ChangeBuffer()**, **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_Init()**, and **SAIx_MspInit()**.

TIM_HandleTypeDef haudio_tim

Definition at line **140** of file **stm324x9i_eval_audio.c**.

Referenced by **TIMx_Init()**.

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL AUDIO Exported Constants

STM324x9I EVAL AUDIO

Modules

CODEC AudioFrame SLOT TDMMode

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** SAI1_Block_B
#define	**AUDIO_SAIx_CLK_ENABLE**() __SAI1_CLK_ENABLE()
#define	**AUDIO_SAIx_MCLK_SCK_SD_FS_AF** GPIO_AF6_SAI1
#define	**AUDIO_SAIx_MCLK_SCK_SD_FS_ENABLE**() __GPIOF_CL
#define	**AUDIO_SAIx_FS_PIN** GPIO_PIN_9
#define	**AUDIO_SAIx_SCK_PIN** GPIO_PIN_8
#define	**AUDIO_SAIx_SD_PIN** GPIO_PIN_6
#define	**AUDIO_SAIx_MCK_PIN** GPIO_PIN_7
#define	**AUDIO_SAIx_MCLK_SCK_SD_FS_GPIO_PORT** GPIOF
#define	**AUDIO_SAIx_DMAx_CLK_ENABLE**() __DMA2_CLK_ENAB
#define	**AUDIO_SAIx_DMAx_STREAM** DMA2_Stream5
#define	**AUDIO_SAIx_DMAx_CHANNEL** DMA_CHANNEL_0
#define	**AUDIO_SAIx_DMAx_IRQ** DMA2_Stream5_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_Stream5_IRQHan
#define	**AUDIO_OUT_IRQ_PREPRIO** 5 /* Select the preemption prio
#define	**AUDIO_I2Sx** SPI3
#define	**AUDIO_I2Sx_CLK_ENABLE**() __SPI3_CLK_ENABLE()
#define	**AUDIO_I2Sx_SCK_PIN** GPIO_PIN_3
#define	**AUDIO_I2Sx_SCK_GPIO_PORT** GPIOB
#define	**AUDIO_I2Sx_SCK_GPIO_CLK_ENABLE**() __GPIOB_CLK_
#define	**AUDIO_I2Sx_SCK_AF** GPIO_AF6_SPI3
#define	**AUDIO_I2Sx_SD_PIN** GPIO_PIN_6
#define	**AUDIO_I2Sx_SD_GPIO_PORT** GPIOD
#define	**AUDIO_I2Sx_SD_GPIO_CLK_ENABLE**() __GPIOD_CLK_E
#define	**AUDIO_I2Sx_SD_AF** GPIO_AF5_I2S3ext
#define	**AUDIO_I2Sx_DMAx_CLK_ENABLE**() __DMA1_CLK_ENAB
#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_IRQHanc
#define AUDIO_IN_IRQ_PREPRIO 6 /* Select the preemption priority
#define AUDIO_TIMx_CLK_ENABLE() __TIM3_CLK_ENABLE()
#define AUDIO_TIMx_CLK_DISABLE() __TIM3_CLK_DISABLE()
#define AUDIO_TIMx TIM3
#define AUDIO_TIMx_IN_CHANNEL TIM_CHANNEL_1
#define AUDIO_TIMx_OUT_CHANNEL TIM_CHANNEL_2 /* Select c
#define AUDIO_TIMx_GPIO_CLK_ENABLE() __GPIOC_CLK_ENAE
#define AUDIO_TIMx_GPIO GPIOC
#define AUDIO_TIMx_IN_GPIO_PIN GPIO_PIN_6
#define AUDIO_TIMx_OUT_GPIO_PIN GPIO_PIN_7
#define AUDIO_TIMx_AF GPIO_AF2_TIM3
#define AUDIODATA_SIZE 2 /* 16-bits audio data size */
#define AUDIO_OK 0
#define AUDIO_ERROR 1
#define AUDIO_TIMEOUT 2
#define DEFAULT_AUDIO_IN_FREQ I2S_AUDIOFREQ_16K
#define DEFAULT_AUDIO_IN_BIT_RESOLUTION 16
#define DEFAULT_AUDIO_IN_CHANNEL_NBR 2 /* Mono = 1, Stere
#define DEFAULT_AUDIO_IN_VOLUME 64
#define INTERNAL_BUFF_SIZE 128*DEFAULT_AUDIO_IN_FREQ/1
#define PCM_OUT_SIZE DEFAULT_AUDIO_IN_FREQ/1000*2
#define CHANNEL_DEMUX_MASK 0x55
#define CODEC_RESET_DELAY 5
```

Define Documentation

#define AUDIO_ERROR 1

Definition at line 178 of file stm324x9i_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 128 of file stm324x9i_eval_audio.h.

Referenced by I2Sx_Init(), and I2Sx_MspInit().

#define AUDIO_I2Sx_CLK_ENABLE ( ) __SPI3_CLK_ENABLE()

Definition at line 129 of file stm324x9i_eval_audio.h.

Referenced by I2Sx_MspInit().

#define AUDIO_I2Sx_DMAx_CHANNEL DMA_CHANNEL_0

Definition at line 143 of file stm324x9i_eval_audio.h.

Referenced by I2Sx_MspInit().

#define AUDIO_I2Sx_DMAx_CLK_ENABLE ( ) __DMA1_CLK_ENA

Definition at line **141** of file **stm324x9i_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

#define AUDIO_I2Sx_DMAx_IRQ DMA1_Stream2_IRQn

Definition at line **144** of file **stm324x9i_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

#define AUDIO_I2Sx_DMAx_IRQHandler DMA1_Stream2_IRQHan

Definition at line **148** of file **stm324x9i_eval_audio.h**.

#define AUDIO_I2Sx_DMAx_MEM_DATA_SIZE DMA_MDATAALIGN

Definition at line **146** of file **stm324x9i_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

#define AUDIO_I2Sx_DMAx_PERIPH_DATA_SIZE DMA_PDATAAL

Definition at line **145** of file **stm324x9i_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

#define AUDIO_I2Sx_DMAx_STREAM DMA1_Stream2

Definition at line **142** of file **stm324x9i_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

#define AUDIO_I2Sx_SCK_AF GPIO_AF6_SPI3

Definition at line 133 of file stm324x9i_eval_audio.h.

Referenced by I2Sx_MspInit().

#define AUDIO_I2Sx_SCK_GPIO_CLK_ENABLE ( ) __GPIOB_CLK

Definition at line 132 of file stm324x9i_eval_audio.h.

Referenced by I2Sx_MspInit().

#define AUDIO_I2Sx_SCK_GPIO_PORT GPIOB

Definition at line 131 of file stm324x9i_eval_audio.h.

Referenced by I2Sx_MspInit().

#define AUDIO_I2Sx_SCK_PIN GPIO_PIN_3

Definition at line 130 of file stm324x9i_eval_audio.h.

Referenced by I2Sx_MspInit().

#define AUDIO_I2Sx_SD_AF GPIO_AF5_I2S3ext

Definition at line 138 of file stm324x9i_eval_audio.h.

Referenced by I2Sx_MspInit().

#define AUDIO_I2Sx_SD_GPIO_CLK_ENABLE ( ) __GPIOD_CLK_

Definition at line **137** of file **stm324x9i_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

#define AUDIO_I2Sx_SD_GPIO_PORT GPIOD

Definition at line **136** of file **stm324x9i_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

#define AUDIO_I2Sx_SD_PIN GPIO_PIN_6

Definition at line **135** of file **stm324x9i_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

#define AUDIO_IN_IRQ_PREPRIO 6 /* Select the preemption prior

Definition at line **151** of file **stm324x9i_eval_audio.h**.

Referenced by **I2Sx_MspInit()**.

#define AUDIO_OK 0

Definition at line **177** of file **stm324x9i_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 5 /* Select the preemption pri

Definition at line 122 of file stm324x9i_eval_audio.h.

Referenced by SAIx_MspInit().

#define AUDIO_SAIx SAI1_Block_B

Definition at line 98 of file stm324x9i_eval_audio.h.

Referenced by SAIx_Init(), and SAIx_MspInit().

#define AUDIO_SAIx_CLK_ENABLE ( ) __SAI1_CLK_ENABLE()

Definition at line 99 of file stm324x9i_eval_audio.h.

Referenced by SAIx_MspInit().

#define AUDIO_SAIx_DMAx_CHANNEL DMA_CHANNEL_0

Definition at line 113 of file stm324x9i_eval_audio.h.

Referenced by SAIx_MspInit().

#define AUDIO_SAIx_DMAx_CLK_ENABLE ( ) __DMA2_CLK_ENA

Definition at line 111 of file stm324x9i_eval_audio.h.

Referenced by SAIx_MspInit().

#define AUDIO_SAIx_DMAx_IRQ DMA2_Stream5_IRQn

Definition at line 114 of file stm324x9i_eval_audio.h.

Referenced by SAIx_MspInit().

#define AUDIO_SAIx_DMAx_IRQHandler DMA2_Stream5_IRQHan

Definition at line 119 of file stm324x9i_eval_audio.h.

#define AUDIO_SAIx_DMAx_MEM_DATA_SIZE DMA_MDATAALIG

Definition at line 116 of file stm324x9i_eval_audio.h.

Referenced by SAIx_MspInit().

#define AUDIO_SAIx_DMAx_PERIPH_DATA_SIZE DMA_PDATAAL

Definition at line 115 of file stm324x9i_eval_audio.h.

Referenced by SAIx_MspInit().

#define AUDIO_SAIx_DMAx_STREAM DMA2_Stream5

Definition at line 112 of file stm324x9i_eval_audio.h.

Referenced by SAIx_MspInit().

#define AUDIO_SAIx_FS_PIN GPIO_PIN_9

Definition at line 103 of file stm324x9i_eval_audio.h.

Referenced by **SAIx_MspInit()**.

#define AUDIO_SAIx_MCK_PIN GPIO_PIN_7

Definition at line **106** of file **stm324x9i_eval_audio.h**.

Referenced by **SAIx_MspInit()**.

#define AUDIO_SAIx_MCLK_SCK_SD_FS_AF GPIO_AF6_SAI1

Definition at line **100** of file **stm324x9i_eval_audio.h**.

Referenced by **SAIx_MspInit()**.

#define AUDIO_SAIx_MCLK_SCK_SD_FS_ENABLE ( ) __GPIOF_

Definition at line **102** of file **stm324x9i_eval_audio.h**.

Referenced by **SAIx_MspInit()**.

#define AUDIO_SAIx_MCLK_SCK_SD_FS_GPIO_PORT GPIOF

Definition at line **107** of file **stm324x9i_eval_audio.h**.

Referenced by **SAIx_MspInit()**.

#define AUDIO_SAIx_SCK_PIN GPIO_PIN_8

Definition at line **104** of file **stm324x9i_eval_audio.h**.

Referenced by **SAIx_MspInit()**.

#define AUDIO_SAIx_SD_PIN GPIO_PIN_6

Definition at line 105 of file stm324x9i_eval_audio.h.

Referenced by SAIx_MspInit().

#define AUDIO_TIMEOUT 2

Definition at line 179 of file stm324x9i_eval_audio.h.

#define AUDIO_TIMx TIM3

Definition at line 161 of file stm324x9i_eval_audio.h.

Referenced by TIMx_Init().

#define AUDIO_TIMx_AF GPIO_AF2_TIM3

Definition at line 168 of file stm324x9i_eval_audio.h.

Referenced by TIMx_IC_MspInit().

#define AUDIO_TIMx_CLK_DISABLE ( ) __TIM3_CLK_DISABLE()

Definition at line 160 of file stm324x9i_eval_audio.h.

Referenced by BSP_AUDIO_IN_Stop().

#define AUDIO_TIMx_CLK_ENABLE ( ) __TIM3_CLK_ENABLE()

Definition at line 159 of file stm324x9i_eval_audio.h.

Referenced by TIMx_IC_MspInit().

#define AUDIO_TIMx_GPIO GPIOC

Definition at line 165 of file stm324x9i_eval_audio.h.

Referenced by TIMx_IC_MspInit().

#define AUDIO_TIMx_GPIO_CLK_ENABLE ( ) __GPIOC_CLK_EN

Definition at line 164 of file stm324x9i_eval_audio.h.

Referenced by TIMx_IC_MspInit().

#define AUDIO_TIMx_IN_CHANNEL TIM_CHANNEL_1

Definition at line 162 of file stm324x9i_eval_audio.h.

Referenced by TIMx_Init().

#define AUDIO_TIMx_IN_GPIO_PIN GPIO_PIN_6

Definition at line 166 of file stm324x9i_eval_audio.h.

Referenced by TIMx_IC_MspInit().

#define AUDIO_TIMx_OUT_CHANNEL TIM_CHANNEL_2 /* Select

Definition at line 163 of file stm324x9i_eval_audio.h.

Referenced by TIMx_Init().

#define AUDIO_TIMx_OUT_GPIO_PIN GPIO_PIN_7

Definition at line 167 of file stm324x9i_eval_audio.h.

Referenced by TIMx_IC_MspInit().

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

Definition at line 174 of file stm324x9i_eval_audio.h.

Referenced by BSP_AUDIO_OUT_Play().

#define CHANNEL_DEMUX_MASK 0x55

Definition at line 191 of file stm324x9i_eval_audio.h.

Referenced by BSP_AUDIO_IN_PDMToPCM().

#define CODEC_RESET_DELAY 5

Definition at line 198 of file stm324x9i_eval_audio.h.

#define DEFAULT_AUDIO_IN_BIT_RESOLUTION 16

Definition at line 183 of file stm324x9i_eval_audio.h.

#define DEFAULT_AUDIO_IN_CHANNEL_NBR 2 /* Mono = 1, Stere

Definition at line 184 of file stm324x9i_eval_audio.h.

Referenced by BSP_AUDIO_IN_PDMToPCM().

#define DEFAULT_AUDIO_IN_FREQ I2S_AUDIOFREQ_16K

Definition at line 182 of file stm324x9i_eval_audio.h.

#define DEFAULT_AUDIO_IN_VOLUME 64

Definition at line 185 of file stm324x9i_eval_audio.h.

#define DMA_MAX_SZE 0xFFFF

Definition at line 117 of file stm324x9i_eval_audio.h.

#define INTERNAL_BUFF_SIZE 128*DEFAULT_AUDIO_IN_FREQ/1

Definition at line 188 of file stm324x9i_eval_audio.h.

Referenced by BSP_AUDIO_IN_PDMToPCM().

#define PCM_OUT_SIZE DEFAULT_AUDIO_IN_FREQ/1000*2

Definition at line 190 of file stm324x9i_eval_audio.h.

STM324x9I_EVAL BSP User Manual

Defines

STM324x9I EVAL LOW LEVEL COM

STM324x9I EVAL LOW LEVEL Exported Constants

Defines

#define	**COMn** 1
#define	**EVAL_COM1** USART1 Definition for COM port1, connected to USART1.
#define	**EVAL_COM1_CLK_ENABLE**() __USART1_CLK_ENABLE()
#define	**EVAL_COM1_CLK_DISABLE**() __USART1_CLK_DISABLE(
#define	**EVAL_COM1_TX_PIN** GPIO_PIN_9
#define	**EVAL_COM1_TX_GPIO_PORT** GPIOA
#define	**EVAL_COM1_TX_GPIO_CLK_ENABLE**() __GPIOA_CLK_E
#define	**EVAL_COM1_TX_GPIO_CLK_DISABLE**() __GPIOA_CLK_D
#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**() __GPIOA_CLK_E
#define	**EVAL_COM1_RX_GPIO_CLK_DISABLE**() __GPIOA_CLK_D
#define	**EVAL_COM1_RX_AF** GPIO_AF7_USART1
#define	**EVAL_COM1_IRQn** USART1_IRQn
#define	**EVAL_COMx_CLK_ENABLE**(__INDEX__)
#define	**EVAL_COMx_CLK_DISABLE**(__INDEX__)
#define	**EVAL_COMx_TX_GPIO_CLK_ENABLE**(__INDEX__)
#define	**EVAL_COMx_TX_GPIO_CLK_DISABLE**(__INDEX__)
#define	**EVAL_COMx_RX_GPIO_CLK_ENABLE**(__INDEX__)
#define	**EVAL_COMx_RX_GPIO_CLK_DISABLE**(__INDEX__)
#define	**JOY_SEL_PIN** **IO_PIN_14** Joystick Pins definition.
#define	**JOY_DOWN_PIN** **IO_PIN_13**
#define	**JOY_LEFT_PIN** **IO_PIN_12**
#define	**JOY_RIGHT_PIN** **IO_PIN_11**
#define	**JOY_UP_PIN** **IO_PIN_10**
#define	**JOY_NONE_PIN** **JOY_ALL_PINS**
#define	**JOY_ALL_PINS** (**IO_PIN_10** | **IO_PIN_11** | **IO_PIN_12** | **IO_P** **IO_PIN_14**)

#define **XSDN_PIN** **IO_PIN_0**
Eval Pins definition.

#define **MII_INT_PIN** **IO_PIN_1**

#define **RSTI_PIN** **IO_PIN_2**

#define **CAM_PLUG_PIN** **IO_PIN_3**

#define **LCD_INT_PIN** **IO_PIN_4**

#define **AUDIO_INT_PIN** **IO_PIN_5**

#define **OTG_FS1_OVER_CURRENT_PIN** **IO_PIN_6**

#define **OTG_FS1_POWER_SWITCH_PIN** **IO_PIN_7**

#define **OTG_FS2_OVER_CURRENT_PIN** **IO_PIN_8**

#define **OTG_FS2_POWER_SWITCH_PIN** **IO_PIN_9**

#define **SD_DETECT_PIN** **IO_PIN_15**

#define **IO_I2C_ADDRESS** 0x84

#define **TS_I2C_ADDRESS** 0x82

#define **TS3510_I2C_ADDRESS** 0x80

#define **EXC7200_I2C_ADDRESS** 0x08

#define **CAMERA_I2C_ADDRESS** 0x60

#define **AUDIO_I2C_ADDRESS** 0x34

#define **EEPROM_I2C_ADDRESS_A01** 0xA0

#define **EEPROM_I2C_ADDRESS_A02** 0xA6

#define **EVAL_I2Cx** I2C1

#define **EVAL_I2Cx_CLK_ENABLE**() __I2C1_CLK_ENABLE()

#define **EVAL_DMAx_CLK_ENABLE**() __DMA1_CLK_ENABLE()

#define **EVAL_I2Cx_SCL_SDA_GPIO_CLK_ENABLE**() __GPIOB_C

#define **EVAL_I2Cx_FORCE_RESET**() __I2C1_FORCE_RESET()

#define **EVAL_I2Cx_RELEASE_RESET**() __I2C1_RELEASE_RESE

#define **EVAL_I2Cx_SCL_PIN** GPIO_PIN_6

#define **EVAL_I2Cx_SCL_SDA_GPIO_PORT** GPIOB

#define **EVAL_I2Cx_SCL_SDA_AF** GPIO_AF4_I2C1

#define **EVAL_I2Cx_SDA_PIN** GPIO_PIN_9

#define **EVAL_I2Cx_EV_IRQn** I2C1_EV_IRQn

#define **EVAL_I2Cx_ER_IRQn** I2C1_ER_IRQn

Define Documentation

#define AUDIO_I2C_ADDRESS 0x34

Definition at line 282 of file stm324x9i_eval.h.

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

#define AUDIO_INT_PIN IO_PIN_5

Definition at line 269 of file stm324x9i_eval.h.

#define CAM_PLUG_PIN IO_PIN_3

Definition at line 267 of file stm324x9i_eval.h.

Referenced by BSP_CAMERA_Init().

#define CAMERA_I2C_ADDRESS 0x60

Definition at line 281 of file stm324x9i_eval.h.

Referenced by BSP_CAMERA_BlackWhiteConfig(), BSP_CAMERA_ColorEffectConfig(), BSP_CAMERA_ContrastBrightnessConfig(), and BSP_CAMERA_Init().

#define COMn 1

Definition at line 212 of file stm324x9i_eval.h.

#define EEPROM_I2C_ADDRESS_A01 0xA0

Definition at line 283 of file stm324x9i_eval.h.

Referenced by BSP_EEPROM_Init().

#define EEPROM_I2C_ADDRESS_A02 0xA6

Definition at line 284 of file stm324x9i_eval.h.

Referenced by BSP_EEPROM_Init().

#define EVAL_COM1 USART1

Definition for COM port1, connected to USART1.

Definition at line 217 of file stm324x9i_eval.h.

#define EVAL_COM1_CLK_DISABLE ( ) __USART1_CLK_DISABL

Definition at line 219 of file stm324x9i_eval.h.

#define EVAL_COM1_CLK_ENABLE ( ) __USART1_CLK_ENABLE

Definition at line 218 of file stm324x9i_eval.h.

#define EVAL_COM1_IRQn USART1_IRQn

Definition at line 233 of file stm324x9i_eval.h.

#define EVAL_COM1_RX_AF GPIO_AF7_USART1

Definition at line 231 of file stm324x9i_eval.h.

#define EVAL_COM1_RX_GPIO_CLK_DISABLE ( ) __GPIOA_CLK

Definition at line 230 of file stm324x9i_eval.h.

#define EVAL_COM1_RX_GPIO_CLK_ENABLE ( ) __GPIOA_CLK_

Definition at line 229 of file stm324x9i_eval.h.

#define EVAL_COM1_RX_GPIO_PORT GPIOA

Definition at line 228 of file stm324x9i_eval.h.

#define EVAL_COM1_RX_PIN GPIO_PIN_10

Definition at line 227 of file stm324x9i_eval.h.

#define EVAL_COM1_TX_AF GPIO_AF7_USART1

Definition at line 225 of file stm324x9i_eval.h.

#define EVAL_COM1_TX_GPIO_CLK_DISABLE ( ) __GPIOA_CLK

Definition at line 224 of file stm324x9i_eval.h.

#define EVAL_COM1_TX_GPIO_CLK_ENABLE ( ) __GPIOA_CLK_

Definition at line 223 of file stm324x9i_eval.h.

#define EVAL_COM1_TX_GPIO_PORT GPIOA

Definition at line 222 of file stm324x9i_eval.h.

#define EVAL_COM1_TX_PIN GPIO_PIN_9

Definition at line 221 of file stm324x9i_eval.h.

#define EVAL_COMx_CLK_DISABLE (__INDEX__)

Value:

```
do{if((__INDEX__) == 0) EVAL_COM1_CLK_DISABLE();
\

      }while(0)
```

Definition at line 237 of file stm324x9i_eval.h.

#define EVAL_COMx_CLK_ENABLE (__INDEX__)

Value:

```
do{if((__INDEX__) == 0) EVAL_COM1_CLK_ENABLE(); \

      }while(0)
```

Definition at line 235 of file stm324x9i_eval.h.

Referenced by BSP_COM_Init().

#define EVAL_COMx_RX_GPIO_CLK_DISABLE (__INDEX__)

Value:

```
do{if((__INDEX__) == 0) EVAL_COM1_RX_GPIO_CLK_DIS
ABLE(); \

      }while(0)
```

Definition at line **247** of file **stm324x9i_eval.h**.

#define EVAL_COMx_RX_GPIO_CLK_ENABLE (__INDEX__)

Value:

```
do{if((__INDEX__) == 0) EVAL_COM1_RX_GPIO_CLK_ENA
BLE(); \

      }while(0)
```

Definition at line **245** of file **stm324x9i_eval.h**.

Referenced by **BSP_COM_Init()**.

#define EVAL_COMx_TX_GPIO_CLK_DISABLE (__INDEX__)

Value:

```
do{if((__INDEX__) == 0) EVAL_COM1_TX_GPIO_CLK_DIS
ABLE(); \

    }while(0)
```

Definition at line **242** of file **stm324x9i_eval.h**.

#define EVAL_COMx_TX_GPIO_CLK_ENABLE (__INDEX__)

Value:

```
do{if((__INDEX__) == 0) EVAL_COM1_TX_GPIO_CLK_ENABLE(); \

    }while(0)
```

Definition at line **240** of file **stm324x9i_eval.h**.

Referenced by **BSP_COM_Init()**.

#define EVAL_DMAx_CLK_ENABLE ( ) __DMA1_CLK_ENABLE()

Definition at line **298** of file **stm324x9i_eval.h**.

#define EVAL_I2Cx I2C1

Definition at line **296** of file **stm324x9i_eval.h**.

#define EVAL_I2Cx_CLK_ENABLE ( ) __I2C1_CLK_ENABLE()

Definition at line **297** of file **stm324x9i_eval.h**.

Referenced by **I2Cx_MspInit()**.

#define EVAL_I2Cx_ER_IRQn I2C1_ER_IRQn

Definition at line **312** of file **stm324x9i_eval.h**.

Referenced by **I2Cx_MspInit()**.

#define EVAL_I2Cx_EV_IRQn I2C1_EV_IRQn

Definition at line **311** of file **stm324x9i_eval.h**.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_FORCE_RESET ( ) __I2C1_FORCE_RESET()

Definition at line 301 of file stm324x9i_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_RELEASE_RESET ( ) __I2C1_RELEASE_RES

Definition at line 302 of file stm324x9i_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SCL_PIN GPIO_PIN_6

Definition at line 305 of file stm324x9i_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SCL_SDA_AF GPIO_AF4_I2C1

Definition at line 307 of file stm324x9i_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SCL_SDA_GPIO_CLK_ENABLE ( ) __GPIOB_

Definition at line 299 of file stm324x9i_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SCL_SDA_GPIO_PORT GPIOB

Definition at line 306 of file stm324x9i_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SDA_PIN GPIO_PIN_9

Definition at line 308 of file stm324x9i_eval.h.

Referenced by I2Cx_MspInit().

#define EXC7200_I2C_ADDRESS 0x08

Definition at line 280 of file stm324x9i_eval.h.

Referenced by BSP_TS_Init(), and I2Cx_ReadMultiple().

#define IO_I2C_ADDRESS 0x84

Definition at line 277 of file stm324x9i_eval.h.

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

#define JOY_ALL_PINS (IO_PIN_10 | IO_PIN_11 | IO_PIN_12 | IO_P

Definition at line 259 of file stm324x9i_eval.h.

Referenced by BSP_JOY_GetState(), and BSP_JOY_Init().

#define JOY_DOWN_PIN IO_PIN_13

Definition at line 254 of file stm324x9i_eval.h.

Referenced by BSP_JOY_GetState().

#define JOY_LEFT_PIN IO_PIN_12

Definition at line 255 of file stm324x9i_eval.h.

Referenced by BSP_JOY_GetState().

#define JOY_NONE_PIN JOY_ALL_PINS

Definition at line 258 of file stm324x9i_eval.h.

Referenced by BSP_JOY_GetState().

#define JOY_RIGHT_PIN IO_PIN_11

Definition at line 256 of file stm324x9i_eval.h.

Referenced by BSP_JOY_GetState().

#define JOY_SEL_PIN IO_PIN_14

Joystick Pins definition.

Definition at line 253 of file stm324x9i_eval.h.

Referenced by BSP_JOY_GetState().

#define JOY_UP_PIN IO_PIN_10

Definition at line **257** of file **stm324x9i_eval.h**.

Referenced by **BSP_JOY_GetState()**.

#define LCD_INT_PIN IO_PIN_4

Definition at line **268** of file **stm324x9i_eval.h**.

#define MII_INT_PIN IO_PIN_1

Definition at line **265** of file **stm324x9i_eval.h**.

#define OTG_FS1_OVER_CURRENT_PIN IO_PIN_6

Definition at line **270** of file **stm324x9i_eval.h**.

#define OTG_FS1_POWER_SWITCH_PIN IO_PIN_7

Definition at line **271** of file **stm324x9i_eval.h**.

#define OTG_FS2_OVER_CURRENT_PIN IO_PIN_8

Definition at line **272** of file **stm324x9i_eval.h**.

#define OTG_FS2_POWER_SWITCH_PIN IO_PIN_9

Definition at line **273** of file **stm324x9i_eval.h**.

#define RSTI_PIN IO_PIN_2

Definition at line **266** of file **stm324x9i_eval.h**.

#define SD_DETECT_PIN IO_PIN_15

Definition at line **274** of file **stm324x9i_eval.h**.

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

#define TS3510_I2C_ADDRESS 0x80

Definition at line **279** of file **stm324x9i_eval.h**.

Referenced by **BSP_TS3510_IsDetected()**, and **BSP_TS_Init()**.

#define TS_I2C_ADDRESS 0x82

Definition at line **278** of file **stm324x9i_eval.h**.

Referenced by **BSP_LCD_ClockConfig()**, **BSP_LCD_InitEx()**, and **BSP_TS_Init()**.

#define XSDN_PIN IO_PIN_0

Eval Pins definition.

Definition at line **264** of file **stm324x9i_eval.h**.

Referenced by **BSP_CAMERA_Init()**, and **BSP_CAMERA_Stop()**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL LOW LEVEL Private FunctionPrototypes

STM324x9I EVAL LOW LEVEL

Functions

static void	**I2Cx_MspInit** (void) Initializes I2C MSP.
static void	**I2Cx_Init** (void) Initializes I2C HAL.
static void	**I2Cx_ITConfig** (void) Configures I2C Interrupt.
void	**IOE_Init** (void) Initializes IOE low level.
void	**IOE_ITConfig** (void) Configures IOE low level interrupt.
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 789 of file stm324x9i_eval.c.

void AUDIO_IO_Init (void)

Initializes Audio low level.

Definition at line 781 of file stm324x9i_eval.c.

References I2Cx_Init().

void CAMERA_IO_Init (void)

Initializes Camera low level.

Definition at line 846 of file stm324x9i_eval.c.

References I2Cx_Init().

void EEPROM_IO_Init (void)

Initializes peripherals used by the I2C EEPROM driver.

Definition at line 887 of file stm324x9i_eval.c.

References I2Cx_Init().

Referenced by BSP_EEPROM_Init().

static void I2Cx_Init (void) [static]

Initializes I2C HAL.

Definition at line **529** of file **stm324x9i_eval.c**.

References **heval_I2c**, and **I2Cx_MspInit()**.

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

static void I2Cx_ITConfig (void) [static]

Configures I2C Interrupt.

Definition at line **552** of file **stm324x9i_eval.c**.

Referenced by **IOE_ITConfig()**.

static void I2Cx_MspInit (void) [static]

Initializes I2C MSP.

Definition at line **487** of file **stm324x9i_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 IOE_Init (void)

Initializes IOE low level.

Definition at line **707** of file **stm324x9i_eval.c**.

References **I2Cx_Init()**.

Referenced by **BSP_TS_Init()**.

void IOE_ITConfig (void)

Configures IOE low level interrupt.

Definition at line **715** of file **stm324x9i_eval.c**.

References **I2Cx_ITConfig()**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL LOW LEVEL Private Functions

STM324x9I EVAL LOW LEVEL

Functions

uint32_t	**BSP_GetVersion** (void) This method returns the STM324x9I EVAL BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LED GPIO.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** Button_Mode) Configures button GPIO and EXTI Line.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) Configures COM port.
uint8_t	**BSP_JOY_Init** (**JOYMode_TypeDef** Joy_Mode) Configures joystick GPIO and EXTI modes.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the current joystick status.
uint8_t	**BSP_TS3510_IsDetected** (void) Check TS3510 touch screen presence.
static void	**I2Cx_Write** (uint8_t Addr, uint8_t Reg, uint8_t Value) Writes a single data.

static uint8_t	**I2Cx_Read** (uint8_t Addr, uint8_t Reg) Reads a single data.
static HAL_StatusTypeDef	**I2Cx_ReadMultiple** (uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t *Buffer, uint16_t Length) Reads multiple data.
static HAL_StatusTypeDef	**I2Cx_WriteMultiple** (uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t *Buffer, uint16_t Length) Writes a value in a register of the device through BUS in using DMA mode.
static HAL_StatusTypeDef	**I2Cx_IsDeviceReady** (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.
void	**IOE_Write** (uint8_t Addr, uint8_t Reg, uint8_t Value) IOE writes single data.
uint8_t	**IOE_Read** (uint8_t Addr, uint8_t Reg) IOE reads single data.
uint16_t	**IOE_ReadMultiple** (uint8_t Addr, uint8_t Reg, uint8_t *Buffer, uint16_t Length) IOE reads multiple data.
void	**IOE_WriteMultiple** (uint8_t Addr, uint8_t Reg, uint8_t *Buffer, uint16_t Length) IOE writes multiple data.
void	**IOE_Delay** (uint32_t Delay) IOE delay.
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_Write** (uint8_t Addr, uint8_t Reg, uint8_t Value) Camera writes single data.
uint8_t	**CAMERA_IO_Read** (uint8_t Addr, uint8_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 **836** of file **stm324x9i_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 **817** of file **stm324x9i_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 **800** of file **stm324x9i_eval.c**.

References **I2Cx_WriteMultiple()**.

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 **342** of file **stm324x9i_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 STM324x9I EVAL BSP Driver revision.

Return values:

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

Definition at line **192** of file **stm324x9i_eval.c**.

References **__STM324x9I_EVAL_BSP_VERSION**.

JOYState_TypeDef BSP_JOY_GetState (void)

Returns the current joystick status.

Return values:

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

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

Definition at line **409** of file **stm324x9i_eval.c**.

References **BSP_IO_ReadPin()**, **JOY_ALL_PINS**, **JOY_DOWN**, **JOY_DOWN_PIN**, **JOY_LEFT**, **JOY_LEFT_PIN**, **JOY_NONE**, **JOY_NONE_PIN**, **JOY_RIGHT**, **JOY_RIGHT_PIN**, **JOY_SEL**, **JOY_SEL_PIN**, **JOY_UP**, and **JOY_UP_PIN**.

uint8_t BSP_JOY_Init (JOYMode_TypeDef Joy_Mode)

Configures joystick GPIO and EXTI modes.

Parameters:

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

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

Return values:

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

Definition at line **381** of file **stm324x9i_eval.c**.

References **BSP_IO_ConfigPin()**, **BSP_IO_Init()**, **JOY_ALL_PINS**, and **JOY_MODE_EXTI**.

void BSP_LED_Init (Led_TypeDef Led)

Configures LED GPIO.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Definition at line **206** of file **stm324x9i_eval.c**.

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

void BSP_LED_Off (Led_TypeDef Led)

Turns selected LED Off.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Definition at line **247** of file **stm324x9i_eval.c**.

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

void BSP_LED_On (Led_TypeDef Led)

Turns selected LED On.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Definition at line **233** of file **stm324x9i_eval.c**.

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

void BSP_LED_Toggle (Led_TypeDef Led)

Toggles the selected LED.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Definition at line **261** of file **stm324x9i_eval.c**.

References **GPIO_PIN**, and **GPIO_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

Return values:

The Button GPIO pin value

Definition at line **328** of file **stm324x9i_eval.c**.

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

void BSP_PB_Init (Button_TypeDef Button,
ButtonMode_TypeDef Button_Mode
)

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_Mode,: Button mode This parameter can be one of the following values:
- BUTTON_MODE_GPIO: Button will be used as simple IO
- BUTTON_MODE_EXTI: Button will be connected to EXTI line with interrupt generation capability

Definition at line **278** of file **stm324x9i_eval.c**.

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

uint8_t BSP_TS3510_IsDetected (void)

Check TS3510 touch screen presence.

Return values:

Return 0 if TS3510 is detected, return 1 if not detected

Definition at line **451** of file **stm324x9i_eval.c**.

References **heval_I2c**, **I2Cx_Error()**, **IOE_WriteMultiple()**, and **TS3510_I2C_ADDRESS**.

Referenced by **BSP_TS_Init()**.

void CAMERA_Delay (uint32_t Delay)

Camera delay.

Parameters:

Delay,: Delay in ms

Definition at line **877** of file **stm324x9i_eval.c**.

uint8_t CAMERA_IO_Read (uint8_t Addr,
uint8_t Reg
)

Camera reads single data.

Parameters:

Addr,: I2C address

Reg,: Register address

Return values:

Read data

Definition at line **868** of file **stm324x9i_eval.c**.

References **I2Cx_Read()**.

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

Camera writes single data.

Parameters:

Addr,: I2C address

Reg,: Register address

Value,: Data to be written

Definition at line **857** of file **stm324x9i_eval.c**.

References **I2Cx_Write()**.

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 **925** of file **stm324x9i_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 **913** of file **stm324x9i_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 **900** of file **stm324x9i_eval.c**.

References I2Cx_WriteMultiple().

Referenced by BSP_EEPROM_WritePage().

static void I2Cx_Error (uint8_t Addr) [static]

Manages error callback by re-initializing I2C.

Parameters:

Addr,: I2C Address

Definition at line 689 of file stm324x9i_eval.c.

References heval_I2c, and I2Cx_Init().

Referenced by BSP_TS3510_IsDetected(), I2Cx_Read(), I2Cx_ReadMultiple(), I2Cx_Write(), 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 680 of file stm324x9i_eval.c.

References heval_I2c.

Referenced by EEPROM_IO_IsDeviceReady().

static uint8_t I2Cx_Read (uint8_t Addr,
uint8_t Reg
) [static]

Reads a single data.

Parameters:

Addr,: I2C address

Reg,: Register address

Return values:

Read data

Definition at line 602 of file stm324x9i_eval.c.

References heval_I2c, and I2Cx_Error().

Referenced by CAMERA_IO_Read(), and IOE_Read().

static HAL_StatusTypeDef I2Cx_ReadMultiple (uint8_t Addr,
uint16_t Reg,
uint16_t MemAddr
uint8_t * Buffer,
uint16_t Length
) [static]

Reads multiple data.

Parameters:

Addr,: I2C address

Reg,: Reg address
MemAddress,: Internal memory address
Buffer,: Pointer to data buffer
Length,: Length of the data

Return values:
Number of read data

Definition at line **627** of file **stm324x9i_eval.c**.

References **EXC7200_I2C_ADDRESS**, **heval_I2c**, and **I2Cx_Error()**.

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

static void I2Cx_Write (uint8_t Addr,
uint8_t Reg,
uint8_t Value
) `[static]`

Writes a single data.

Parameters:
Addr,: I2C address
Reg,: Register address
Value,: Data to be written

Definition at line **582** of file **stm324x9i_eval.c**.

References **heval_I2c**, and **I2Cx_Error()**.

Referenced by **CAMERA_IO_Write()**, and **IOE_Write()**.

static HAL_StatusTypeDef I2Cx_WriteMultiple (uint8_t Addr,

```
                                    uint16_t  Reg,
                                    uint16_t  MemAddr
                                    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
MemAddress,:	Internal memory address
Buffer,:	The target register value to be written
Length,:	buffer size to be written

Return values:

HAL status

Definition at line **658** of file **stm324x9i_eval.c**.

References **heval_I2c**, and **I2Cx_Error()**.

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

void IOE_Delay (uint32_t Delay)

IOE delay.

Parameters:

Delay,: Delay in ms

Definition at line **771** of file **stm324x9i_eval.c**.

```
uint8_t IOE_Read ( uint8_t  Addr,
                   uint8_t  Reg
                 )
```

IOE reads single data.

Parameters:

Addr,: I2C address

Reg,: Register address

Return values:

Read data

Definition at line **737** of file **stm324x9i_eval.c**.

References **I2Cx_Read()**.

```
uint16_t IOE_ReadMultiple ( uint8_t    Addr,
                            uint8_t    Reg,
                            uint8_t *  Buffer,
                            uint16_t   Length
                          )
```

IOE reads multiple data.

Parameters:

Addr,: I2C address

Reg,: Register address

Buffer,: Pointer to data buffer

Length,: Length of the data

Return values:

Number of read data

Definition at line 750 of file stm324x9i_eval.c.

References I2Cx_ReadMultiple().

```
void IOE_Write ( uint8_t  Addr,
                 uint8_t  Reg,
                 uint8_t  Value
               )
```

IOE writes single data.

Parameters:

Addr,:	I2C address
Reg,:	Register address
Value,:	Data to be written

Definition at line 726 of file stm324x9i_eval.c.

References I2Cx_Write().

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

IOE writes multiple data.

Parameters:

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

Definition at line **762** of file **stm324x9i_eval.c**.

References **I2Cx_WriteMultiple()**.

Referenced by **BSP_TS3510_IsDetected()**.

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL AUDIO Exported Variables

STM324x9I EVAL AUDIO

Variables

__IO uint16_t	**AudioInVolume**

Variable Documentation

__IO uint16_t AudioInVolume

Definition at line **162** of file **stm324x9i_eval_audio.c**.

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

STM324x9I_EVAL BSP User Manual

Defines

STM324x9I EVAL NOR Exported Constants

STM324x9I EVAL NOR

Defines

#define	**NOR_DEVICE_ADDR** ((uint32_t)0x60000000)
#define	**NOR_MEMORY_WIDTH** FMC_NORSRAM_MEM_BUS_WID
#define	**NOR_BURSTACCESS** FMC_BURST_ACCESS_MODE_DIS
#define	**NOR_WRITEBURST** FMC_WRITE_BURST_DISABLE
#define	**CONTINUOUSCLOCK_FEATURE** FMC_CONTINUOUS_CL
#define	**BLOCKERASE_TIMEOUT** ((uint32_t)0x00A00000) /* NOR b */
#define	**CHIPERASE_TIMEOUT** ((uint32_t)0x30000000) /* NOR chip
#define	**PROGRAM_TIMEOUT** ((uint32_t)0x00004400) /* NOR progra
#define	**NOR_READY_BUSY_PIN** GPIO_PIN_6
#define	**NOR_READY_BUSY_GPIO** GPIOD
#define	**NOR_READY_STATE** GPIO_PIN_SET
#define	**NOR_BUSY_STATE** GPIO_PIN_RESET

Define Documentation

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

Definition at line 93 of file stm324x9i_eval_nor.h.

Referenced by BSP_NOR_Erase_Block().

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

Definition at line 94 of file stm324x9i_eval_nor.h.

Referenced by BSP_NOR_Erase_Chip().

#define CONTINUOUSCLOCK_FEATURE FMC_CONTINUOUS_CL

Definition at line 89 of file stm324x9i_eval_nor.h.

Referenced by BSP_NOR_Init(), and BSP_SRAM_Init().

#define NOR_BURSTACCESS FMC_BURST_ACCESS_MODE_DIS

Definition at line 83 of file stm324x9i_eval_nor.h.

Referenced by BSP_NOR_Init().

#define NOR_BUSY_STATE GPIO_PIN_RESET

Definition at line 101 of file stm324x9i_eval_nor.h.

Referenced by HAL_NOR_MspWait().

#define NOR_DEVICE_ADDR ((uint32_t)0x60000000)

Definition at line 78 of file stm324x9i_eval_nor.h.

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

#define NOR_MEMORY_WIDTH FMC_NORSRAM_MEM_BUS_WID

Definition at line 81 of file stm324x9i_eval_nor.h.

Referenced by BSP_NOR_Init().

#define NOR_READY_BUSY_GPIO GPIOD

Definition at line 99 of file stm324x9i_eval_nor.h.

Referenced by HAL_NOR_MspWait().

#define NOR_READY_BUSY_PIN GPIO_PIN_6

Definition at line 98 of file stm324x9i_eval_nor.h.

Referenced by HAL_NOR_MspWait().

#define NOR_READY_STATE GPIO_PIN_SET

Definition at line 100 of file stm324x9i_eval_nor.h.

Referenced by HAL_NOR_MspWait().

#define NOR_WRITEBURST FMC_WRITE_BURST_DISABLE

Definition at line 86 of file stm324x9i_eval_nor.h.

Referenced by BSP_NOR_Init().

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

Definition at line 95 of file stm324x9i_eval_nor.h.

Referenced by BSP_NOR_ProgramData(), and BSP_NOR_WriteData().

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL
AUDIO OUT Private Functions

STM324x9I 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	**HAL_SAI_TxCpltCallback** (SAI_HandleTypeDef *hsai) Tx Transfer completed callbacks.
	HAL_SAI_TxHalfCpltCallback (SAI_HandleTypeDef

void	*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.
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.
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.
static void	**SAIx_Init** (uint32_t AudioFreq) Initializes the Audio Codec audio interface (SAI).
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_Init** (void) Configure TIM as a clock divider by 2.

Function Documentation

__weak void BSP_AUDIO_IN_Error_Callback (void)

Audio IN Error callback function.

Definition at line 870 of file stm324x9i_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 860 of file stm324x9i_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 **681** of file **stm324x9i_eval_audio.c**.

References **AUDIO_OK**, **I2Sx_Init()**, **PDMDecoder_Init()**, and **TIMx_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 **747** of file **stm324x9i_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 **790** of file **stm324x9i_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 **710** of file **stm324x9i_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 **760** of file **stm324x9i_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 **775** of file **stm324x9i_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 **727** of file **stm324x9i_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 **850** of file **stm324x9i_eval_audio.c**.

Referenced by **HAL_I2S_RxCpltCallback()**.

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 **stm324x9i_eval_audio.c**.

References **haudio_out_sai**.

__weak void BSP_AUDIO_OUT_Error_CallBack (void)

Manages the DMA FIFO error event.

Definition at line **543** of file **stm324x9i_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 **536** of file **stm324x9i_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 **195** of file **stm324x9i_eval_audio.c**.

References **audio_drv**, **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, and **SAIx_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 **stm324x9i_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 **stm324x9i_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 **stm324x9i_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 **CODEC AudioFrame SLOT TDMMode**

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

Definition at line **482** of file **stm324x9i_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 **stm324x9i_eval_audio.c**.

References **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 **stm324x9i_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 **stm324x9i_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 **stm324x9i_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 **stm324x9i_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 **529** of file **stm324x9i_eval_audio.c**.

Referenced by **HAL_SAI_TxCpltCallback()**.

void HAL_I2S_ErrorCallback (I2S_HandleTypeDef * hi2s)

I2S error callbacks.

Parameters:

hi2s,: I2S handle

Definition at line **840** of file **stm324x9i_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 **819** of file **stm324x9i_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 **829** of file **stm324x9i_eval_audio.c**.

References **BSP_AUDIO_IN_HalfTransfer_CallBack()**.

void HAL_SAI_ErrorCallback (SAI_HandleTypeDef * hsai)

SAI error callbacks.

Parameters:

hsai,: SAI handle

Definition at line **521** of file **stm324x9i_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 **499** of file **stm324x9i_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 **510** of file **stm324x9i_eval_audio.c**.

References **BSP_AUDIO_OUT_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 **974** of file **stm324x9i_eval_audio.c**.

References **AUDIO_I2Sx**, **haudio_in_i2s**, and **I2Sx_MspInit()**.

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 **885** of file **stm324x9i_eval_audio.c**.

References **Filter**.

Referenced by **BSP_AUDIO_IN_Init()**.

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 **616** of file **stm324x9i_eval_audio.c**.

References **AUDIO_SAIx**, **CODEC_AUDIOFRAME_SLOT_0123**, **haudio_out_sai**, and **SAIx_MspInit()**.

Referenced by **BSP_AUDIO_OUT_Init()**.

static void TIMx_IC_MspInit (TIM_HandleTypeDef * htim) [static]

Initializes the TIM INput Capture MSP.

Parameters:

htim,: TIM handle

Definition at line **1004** of file **stm324x9i_eval_audio.c**.

References **AUDIO_TIMx_AF**, **AUDIO_TIMx_CLK_ENABLE**, **AUDIO_TIMx_GPIO**, **AUDIO_TIMx_GPIO_CLK_ENABLE**, **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 **1035** of file **stm324x9i_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_Init()**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL
AUDIO IN Exported Functions

STM324x9I 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.
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.

Function Documentation

void BSP_AUDIO_IN_Error_Callback (void)

Audio IN Error callback function.

Definition at line 870 of file stm324x9i_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 860 of file stm324x9i_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 **681** of file **stm324x9i_eval_audio.c**.

References **AUDIO_OK**, **I2Sx_Init()**, **PDMDecoder_Init()**, and **TIMx_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 **747** of file **stm324x9i_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 **790** of file **stm324x9i_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 **710** of file **stm324x9i_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 **760** of file **stm324x9i_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 **775** of file **stm324x9i_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 **727** of file **stm324x9i_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 **850** of file **stm324x9i_eval_audio.c**.

Referenced by **HAL_I2S_RxCpltCallback()**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL
AUDIO OUT Exported Functions

STM324x9I 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_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.

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 **stm324x9i_eval_audio.c**.

References **haudio_out_sai**.

void BSP_AUDIO_OUT_Error_CallBack (void)

Manages the DMA FIFO error event.

Definition at line **543** of file **stm324x9i_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 **536** of file **stm324x9i_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 **195** of file **stm324x9i_eval_audio.c**.

References **audio_drv**, **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, and **SAIx_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 **stm324x9i_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 **stm324x9i_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 **stm324x9i_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 **CODEC AudioFrame SLOT TDMMode**

Note:

This API should be called after the **BSP_AUDIO_OUT_Init()** to adjust the audio frame slot.

Definition at line **482** of file **stm324x9i_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 **stm324x9i_eval_audio.c**.

References **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 **stm324x9i_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 **stm324x9i_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 **stm324x9i_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 **stm324x9i_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 **529** of file **stm324x9i_eval_audio.c**.

Referenced by **HAL_SAI_TxCpltCallback()**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL CAMERA Private Functions

STM324x9I EVAL CAMERA

Functions

uint8_t	**BSP_CAMERA_Init** (uint32_t Resolution) Initializes 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_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	**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 **288** of file **stm324x9i_eval_camera.c**.

References **camera_drv**, and **CAMERA_I2C_ADDRESS**.

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 **305** of file **stm324x9i_eval_camera.c**.

References **camera_drv**, and **CAMERA_I2C_ADDRESS**.

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 **187** of file **stm324x9i_eval_camera.c**.

References **current_resolution**, **GetSize()**, and **hdcmi_eval**.

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 **271** of file **stm324x9i_eval_camera.c**.

References **camera_drv**, and **CAMERA_I2C_ADDRESS**.

void BSP_CAMERA_DMA_IRQHandler (void)

Handles DMA interrupt request.

Definition at line **324** of file **stm324x9i_eval_camera.c**.

References **hdcmi_eval**.

__weak void BSP_CAMERA_ErrorCallback (void)

Error callback.

Definition at line **541** of file **stm324x9i_eval_camera.c**.

Referenced by **HAL_DCMI_ErrorCallback()**.

__weak void BSP_CAMERA_FrameEventCallback (void)

Frame Event callback.

Definition at line **522** of file **stm324x9i_eval_camera.c**.

Referenced by **HAL_DCMI_FrameEventCallback()**.

uint8_t BSP_CAMERA_Init (uint32_t Resolution)

Initializes the camera.

Parameters:

Resolution,: Camera Resolution

Return values:

Camera status

Definition at line 130 of file stm324x9i_eval_camera.c.

References BSP_IO_ConfigPin(), BSP_IO_Init(), BSP_IO_ReadPin(), BSP_IO_WritePin(), CAM_PLUG_PIN, camera_drv, CAMERA_ERROR, CAMERA_I2C_ADDRESS, CAMERA_OK, current_resolution, DCMI_MspInit(), hdcmi_eval, and XSDN_PIN.

void BSP_CAMERA_IRQHandler (void)

Handles DCMI interrupt request.

Definition at line 316 of file stm324x9i_eval_camera.c.

References hdcmi_eval.

__weak void BSP_CAMERA_LineEventCallback (void)

Line Event callback.

Definition at line 484 of file stm324x9i_eval_camera.c.

Referenced by HAL_DCMI_LineEventCallback().

void BSP_CAMERA_Resume (void)

Resume the CAMERA capture.

Definition at line 218 of file stm324x9i_eval_camera.c.

References hdcmi_eval.

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 197 of file stm324x9i_eval_camera.c.

References current_resolution, GetSize(), and hdcmi_eval.

uint8_t BSP_CAMERA_Stop (void)

Stop the CAMERA capture.

Return values:

Camera status

Definition at line 230 of file stm324x9i_eval_camera.c.

References BSP_IO_ConfigPin(), BSP_IO_Init(), BSP_IO_WritePin(), CAMERA_ERROR, CAMERA_OK, hdcmi_eval, and XSDN_PIN.

void BSP_CAMERA_Suspend (void)

Suspend the CAMERA capture.

Definition at line **206** of file **stm324x9i_eval_camera.c**.

References **hdcmi_eval**.

__weak void BSP_CAMERA_VsyncEventCallback (void)

VSYNC Event callback.

Definition at line **503** of file **stm324x9i_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 **334** of file **stm324x9i_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 **533** of file **stm324x9i_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 **514** of file **stm324x9i_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 **476** of file **stm324x9i_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 **495** of file **stm324x9i_eval_camera.c**.

References **BSP_CAMERA_VsyncEventCallback()**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL CAMERA Exported Functions

STM324x9I EVAL CAMERA

Functions

uint8_t	**BSP_CAMERA_Init** (uint32_t Resolution) Initializes 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_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.

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 **288** of file **stm324x9i_eval_camera.c**.

References **camera_drv**, and **CAMERA_I2C_ADDRESS**.

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 **305** of file **stm324x9i_eval_camera.c**.

References **camera_drv**, and **CAMERA_I2C_ADDRESS**.

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 **187** of file **stm324x9i_eval_camera.c**.

References **current_resolution**, **GetSize()**, and **hdcmi_eval**.

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 **271** of file **stm324x9i_eval_camera.c**.

References **camera_drv**, and **CAMERA_I2C_ADDRESS**.

void BSP_CAMERA_DMA_IRQHandler (void)

Handles DMA interrupt request.

Definition at line **324** of file **stm324x9i_eval_camera.c**.

References **hdcmi_eval**.

void BSP_CAMERA_ErrorCallback (void)

Error callback.

Definition at line **541** of file **stm324x9i_eval_camera.c**.

Referenced by **HAL_DCMI_ErrorCallback()**.

void BSP_CAMERA_FrameEventCallback (void)

Frame Event callback.

Definition at line **522** of file **stm324x9i_eval_camera.c**.

Referenced by **HAL_DCMI_FrameEventCallback()**.

uint8_t BSP_CAMERA_Init (uint32_t Resolution)

Initializes the camera.

Parameters:

Resolution,: Camera Resolution

Return values:

Camera status

Definition at line 130 of file stm324x9i_eval_camera.c.

References BSP_IO_ConfigPin(), BSP_IO_Init(), BSP_IO_ReadPin(), BSP_IO_WritePin(), CAM_PLUG_PIN, camera_drv, CAMERA_ERROR, CAMERA_I2C_ADDRESS, CAMERA_OK, current_resolution, DCMI_MspInit(), hdcmi_eval, and XSDN_PIN.

void BSP_CAMERA_IRQHandler (void)

Handles DCMI interrupt request.

Definition at line 316 of file stm324x9i_eval_camera.c.

References hdcmi_eval.

void BSP_CAMERA_LineEventCallback (void)

Line Event callback.

Definition at line 484 of file stm324x9i_eval_camera.c.

Referenced by HAL_DCMI_LineEventCallback().

void BSP_CAMERA_Resume (void)

Resume the CAMERA capture.

Definition at line 218 of file stm324x9i_eval_camera.c.

References hdcmi_eval.

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 197 of file stm324x9i_eval_camera.c.

References current_resolution, GetSize(), and hdcmi_eval.

uint8_t BSP_CAMERA_Stop (void)

Stop the CAMERA capture.

Return values:

Camera status

Definition at line 230 of file stm324x9i_eval_camera.c.

References BSP_IO_ConfigPin(), BSP_IO_Init(), BSP_IO_WritePin(), CAMERA_ERROR, CAMERA_OK, hdcmi_eval, and XSDN_PIN.

void BSP_CAMERA_Suspend (void)

Suspend the CAMERA capture.

Definition at line **206** of file **stm324x9i_eval_camera.c**.

References **hdcmi_eval**.

void BSP_CAMERA_VsyncEventCallback (void)

VSYNC Event callback.

Definition at line **503** of file **stm324x9i_eval_camera.c**.

Referenced by **HAL_DCMI_VsyncEventCallback()**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL LOW LEVEL Exported Functions

STM324x9I EVAL LOW LEVEL

Functions

uint32_t	**BSP_GetVersion** (void) This method returns the STM324x9I EVAL BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LED GPIO.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** Button_Mode) Configures button GPIO and EXTI Line.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *husart) Configures COM port.
uint8_t	**BSP_JOY_Init** (**JOYMode_TypeDef** Joy_Mode) Configures joystick GPIO and EXTI modes.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the current joystick status.
uint8_t	**BSP_TS3510_IsDetected** (void) Check TS3510 touch screen presence.

Function Documentation

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 **342** of file **stm324x9i_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 STM324x9I EVAL BSP Driver revision.

Return values:

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

Definition at line **192** of file **stm324x9i_eval.c**.

References **__STM324x9I_EVAL_BSP_VERSION**.

JOYState_TypeDef BSP_JOY_GetState (void)

Returns the current joystick status.

Return values:

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

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

Definition at line **409** of file **stm324x9i_eval.c**.

References **BSP_IO_ReadPin()**, **JOY_ALL_PINS**, **JOY_DOWN**, **JOY_DOWN_PIN**, **JOY_LEFT**, **JOY_LEFT_PIN**, **JOY_NONE**, **JOY_NONE_PIN**, **JOY_RIGHT**, **JOY_RIGHT_PIN**, **JOY_SEL**, **JOY_SEL_PIN**, **JOY_UP**, and **JOY_UP_PIN**.

uint8_t BSP_JOY_Init (JOYMode_TypeDef Joy_Mode)

Configures joystick GPIO and EXTI modes.

Parameters:

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

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

Return values:

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

Definition at line **381** of file **stm324x9i_eval.c**.

References **BSP_IO_ConfigPin()**, **BSP_IO_Init()**, **JOY_ALL_PINS**, and **JOY_MODE_EXTI**.

void BSP_LED_Init (Led_TypeDef Led)

Configures LED GPIO.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Definition at line **206** of file **stm324x9i_eval.c**.

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

void BSP_LED_Off (Led_TypeDef Led)

Turns selected LED Off.

Parameters:

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

- LED1
- LED2
- LED3

- LED4

Definition at line **247** of file **stm324x9i_eval.c**.

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

void BSP_LED_On (Led_TypeDef Led)

Turns selected LED On.

Parameters:

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

Definition at line **233** of file **stm324x9i_eval.c**.

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

void BSP_LED_Toggle (Led_TypeDef Led)

Toggles the selected LED.

Parameters:

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

Definition at line **261** of file **stm324x9i_eval.c**.

References **GPIO_PIN**, and **GPIO_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

Return values:

The Button GPIO pin value

Definition at line **328** of file **stm324x9i_eval.c**.

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

void BSP_PB_Init (Button_TypeDef Button, ButtonMode_TypeDef Button_Mode)

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_Mode,: Button mode This parameter can be one of the following values:
- BUTTON_MODE_GPIO: Button will be

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

Definition at line **278** of file **stm324x9i_eval.c**.

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

uint8_t BSP_TS3510_IsDetected (void)

Check TS3510 touch screen presence.

Return values:

Return 0 if TS3510 is detected, return 1 if not detected

Definition at line **451** of file **stm324x9i_eval.c**.

References **heval_I2c**, **I2Cx_Error()**, **IOE_WriteMultiple()**, and **TS3510_I2C_ADDRESS**.

Referenced by **BSP_TS_Init()**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL EEPROM Private Functions

STM324x9I EVAL EEPROM

Functions

uint32_t	**BSP_EEPROM_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
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

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 **156** of file **stm324x9i_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 **190** of file **stm324x9i_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 **444** of file **stm324x9i_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 **429** of file **stm324x9i_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 **272** of file **stm324x9i_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 **237** of file **stm324x9i_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()**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL EEPROM Exported Functions

STM324x9I EVAL EEPROM

Functions

uint32_t	**BSP_EEPROM_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
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

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 **156** of file **stm324x9i_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 **190** of file **stm324x9i_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 **444** of file **stm324x9i_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 **429** of file **stm324x9i_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 **272** of file **stm324x9i_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 **237** of file **stm324x9i_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 **887** of file **stm324x9i_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 **925** of file **stm324x9i_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 **913** of file **stm324x9i_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 **900** of file **stm324x9i_eval.c**.

References **I2Cx_WriteMultiple()**.

Referenced by **BSP_EEPROM_WritePage()**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL IO
Private Functions

STM324x9I EVAL IO

Functions

uint8_t	**BSP_IO_Init** (void) Initializes and configures the IO functionalities and configures all necessary hardware resources (GPIOs, clocks..).
uint8_t	**BSP_IO_ITGetStatus** (uint16_t IO_Pin) Gets the selected pins IT status.
void	**BSP_IO_ITClear** (void) Clears all the IO IT pending bits.
uint8_t	**BSP_IO_ConfigPin** (uint16_t IO_Pin, IO_ModeTypedef IO_Mode) Configures the IO pin(s) according to IO mode structure value.
void	**BSP_IO_WritePin** (uint16_t IO_Pin, uint8_t PinState) Sets the selected pins state.
uint16_t	**BSP_IO_ReadPin** (uint16_t IO_Pin) Gets the selected pins current state.
void	**BSP_IO_TogglePin** (uint16_t IO_Pin) Toggles the selected pins state.

Function Documentation

uint8_t BSP_IO_ConfigPin (uint16_t IO_Pin, IO_ModeTypedef IO_Mode)

Configures the IO pin(s) according to IO mode structure value.

Parameters:

IO_Pin,: IO pin(s) to be configured. This parameter can be one of the following values:
- STMPE1600_PIN_x: where x can be from 0 to 15.

IO_Mode,: 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

Return values:

IO_OK if all initializations are OK. Other value if error.

Definition at line **192** of file **stm324x9i_eval_io.c**.

References **io_driver**, **IO_I2C_ADDRESS**, and **IO_OK**.

Referenced by **BSP_CAMERA_Init()**, **BSP_CAMERA_Stop()**, **BSP_JOY_Init()**, **BSP_SD_ITConfig()**, and **BSP_TS_ITConfig()**.

uint8_t BSP_IO_Init (void)

Initializes and configures the IO functionalities and configures all necessary hardware resources (GPIOs, clocks..).

Note:

BSP_IO_Init() is using HAL_Delay() function to ensure that stmpe1600 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 138 of file stm324x9i_eval_io.c.

References io_driver, IO_ERROR, IO_I2C_ADDRESS, IO_OK, and IO_PIN_ALL.

Referenced by BSP_CAMERA_Init(), BSP_CAMERA_Stop(), BSP_JOY_Init(), BSP_SD_Init(), and BSP_TS_ITConfig().

void BSP_IO_ITClear (void)

Clears all the IO IT pending bits.

Definition at line 173 of file stm324x9i_eval_io.c.

References io_driver, and IO_I2C_ADDRESS.

Referenced by BSP_SD_DetectIT(), and BSP_TS_ITClear().

uint8_t BSP_IO_ITGetStatus (uint16_t IO_Pin)

Gets the selected pins IT status.

Parameters:

IO_Pin,: 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 **164** of file **stm324x9i_eval_io.c**.

References **io_driver**, and **IO_I2C_ADDRESS**.

uint16_t BSP_IO_ReadPin (uint16_t IO_Pin)

Gets the selected pins current state.

Parameters:

IO_Pin,: Selected pins to read. This parameter can be any combination of the IO pins.

Return values:

The current pins state

Definition at line **218** of file **stm324x9i_eval_io.c**.

References **io_driver**, and **IO_I2C_ADDRESS**.

Referenced by **BSP_CAMERA_Init()**, **BSP_JOY_GetState()**, and **BSP_SD_IsDetected()**.

void BSP_IO_TogglePin (uint16_t IO_Pin)

Toggles the selected pins state.

Parameters:

IO_Pin,: Selected pins to toggle. This parameter can be any combination of the IO pins.

Definition at line **228** of file **stm324x9i_eval_io.c**.

References **io_driver**, and **IO_I2C_ADDRESS**.

void BSP_IO_WritePin (uint16_t IO_Pin,
uint8_t PinState
)

Sets the selected pins state.

Parameters:

IO_Pin,: Selected pins to write. This parameter can be any combination of the IO pins.

PinState,: New pins state to write

Definition at line **206** of file **stm324x9i_eval_io.c**.

References **io_driver**, and **IO_I2C_ADDRESS**.

Referenced by **BSP_CAMERA_Init()**, and **BSP_CAMERA_Stop()**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL IO Exported Functions

STM324x9I EVAL IO

Functions

uint8_t	**BSP_IO_Init** (void) Initializes and configures the IO functionalities and configures all necessary hardware resources (GPIOs, clocks..).
uint8_t	**BSP_IO_ITGetStatus** (uint16_t IO_Pin) Gets the selected pins IT status.
void	**BSP_IO_ITClear** (void) Clears all the IO IT pending bits.
uint8_t	**BSP_IO_ConfigPin** (uint16_t IO_Pin, IO_ModeTypedef IO_Mode) Configures the IO pin(s) according to IO mode structure value.
void	**BSP_IO_WritePin** (uint16_t IO_Pin, uint8_t PinState) Sets the selected pins state.
uint16_t	**BSP_IO_ReadPin** (uint16_t IO_Pin) Gets the selected pins current state.
void	**BSP_IO_TogglePin** (uint16_t IO_Pin) Toggles the selected pins state.

Function Documentation

uint8_t BSP_IO_ConfigPin (uint16_t IO_Pin, IO_ModeTypedef IO_Mode)

Configures the IO pin(s) according to IO mode structure value.

Parameters:

IO_Pin,: IO pin(s) to be configured. This parameter can be one of the following values:
- STMPE1600_PIN_x: where x can be from 0 to 15.

IO_Mode,: 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

Return values:

IO_OK if all initializations are OK. Other value if error.

Definition at line **192** of file **stm324x9i_eval_io.c**.

References **io_driver**, **IO_I2C_ADDRESS**, and **IO_OK**.

Referenced by **BSP_CAMERA_Init()**, **BSP_CAMERA_Stop()**, **BSP_JOY_Init()**, **BSP_SD_ITConfig()**, and **BSP_TS_ITConfig()**.

uint8_t BSP_IO_Init (void)

Initializes and configures the IO functionalities and configures all necessary hardware resources (GPIOs, clocks..).

Note:

BSP_IO_Init() is using HAL_Delay() function to ensure that stmpe1600 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 138 of file stm324x9i_eval_io.c.

References io_driver, IO_ERROR, IO_I2C_ADDRESS, IO_OK, and IO_PIN_ALL.

Referenced by BSP_CAMERA_Init(), BSP_CAMERA_Stop(), BSP_JOY_Init(), BSP_SD_Init(), and BSP_TS_ITConfig().

void BSP_IO_ITClear (void)

Clears all the IO IT pending bits.

Definition at line 173 of file stm324x9i_eval_io.c.

References io_driver, and IO_I2C_ADDRESS.

Referenced by BSP_SD_DetectIT(), and BSP_TS_ITClear().

uint8_t BSP_IO_ITGetStatus (uint16_t IO_Pin)

Gets the selected pins IT status.

Parameters:

IO_Pin,: 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 **164** of file **stm324x9i_eval_io.c**.

References **io_driver**, and **IO_I2C_ADDRESS**.

uint16_t BSP_IO_ReadPin (uint16_t IO_Pin)

Gets the selected pins current state.

Parameters:

IO_Pin,: Selected pins to read. This parameter can be any combination of the IO pins.

Return values:

The current pins state

Definition at line **218** of file **stm324x9i_eval_io.c**.

References **io_driver**, and **IO_I2C_ADDRESS**.

Referenced by **BSP_CAMERA_Init()**, **BSP_JOY_GetState()**, and **BSP_SD_IsDetected()**.

void BSP_IO_TogglePin (uint16_t IO_Pin)

Toggles the selected pins state.

Parameters:

IO_Pin,: Selected pins to toggle. This parameter can be any combination of the IO pins.

Definition at line **228** of file **stm324x9i_eval_io.c**.

References **io_driver**, and **IO_I2C_ADDRESS**.

void BSP_IO_WritePin (uint16_t IO_Pin,
uint8_t PinState
)

Sets the selected pins state.

Parameters:

IO_Pin,: Selected pins to write. This parameter can be any combination of the IO pins.

PinState,: New pins state to write

Definition at line **206** of file **stm324x9i_eval_io.c**.

References **io_driver**, and **IO_I2C_ADDRESS**.

Referenced by **BSP_CAMERA_Init()**, and **BSP_CAMERA_Stop()**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL LCD Private Functions

STM324x9I EVAL LCD

Functions

uint8_t	**BSP_LCD_Init** (void) Initializes the LCD.
uint8_t	**BSP_LCD_InitEx** (uint32_t PclkConfig) Initializes the LCD.
uint32_t	**BSP_LCD_GetXSize** (void) Gets the LCD X size.
uint32_t	**BSP_LCD_GetYSize** (void) Gets the LCD Y size.
void	**BSP_LCD_LayerDefaultInit** (uint16_t LayerIndex, uint32_t FB_Address) Initializes the LCD layers.
void	**BSP_LCD_SelectLayer** (uint32_t LayerIndex) Selects the LCD Layer.
void	**BSP_LCD_SetLayerVisible** (uint32_t LayerIndex, FunctionalState State) Sets an LCD Layer visible.
void	**BSP_LCD_SetTransparency** (uint32_t LayerIndex, uint8_t Transparency) Configures the transparency.
void	**BSP_LCD_SetLayerAddress** (uint32_t LayerIndex, uint32_t Address) Sets an LCD layer frame buffer address.
void	**BSP_LCD_SetLayerWindow** (uint16_t LayerIndex, uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Sets display window.
void	**BSP_LCD_SetColorKeying** (uint32_t LayerIndex, uint32_t RGBValue) Configures and sets the color keying.
void	**BSP_LCD_ResetColorKeying** (uint32_t LayerIndex) Disables the color keying.

void	**BSP_LCD_SetTextColor** (uint32_t Color) Sets the LCD text color.
uint32_t	**BSP_LCD_GetTextColor** (void) Gets the LCD text color.
void	**BSP_LCD_SetBackColor** (uint32_t Color) Sets the LCD background color.
uint32_t	**BSP_LCD_GetBackColor** (void) Gets 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.
uint32_t	**BSP_LCD_ReadPixel** (uint16_t Xpos, uint16_t Ypos) Reads an LCD pixel.
void	**BSP_LCD_Clear** (uint32_t Color) Clears the hole LCD.
void	**BSP_LCD_ClearStringLine** (uint32_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, **Text_AlignModeTypdef** Mode) Displays characters on the LCD.
void	**BSP_LCD_DisplayStringAtLine** (uint16_t Line, uint8_t *ptr) Displays a maximum of 60 characters on the 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.
	BSP_LCD_DrawLine (uint16_t x1, uint16_t y1, uint16_t

void	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** (uint32_t Xpos, uint32_t Ypos, uint8_t *pbmp) Draws a bitmap picture loaded in the internal Flash (32 bpp).
void	**BSP_LCD_FillRect** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Draws a full rectangle.
void	**BSP_LCD_FillCircle** (uint16_t Xpos, uint16_t Ypos, uint16_t Radius) Draws a full circle.
void	**BSP_LCD_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.

__weak void	**BSP_LCD_ClockConfig** (LTDC_HandleTypeDef *hltdc, void *Params) Clock Config.
void	**BSP_LCD_DrawPixel** (uint16_t Xpos, uint16_t Ypos, uint32_t RGB_Code) Draws a pixel on LCD.
static void	**DrawChar** (uint16_t Xpos, uint16_t Ypos, const uint8_t *c) Draws a character on LCD.
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).
static void	**LL_FillBuffer** (uint32_t LayerIndex, void *pDst, uint32_t xSize, uint32_t ySize, uint32_t OffLine, uint32_t ColorIndex) Fills a buffer.
static void	**LL_ConvertLineToARGB8888** (void *pSrc, void *pDst, uint32_t xSize, uint32_t ColorMode) Converts a line to an ARGB8888 pixel format.

Function Documentation

void BSP_LCD_Clear (uint32_t Color)

Clears the hole LCD.

Parameters:

Color,: Color of the background

Definition at line 471 of file stm324x9i_eval_lcd.c.

References ActiveLayer, BSP_LCD_GetXSize(), BSP_LCD_GetYSize(), hltdc_eval, and LL_FillBuffer().

void BSP_LCD_ClearStringLine (uint32_t Line)

Clears the selected line.

Parameters:

Line,: Line to be cleared

Definition at line 481 of file stm324x9i_eval_lcd.c.

References ActiveLayer, LCD_DrawPropTypeDef::BackColor, BSP_LCD_FillRect(), BSP_LCD_GetXSize(), BSP_LCD_SetTextColor(), and LCD_DrawPropTypeDef::TextColor.

__weak void BSP_LCD_ClockConfig (LTDC_HandleTypeDef * hltdc, void * Para)

Clock Config.

Parameters:

hltdc,: LTDC handle

Params,: LTDC pixel clock

Note:

This API is called by **BSP_LCD_Init()** Being __weak it can be overwritten by the application

Definition at line **1108** of file **stm324x9i_eval_lcd.c**.

References **LCD_MAX_PCLK**, and **TS_I2C_ADDRESS**.

Referenced by **BSP_LCD_InitEx()**.

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 **500** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, **DrawChar()**, and **LCD_DrawPropTypeDef::pFont**.

Referenced by **BSP_LCD_DisplayStringAt()**.

void BSP_LCD_DisplayOff (void)

Disables the display.

Definition at line **1045** of file **stm324x9i_eval_lcd.c**.

References **hltdc_eval**.

void BSP_LCD_DisplayOn (void)

Enables the display.

Definition at line **1036** of file **stm324x9i_eval_lcd.c**.

References **hltdc_eval**.

void BSP_LCD_DisplayStringAt (
uint16_t Xpos,
uint16_t Ypos,
uint8_t * Text,
Text_AlignModeTypdef 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 517 of file stm324x9i_eval_lcd.c.

References ActiveLayer, 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 maximum of 60 characters on the LCD.

Parameters:

Line,: Line where to display the character shape
ptr,: Pointer to string to display on LCD

Definition at line 571 of file stm324x9i_eval_lcd.c.

References BSP_LCD_DisplayStringAt(), and LEFT_MODE.

void BSP_LCD_DrawBitmap (uint32_t Xpos,
uint32_t Ypos,
uint8_t * pbmp
)

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

Parameters:

Xpos,: Bmp X position in the LCD
Ypos,: Bmp Y position in the LCD
pbmp,: Pointer to Bmp picture address in the internal Flash

Definition at line **813** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, **BSP_LCD_GetXSize()**, **hltdc_eval**, and **LL_ConvertLineToARGB8888()**.

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 **708** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **BSP_LCD_DrawPixel()**.

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 **781** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **BSP_LCD_DrawPixel()**.

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 **582** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, **BSP_LCD_GetXSize()**, **hltdc_eval**, and **LL_FillBuffer()**.

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

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 **617** of file **stm324x9i_eval_lcd.c**.

References **ABS**, **ActiveLayer**, and **BSP_LCD_DrawPixel()**.

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

void BSP_LCD_DrawPixel (uint16_t Xpos,
uint16_t Ypos,
uint32_t RGB_Code
)

Draws a pixel on LCD.

Parameters:

- **Xpos,:** X position
- **Ypos,:** Y position
- **RGB_Code,:** Pixel color in ARGB mode (8-8-8-8)

Definition at line **1168** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, **BSP_LCD_GetXSize()**, and **hltdc_eval**.

Referenced by **BSP_LCD_DrawCircle()**, **BSP_LCD_DrawEllipse()**, **BSP_LCD_DrawLine()**, 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 **754** of file **stm324x9i_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 **691** of file **stm324x9i_eval_lcd.c**.

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

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 **599** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, **BSP_LCD_GetXSize()**, **hltdc_eval**, and **LL_FillBuffer()**.

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 **893** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, **BSP_LCD_DrawCircle()**, **BSP_LCD_DrawHLine()**, and **BSP_LCD_SetTextColor()**.

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 **1007** of file **stm324x9i_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 **940** of file **stm324x9i_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 **873** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, **BSP_LCD_GetXSize()**, **BSP_LCD_SetTextColor()**, **hltdc_eval**, and **LL_FillBuffer()**.

Referenced by **BSP_LCD_ClearStringLine()**.

uint32_t BSP_LCD_GetBackColor (void)

Gets the LCD background color.

Return values:

Used	background color

Definition at line **408** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **LCD_DrawPropTypeDef::BackColor**.

sFONT* BSP_LCD_GetFont (void)

Gets the LCD text font.

Return values:

Used	layer font

Definition at line **426** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **LCD_DrawPropTypeDef::pFont**.

uint32_t BSP_LCD_GetTextColor (void)

Gets the LCD text color.

Return values:

Used text color.

Definition at line **390** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **LCD_DrawPropTypeDef::TextColor**.

uint32_t BSP_LCD_GetXSize (void)

Gets the LCD X size.

Return values:

Used LCD X size

Definition at line **240** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **hltdc_eval**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_ClearStringLine()**, **BSP_LCD_DisplayStringAt()**, **BSP_LCD_DrawBitmap()**, **BSP_LCD_DrawHLine()**, **BSP_LCD_DrawPixel()**, **BSP_LCD_DrawVLine()**, **BSP_LCD_FillRect()**, **BSP_LCD_LayerDefaultInit()**, and **BSP_LCD_ReadPixel()**.

uint32_t BSP_LCD_GetYSize (void)

Gets the LCD Y size.

Return values:

Used LCD Y size

Definition at line **249** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **hltdc_eval**.

Referenced by **BSP_LCD_Clear()**, and **BSP_LCD_LayerDefaultInit()**.

uint8_t BSP_LCD_Init (void)

Initializes the LCD.

Return values:
LCD state

Definition at line **155** of file **stm324x9i_eval_lcd.c**.

References **BSP_LCD_InitEx()**, and **LCD_MAX_PCLK**.

uint8_t BSP_LCD_InitEx (uint32_t PclkConfig)

Initializes the LCD.

Parameters:
PclkConfig : pixel clock profile

Return values:
LCD state

Definition at line **165** of file **stm324x9i_eval_lcd.c**.

References **BSP_LCD_ClockConfig()**, **BSP_LCD_SetFont()**, **BSP_SDRAM_Init()**, **hltdc_eval**, **LCD_DEFAULT_FONT**, **LCD_OK**, **MspInit()**, **PCLK_profile**, and **TS_I2C_ADDRESS**.

Referenced by **BSP_LCD_Init()**.

void BSP_LCD_LayerDefaultInit (uint16_t LayerIndex,
uint32_t FB_Address
)

Initializes the LCD layers.

Parameters:

LayerIndex,: Layer foreground or background
FB_Address,: Layer frame buffer

Definition at line **259** of file **stm324x9i_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**, **BSP_LCD_GetXSize()**, **BSP_LCD_GetYSize()**, **hltdc_eval**, **LCD_COLOR_BLACK**, **LCD_COLOR_WHITE**, **LCD_LayerCfgTypeDef**, **LCD_DrawPropTypeDef::pFont**, and **LCD_DrawPropTypeDef::TextColor**.

uint32_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 **437** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, **BSP_LCD_GetXSize()**, and **hltdc_eval**.

void BSP_LCD_ResetColorKeying (uint32_t LayerIndex)

Disables the color keying.

Parameters:

LayerIndex,: Layer foreground or background

Definition at line **371** of file **stm324x9i_eval_lcd.c**.

References **hltdc_eval**.

void BSP_LCD_SelectLayer (uint32_t LayerIndex)

Selects the LCD Layer.

Parameters:

LayerIndex,: Layer foreground or background

Definition at line **291** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**.

void BSP_LCD_SetBackColor (uint32_t Color)

Sets the LCD background color.

Parameters:

Color,: Layer background color code ARGB(8-8-8-8)

Definition at line **399** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **LCD_DrawPropTypeDef::BackColor**.

void BSP_LCD_SetColorKeying (uint32_t LayerIndex,
uint32_t RGBValue
)

Configures and sets the color keying.

Parameters:

LayerIndex,: Layer foreground or background

RGBValue,: Color reference

Definition at line **360** of file **stm324x9i_eval_lcd.c**.

References **hltdc_eval**.

void BSP_LCD_SetFont (sFONT * fonts)

Sets the LCD text font.

Parameters:

fonts,: Layer font to be used

Definition at line **417** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **LCD_DrawPropTypeDef::pFont**.

Referenced by **BSP_LCD_InitEx()**.

void BSP_LCD_SetLayerAddress (uint32_t LayerIndex,
uint32_t Address
)

Sets an LCD layer frame buffer address.

Parameters:

LayerIndex,: Layer foreground or background
Address,: New LCD frame buffer value

Definition at line **333** of file **stm324x9i_eval_lcd.c**.

References **hltdc_eval**.

void BSP_LCD_SetLayerVisible (uint32_t LayerIndex,
FunctionalState State
)

Sets an LCD Layer visible.

Parameters:

LayerIndex,: Visible Layer
State,: New state of the specified layer This parameter can be one of the following values:
- ENABLE
- DISABLE

Definition at line **304** of file **stm324x9i_eval_lcd.c**.

References **hltdc_eval**.

void BSP_LCD_SetLayerWindow (uint16_t LayerIndex,
uint16_t Xpos,
uint16_t Ypos,
uint16_t Width,
uint16_t Height
)

Sets display window.

Parameters:

LayerIndex,: Layer index
Xpos,: LCD X position
Ypos,: LCD Y position
Width,: LCD window width
Height,: LCD window height

Definition at line **346** of file **stm324x9i_eval_lcd.c**.

References **hltdc_eval**.

void BSP_LCD_SetTextColor (uint32_t Color)

Sets the LCD text color.

Parameters:

Color,: Text color code ARGB(8-8-8-8)

Definition at line **381** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **LCD_DrawPropTypeDef::TextColor**.

Referenced by **BSP_LCD_ClearStringLine()**, **BSP_LCD_FillCircle()**, and **BSP_LCD_FillRect()**.

void BSP_LCD_SetTransparency (uint32_t LayerIndex, uint8_t Transparency)

Configures the transparency.

Parameters:

LayerIndex,: Layer foreground or background.
Transparency,: Transparency This parameter must be a number between Min_Data = 0x00 and

Max_Data = 0xFF

Definition at line **323** of file **stm324x9i_eval_lcd.c**.

References **hltdc_eval**.

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 **1180** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, **BSP_LCD_DrawPixel()**, and **LCD_DrawPropTypeDef::pFont**.

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 **1238** of file **stm324x9i_eval_lcd.c**.

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

Referenced by **BSP_LCD_FillPolygon()**.

static void LL_ConvertLineToARGB8888 (void * pSrc, void * pDst, uint32_t xSize, uint32_t ColorMode) [static]

Converts a line to an ARGB8888 pixel format.

Parameters:

- **pSrc,:** Pointer to source buffer
- **pDst,:** Output color
- **xSize,:** Buffer width
- **ColorMode,:** Input color mode

Definition at line **1345** of file **stm324x9i_eval_lcd.c**.

References **hdma2d_eval**.

Referenced by **BSP_LCD_DrawBitmap()**.

```
static void LL_FillBuffer ( uint32_t  LayerIndex,
                            void *    pDst,
                            uint32_t  xSize,
                            uint32_t  ySize,
                            uint32_t  OffLine,
                            uint32_t  ColorIndex
                          )           [static]
```

Fills a buffer.

Parameters:

LayerIndex,:	Layer index
pDst,:	Pointer to destination buffer
xSize,:	Buffer width
ySize,:	Buffer height
OffLine,:	Offset
ColorIndex,:	Color index

Definition at line **1315** of file **stm324x9i_eval_lcd.c**.

References **hdma2d_eval**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_DrawHLine()**, **BSP_LCD_DrawVLine()**, and **BSP_LCD_FillRect()**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL LCD Exported Functions

STM324x9I EVAL LCD

Functions

uint8_t	**BSP_LCD_Init** (void) Initializes the LCD.
uint8_t	**BSP_LCD_InitEx** (uint32_t PclkConfig) Initializes the LCD.
uint32_t	**BSP_LCD_GetXSize** (void) Gets the LCD X size.
uint32_t	**BSP_LCD_GetYSize** (void) Gets the LCD Y size.
void	**BSP_LCD_LayerDefaultInit** (uint16_t LayerIndex, uint32_t FrameBuffer) Initializes the LCD layers.
void	**BSP_LCD_SetTransparency** (uint32_t LayerIndex, uint8_t Transparency) Configures the transparency.
void	**BSP_LCD_SetLayerAddress** (uint32_t LayerIndex, uint32_t Address) Sets an LCD layer frame buffer address.
void	**BSP_LCD_SetColorKeying** (uint32_t LayerIndex, uint32_t RGBValue) Configures and sets the color keying.
void	**BSP_LCD_ResetColorKeying** (uint32_t LayerIndex) Disables the color keying.
void	**BSP_LCD_SetLayerWindow** (uint16_t LayerIndex, uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Sets display window.
void	**BSP_LCD_SelectLayer** (uint32_t LayerIndex) Selects the LCD Layer.
void	**BSP_LCD_SetLayerVisible** (uint32_t LayerIndex, FunctionalState State) Sets an LCD Layer visible.

void	**BSP_LCD_SetTextColor** (uint32_t Color) Sets the LCD text color.
uint32_t	**BSP_LCD_GetTextColor** (void) Gets the LCD text color.
void	**BSP_LCD_SetBackColor** (uint32_t Color) Sets the LCD background color.
uint32_t	**BSP_LCD_GetBackColor** (void) Gets 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.
uint32_t	**BSP_LCD_ReadPixel** (uint16_t Xpos, uint16_t Ypos) Reads an LCD pixel.
void	**BSP_LCD_DrawPixel** (uint16_t Xpos, uint16_t Ypos, uint32_t pixel) Draws a pixel on LCD.
void	**BSP_LCD_Clear** (uint32_t Color) Clears the hole LCD.
void	**BSP_LCD_ClearStringLine** (uint32_t Line) Clears the selected line.
void	**BSP_LCD_DisplayStringAtLine** (uint16_t Line, uint8_t *ptr) Displays a maximum of 60 characters on the LCD.
void	**BSP_LCD_DisplayStringAt** (uint16_t Xpos, uint16_t Ypos, uint8_t *Text, **Text_AlignModeTypdef** Mode) Displays characters on the LCD.
void	**BSP_LCD_DisplayChar** (uint16_t Xpos, uint16_t Ypos, uint8_t Ascii) Displays one character.
void	**BSP_LCD_DrawHLine** (uint16_t Xpos, uint16_t Ypos, uint16_t Length) Draws an horizontal line.
	BSP_LCD_DrawVLine (uint16_t Xpos, uint16_t Ypos,

void	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** (uint32_t Xpos, uint32_t Ypos, uint8_t *pbmp) Draws a bitmap picture loaded in the internal Flash (32 bpp).
void	**BSP_LCD_FillRect** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Draws a full rectangle.
void	**BSP_LCD_FillCircle** (uint16_t Xpos, uint16_t Ypos, uint16_t Radius) Draws a full circle.
void	**BSP_LCD_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.
void	**BSP_LCD_ClockConfig** (LTDC_HandleTypeDef *hltdc, void *Params) Clock Config.

Function Documentation

void BSP_LCD_Clear (uint32_t Color)

Clears the hole LCD.

Parameters:

Color,: Color of the background

Definition at line 471 of file stm324x9i_eval_lcd.c.

References ActiveLayer, BSP_LCD_GetXSize(), BSP_LCD_GetYSize(), hltdc_eval, and LL_FillBuffer().

void BSP_LCD_ClearStringLine (uint32_t Line)

Clears the selected line.

Parameters:

Line,: Line to be cleared

Definition at line 481 of file stm324x9i_eval_lcd.c.

References ActiveLayer, LCD_DrawPropTypeDef::BackColor, BSP_LCD_FillRect(), BSP_LCD_GetXSize(), BSP_LCD_SetTextColor(), and LCD_DrawPropTypeDef::TextColor.

void BSP_LCD_ClockConfig (LTDC_HandleTypeDef * hltdc, void * Params)

Clock Config.

Parameters:

hltdc,: LTDC handle

Params,: LTDC pixel clock

Note:

This API is called by **BSP_LCD_Init()** Being __weak it can be overwritten by the application

Definition at line **1108** of file **stm324x9i_eval_lcd.c**.

References **LCD_MAX_PCLK**, and **TS_I2C_ADDRESS**.

Referenced by **BSP_LCD_InitEx()**.

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 **500** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, **DrawChar()**, and **LCD_DrawPropTypeDef::pFont**.

Referenced by **BSP_LCD_DisplayStringAt()**.

void BSP_LCD_DisplayOff (void)

Disables the display.

Definition at line **1045** of file **stm324x9i_eval_lcd.c**.

References **hltdc_eval**.

void BSP_LCD_DisplayOn (void)

Enables the display.

Definition at line **1036** of file **stm324x9i_eval_lcd.c**.

References **hltdc_eval**.

void BSP_LCD_DisplayStringAt (
uint16_t Xpos,
uint16_t Ypos,
uint8_t * Text,
Text_AlignModeTypdef 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 517 of file stm324x9i_eval_lcd.c.

References ActiveLayer, 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 maximum of 60 characters on the LCD.

Parameters:

Line,: Line where to display the character shape
ptr,: Pointer to string to display on LCD

Definition at line 571 of file stm324x9i_eval_lcd.c.

References BSP_LCD_DisplayStringAt(), and LEFT_MODE.

void BSP_LCD_DrawBitmap (uint32_t Xpos,
uint32_t Ypos,
uint8_t * pbmp
)

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

Parameters:

Xpos,: Bmp X position in the LCD
Ypos,: Bmp Y position in the LCD
pbmp,: Pointer to Bmp picture address in the internal Flash

Definition at line **813** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, **BSP_LCD_GetXSize()**, **hltdc_eval**, and **LL_ConvertLineToARGB8888()**.

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 **708** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **BSP_LCD_DrawPixel()**.

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 **781** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **BSP_LCD_DrawPixel()**.

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 **582** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, **BSP_LCD_GetXSize()**, **hltdc_eval**, and **LL_FillBuffer()**.

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

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 **617** of file **stm324x9i_eval_lcd.c**.

References **ABS**, **ActiveLayer**, and **BSP_LCD_DrawPixel()**.

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

void BSP_LCD_DrawPixel (uint16_t Xpos,
uint16_t Ypos,
uint32_t RGB_Code
)

Draws a pixel on LCD.

Parameters:

- **Xpos,:** X position
- **Ypos,:** Y position
- **RGB_Code,:** Pixel color in ARGB mode (8-8-8-8)

Definition at line **1168** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, **BSP_LCD_GetXSize()**, and **hltdc_eval**.

Referenced by **BSP_LCD_DrawCircle()**, **BSP_LCD_DrawEllipse()**, **BSP_LCD_DrawLine()**, 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 **754** of file **stm324x9i_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 **691** of file **stm324x9i_eval_lcd.c**.

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

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 **599** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, **BSP_LCD_GetXSize()**, **hltdc_eval**, and **LL_FillBuffer()**.

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 **893** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, **BSP_LCD_DrawCircle()**, **BSP_LCD_DrawHLine()**, and **BSP_LCD_SetTextColor()**.

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 **1007** of file **stm324x9i_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 **940** of file **stm324x9i_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 **873** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, **BSP_LCD_GetXSize()**, **BSP_LCD_SetTextColor()**, **hltdc_eval**, and **LL_FillBuffer()**.

Referenced by **BSP_LCD_ClearStringLine()**.

uint32_t BSP_LCD_GetBackColor (void)

Gets the LCD background color.

Return values:

Used	background color

Definition at line **408** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **LCD_DrawPropTypeDef::BackColor**.

sFONT* BSP_LCD_GetFont (void)

Gets the LCD text font.

Return values:

Used	layer font

Definition at line **426** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **LCD_DrawPropTypeDef::pFont**.

uint32_t BSP_LCD_GetTextColor (void)

Gets the LCD text color.

Return values:

Used text color.

Definition at line **390** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **LCD_DrawPropTypeDef::TextColor**.

uint32_t BSP_LCD_GetXSize (void)

Gets the LCD X size.

Return values:

Used LCD X size

Definition at line **240** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **hltdc_eval**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_ClearStringLine()**, **BSP_LCD_DisplayStringAt()**, **BSP_LCD_DrawBitmap()**, **BSP_LCD_DrawHLine()**, **BSP_LCD_DrawPixel()**, **BSP_LCD_DrawVLine()**, **BSP_LCD_FillRect()**, **BSP_LCD_LayerDefaultInit()**, and **BSP_LCD_ReadPixel()**.

uint32_t BSP_LCD_GetYSize (void)

Gets the LCD Y size.

Return values:

Used LCD Y size

Definition at line **249** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **hltdc_eval**.

Referenced by **BSP_LCD_Clear()**, and **BSP_LCD_LayerDefaultInit()**.

uint8_t BSP_LCD_Init (void)

Initializes the LCD.

Return values:
LCD state

Definition at line **155** of file **stm324x9i_eval_lcd.c**.

References **BSP_LCD_InitEx()**, and **LCD_MAX_PCLK**.

uint8_t BSP_LCD_InitEx (uint32_t PclkConfig)

Initializes the LCD.

Parameters:
PclkConfig : pixel clock profile

Return values:
LCD state

Definition at line **165** of file **stm324x9i_eval_lcd.c**.

References **BSP_LCD_ClockConfig()**, **BSP_LCD_SetFont()**, **BSP_SDRAM_Init()**, **hltdc_eval**, **LCD_DEFAULT_FONT**, **LCD_OK**, **MspInit()**, **PCLK_profile**, and **TS_I2C_ADDRESS**.

Referenced by **BSP_LCD_Init()**.

void BSP_LCD_LayerDefaultInit (uint16_t LayerIndex,
uint32_t FB_Address
)

Initializes the LCD layers.

Parameters:

LayerIndex,: Layer foreground or background
FB_Address,: Layer frame buffer

Definition at line **259** of file **stm324x9i_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**, **BSP_LCD_GetXSize()**, **BSP_LCD_GetYSize()**, **hltdc_eval**, **LCD_COLOR_BLACK**, **LCD_COLOR_WHITE**, **LCD_LayerCfgTypeDef**, **LCD_DrawPropTypeDef::pFont**, and **LCD_DrawPropTypeDef::TextColor**.

uint32_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 **437** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, **BSP_LCD_GetXSize()**, and **hltdc_eval**.

void BSP_LCD_ResetColorKeying (uint32_t LayerIndex)

Disables the color keying.

Parameters:

LayerIndex,: Layer foreground or background

Definition at line **371** of file **stm324x9i_eval_lcd.c**.

References **hltdc_eval**.

void BSP_LCD_SelectLayer (uint32_t LayerIndex)

Selects the LCD Layer.

Parameters:

LayerIndex,: Layer foreground or background

Definition at line **291** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**.

void BSP_LCD_SetBackColor (uint32_t Color)

Sets the LCD background color.

Parameters:

Color,: Layer background color code ARGB(8-8-8-8)

Definition at line **399** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **LCD_DrawPropTypeDef::BackColor**.

void BSP_LCD_SetColorKeying (uint32_t LayerIndex,
uint32_t RGBValue
)

Configures and sets the color keying.

Parameters:

LayerIndex,: Layer foreground or background

RGBValue,: Color reference

Definition at line **360** of file **stm324x9i_eval_lcd.c**.

References **hltdc_eval**.

void BSP_LCD_SetFont (sFONT * fonts)

Sets the LCD text font.

Parameters:

fonts,: Layer font to be used

Definition at line **417** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **LCD_DrawPropTypeDef::pFont**.

Referenced by **BSP_LCD_InitEx()**.

void BSP_LCD_SetLayerAddress (uint32_t LayerIndex,
uint32_t Address
)

Sets an LCD layer frame buffer address.

Parameters:

LayerIndex,: Layer foreground or background
Address,: New LCD frame buffer value

Definition at line **333** of file **stm324x9i_eval_lcd.c**.

References **hltdc_eval**.

void BSP_LCD_SetLayerVisible (uint32_t LayerIndex,
FunctionalState State
)

Sets an LCD Layer visible.

Parameters:

LayerIndex,: Visible Layer
State,: New state of the specified layer This parameter can be one of the following values:
- ENABLE
- DISABLE

Definition at line **304** of file **stm324x9i_eval_lcd.c**.

References **hltdc_eval**.

void BSP_LCD_SetLayerWindow (uint16_t LayerIndex,
uint16_t Xpos,
uint16_t Ypos,
uint16_t Width,
uint16_t Height
)

Sets display window.

Parameters:

LayerIndex,: Layer index
Xpos,: LCD X position
Ypos,: LCD Y position
Width,: LCD window width
Height,: LCD window height

Definition at line **346** of file **stm324x9i_eval_lcd.c**.

References **hltdc_eval**.

void BSP_LCD_SetTextColor (uint32_t Color)

Sets the LCD text color.

Parameters:

Color,: Text color code ARGB(8-8-8-8)

Definition at line **381** of file **stm324x9i_eval_lcd.c**.

References **ActiveLayer**, and **LCD_DrawPropTypeDef::TextColor**.

Referenced by **BSP_LCD_ClearStringLine()**, **BSP_LCD_FillCircle()**, and **BSP_LCD_FillRect()**.

void BSP_LCD_SetTransparency (uint32_t LayerIndex, uint8_t Transparency)

Configures the transparency.

Parameters:

LayerIndex,: Layer foreground or background.
Transparency,: Transparency This parameter must be a number between Min_Data = 0x00 and

Max_Data = 0xFF

Definition at line **323** of file **stm324x9i_eval_lcd.c**.

References **hltdc_eval**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL NOR Exported Functions

STM324x9I EVAL NOR

Functions

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

Function Documentation

uint8_t BSP_NOR_Erase_Block (uint32_t BlockAddress)

Erases the specified block of the NOR device.

Parameters:

BlockAddress,: Block address to erase

Return values:

NOR memory status

Definition at line **282** of file **stm324x9i_eval_nor.c**.

References **BLOCKERASE_TIMEOUT**, **NOR_DEVICE_ADDR**, **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, and **norHandle**.

uint8_t BSP_NOR_Erase_Chip (void)

Erases the entire NOR chip.

Return values:

NOR memory status

Definition at line **302** of file **stm324x9i_eval_nor.c**.

References **CHIPERASE_TIMEOUT**, **NOR_DEVICE_ADDR**, **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, and **norHandle**.

uint8_t BSP_NOR_Init (void)

Initializes the NOR device.

Return values:

NOR memory status

Definition at line **154** of file **stm324x9i_eval_nor.c**.

References **CONTINUOUSCLOCK_FEATURE**, **NOR_BURSTACCESS**, **NOR_MEMORY_WIDTH**, **NOR_MspInit()**, **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, **NOR_WRITEBURST**, **norHandle**, and **Timing**.

uint8_t BSP_NOR_ProgramData (uint32_t uwStartAddress,
uint16_t * pData,
uint32_t uwDataSize
)

Programs an amount of data to the NOR device.

Parameters:

uwStartAddress,: Write start address
pData,: Pointer to data to be written
uwDataSize,: Size of data to write

Return values:

NOR memory status

Definition at line **261** of file **stm324x9i_eval_nor.c**.

References **NOR_DEVICE_ADDR**, **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, **norHandle**, and **PROGRAM_TIMEOUT**.

uint8_t BSP_NOR_Read_ID (NOR_IDTypeDef * pNOR_ID)

Reads NOR flash IDs.

Parameters:

pNOR_ID : Pointer to NOR ID structure

Return values:

NOR memory status

Definition at line **323** of file **stm324x9i_eval_nor.c**.

References **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, and **norHandle**.

uint8_t BSP_NOR_ReadData (uint32_t uwStartAddress, uint16_t * pData, uint32_t uwDataSize)

Reads an amount of data from the NOR device.

Parameters:

uwStartAddress,:	Read start address
pData,:	Pointer to data to be read
uwDataSize,:	Size of data to read

Return values:

NOR memory status

Definition at line **203** of file **stm324x9i_eval_nor.c**.

References **NOR_DEVICE_ADDR**, **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, and **norHandle**.

void BSP_NOR_ReturnToReadMode (void)

Returns the NOR memory to read mode.

Definition at line **218** of file **stm324x9i_eval_nor.c**.

References **norHandle**.

uint8_t BSP_NOR_WriteData (uint32_t uwStartAddress, uint16_t * pData, uint32_t uwDataSize)

Writes an amount of data to the NOR device.

Parameters:

uwStartAddress,:	Write start address
pData,:	Pointer to data to be written
uwDataSize,:	Size of data to write

Return values:

NOR memory status

Definition at line **230** of file **stm324x9i_eval_nor.c**.

References **NOR_DEVICE_ADDR**, **NOR_STATUS_ERROR**, **NOR_STATUS_OK**, **norHandle**, and **PROGRAM_TIMEOUT**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL SD Exported Functions

STM324x9I EVAL SD

Functions

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

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

Function Documentation

void BSP_SD_DetectCallback (void)

SD detect IT detection callback.

Definition at line 236 of file stm324x9i_eval_sd.c.

Referenced by BSP_SD_DetectIT().

void BSP_SD_DetectIT (void)

SD detect IT treatment.

Definition at line 222 of file stm324x9i_eval_sd.c.

References BSP_IO_ITClear(), BSP_SD_DetectCallback(), and BSP_SD_ITConfig().

void BSP_SD_DMA_Rx_IRQHandler (void)

Handles SD DMA Rx transfer interrupt request.

Definition at line 486 of file stm324x9i_eval_sd.c.

References uSdHandle.

void BSP_SD_DMA_Tx_IRQHandler (void)

Handles SD DMA Tx transfer interrupt request.

Definition at line 478 of file stm324x9i_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 **357** of file **stm324x9i_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 **508** of file **stm324x9i_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 **499** of file **stm324x9i_eval_sd.c**.

References **uSdHandle**.

uint8_t BSP_SD_Init (void)

Initializes the SD card device.

Return values:

SD status

Definition at line **144** of file **stm324x9i_eval_sd.c**.

References **BSP_IO_Init()**, **BSP_SD_IsDetected()**, **MSD_ERROR**, **MSD_OK**, **SD_MspInit()**, **SD_PRESENT**, **uSdCardInfo**, and **uSdHandle**.

void BSP_SD_IRQHandler (void)

Handles SD card interrupt request.

Definition at line **470** of file **stm324x9i_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 **207** of file **stm324x9i_eval_sd.c**.

References **BSP_IO_ReadPin()**, **SD_DETECT_PIN**, **SD_NOT_PRESENT**, and **SD_PRESENT**.

Referenced by **BSP_SD_Init()**.

uint8_t BSP_SD_ITConfig (void)

Configures Interrupt mode for SD detection pin.

Return values:

Returns 0

Definition at line **195** of file **stm324x9i_eval_sd.c**.

References **BSP_IO_ConfigPin()**, and **SD_DETECT_PIN**.

Referenced by **BSP_SD_DetectIT()**.

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 **251** of file **stm324x9i_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 **291** of file **stm324x9i_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 **271** of file **stm324x9i_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 **325** of file **stm324x9i_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, **SD_DATATIMEOUT**, and **uSdHandle**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL SD Private Functions

STM324x9I EVAL SD

Functions

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

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

Function Documentation

__weak void BSP_SD_DetectCallback (void)

SD detect IT detection callback.

Definition at line **236** of file **stm324x9i_eval_sd.c**.

Referenced by **BSP_SD_DetectIT()**.

void BSP_SD_DetectIT (void)

SD detect IT treatment.

Definition at line **222** of file **stm324x9i_eval_sd.c**.

References **BSP_IO_ITClear()**, **BSP_SD_DetectCallback()**, and **BSP_SD_ITConfig()**.

void BSP_SD_DMA_Rx_IRQHandler (void)

Handles SD DMA Rx transfer interrupt request.

Definition at line **486** of file **stm324x9i_eval_sd.c**.

References **uSdHandle**.

void BSP_SD_DMA_Tx_IRQHandler (void)

Handles SD DMA Tx transfer interrupt request.

Definition at line **478** of file **stm324x9i_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 **357** of file **stm324x9i_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 **508** of file **stm324x9i_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 **499** of file **stm324x9i_eval_sd.c**.

References **uSdHandle**.

uint8_t BSP_SD_Init (void)

Initializes the SD card device.

Return values:

SD status

Definition at line **144** of file **stm324x9i_eval_sd.c**.

References **BSP_IO_Init()**, **BSP_SD_IsDetected()**, **MSD_ERROR**, **MSD_OK**, **SD_MspInit()**, **SD_PRESENT**, **uSdCardInfo**, and **uSdHandle**.

void BSP_SD_IRQHandler (void)

Handles SD card interrupt request.

Definition at line **470** of file **stm324x9i_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 **207** of file **stm324x9i_eval_sd.c**.

References **BSP_IO_ReadPin()**, **SD_DETECT_PIN**, **SD_NOT_PRESENT**, and **SD_PRESENT**.

Referenced by **BSP_SD_Init()**.

uint8_t BSP_SD_ITConfig (void)

Configures Interrupt mode for SD detection pin.

Return values:

Returns 0

Definition at line **195** of file **stm324x9i_eval_sd.c**.

References **BSP_IO_ConfigPin()**, and **SD_DETECT_PIN**.

Referenced by **BSP_SD_DetectIT()**.

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 **251** of file **stm324x9i_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 **291** of file **stm324x9i_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 **271** of file **stm324x9i_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 **325** of file **stm324x9i_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, **SD_DATATIMEOUT**, and **uSdHandle**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL
SDRAM Exported Functions

STM324x9I EVAL SDRAM

Functions

uint8_t	**BSP_SDRAM_Init** (void) Initializes 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.

Function Documentation

void BSP_SDRAM_DMA_IRQHandler (void)

Handles SDRAM DMA transfer interrupt request.

Definition at line 340 of file stm324x9i_eval_sdram.c.

References sdramHandle.

uint8_t BSP_SDRAM_Init (void)

Initializes the SDRAM device.

Return values:

SDRAM status

Definition at line 142 of file stm324x9i_eval_sdram.c.

References BSP_SDRAM_Initialization_sequence(), REFRESH_COUNT, SDCLOCK_PERIOD, SDRAM_ERROR, SDRAM_MEMORY_WIDTH, SDRAM_MspInit(), SDRAM_OK, sdramHandle, and Timing.

Referenced by BSP_LCD_InitEx().

void BSP_SDRAM_Initialization_sequence (uint32_t RefreshCount

Programs the SDRAM device.

Parameters:

RefreshCount,: SDRAM refresh counter value

Definition at line 189 of file stm324x9i_eval_sdram.c.

References **Command**, **SDRAM_MODEREG_BURST_LENGTH_1**, **SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL**, **SDRAM_MODEREG_CAS_LATENCY_3**, **SDRAM_MODEREG_OPERATING_MODE_STANDARD**, **SDRAM_MODEREG_WRITEBURST_MODE_SINGLE**, **SDRAM_TIMEOUT**, and **sdramHandle**.

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 **251** of file **stm324x9i_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 **270** of file **stm324x9i_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 **325** of file **stm324x9i_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 **289** of file **stm324x9i_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 **308** of file **stm324x9i_eval_sdram.c**.

References **SDRAM_ERROR**, **SDRAM_OK**, and **sdramHandle**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL
SDRAM Private Functions

STM324x9I EVAL SDRAM

Functions

uint8_t	**BSP_SDRAM_Init** (void) Initializes 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.

Function Documentation

void BSP_SDRAM_DMA_IRQHandler (void)

Handles SDRAM DMA transfer interrupt request.

Definition at line 340 of file stm324x9i_eval_sdram.c.

References sdramHandle.

uint8_t BSP_SDRAM_Init (void)

Initializes the SDRAM device.

Return values:

SDRAM status

Definition at line 142 of file stm324x9i_eval_sdram.c.

References BSP_SDRAM_Initialization_sequence(), REFRESH_COUNT, SDCLOCK_PERIOD, SDRAM_ERROR, SDRAM_MEMORY_WIDTH, SDRAM_MspInit(), SDRAM_OK, sdramHandle, and Timing.

Referenced by BSP_LCD_InitEx().

void BSP_SDRAM_Initialization_sequence (uint32_t RefreshCount

Programs the SDRAM device.

Parameters:

RefreshCount,: SDRAM refresh counter value

Definition at line 189 of file stm324x9i_eval_sdram.c.

References **Command**, **SDRAM_MODEREG_BURST_LENGTH_1**, **SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL**, **SDRAM_MODEREG_CAS_LATENCY_3**, **SDRAM_MODEREG_OPERATING_MODE_STANDARD**, **SDRAM_MODEREG_WRITEBURST_MODE_SINGLE**, **SDRAM_TIMEOUT**, and **sdramHandle**.

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 **251** of file **stm324x9i_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 **270** of file **stm324x9i_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 **325** of file **stm324x9i_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 **289** of file **stm324x9i_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 **308** of file **stm324x9i_eval_sdram.c**.

References **SDRAM_ERROR**, **SDRAM_OK**, and **sdramHandle**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL
SRAM Exported Functions

STM324x9I EVAL SRAM

Functions

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

Function Documentation

void BSP_SRAM_DMA_IRQHandler (void)

Handles SRAM DMA transfer interrupt request.

Definition at line 253 of file stm324x9i_eval_sram.c.

References sramHandle.

uint8_t BSP_SRAM_Init (void)

Initializes the SRAM device.

Return values:

SRAM status

Definition at line 133 of file stm324x9i_eval_sram.c.

References CONTINUOUSCLOCK_FEATURE, SRAM_BURSTACCESS, SRAM_ERROR, SRAM_MEMORY_WIDTH, SRAM_MspInit(), SRAM_OK, SRAM_WRITEBURST, sramHandle, and Timing.

uint8_t BSP_SRAM_ReadData (uint32_t uwStartAddress, uint16_t * pData, uint32_t uwDataSize)

Reads an amount of data from the SRAM device in polling mode.

Parameters:

uwStartAddress,: Read start address

pData,: Pointer to data to be read
uwDataSize,: Size of read data from the memory

Return values:
SRAM status

Definition at line **181** of file **stm324x9i_eval_sram.c**.

References **SRAM_ERROR**, **SRAM_OK**, and **sramHandle**.

uint8_t BSP_SRAM_ReadData_DMA (uint32_t uwStartAddress,
uint16_t * pData,
uint32_t uwDataSize
)

Reads an amount of data from the SRAM device in DMA mode.

Parameters:
uwStartAddress,: Read start address
pData,: Pointer to data to be read
uwDataSize,: Size of read data from the memory

Return values:
SRAM status

Definition at line **200** of file **stm324x9i_eval_sram.c**.

References **SRAM_ERROR**, **SRAM_OK**, and **sramHandle**.

uint8_t BSP_SRAM_WriteData (uint32_t uwStartAddress,
uint16_t * pData,
uint32_t uwDataSize
)

Writes an amount of data from the SRAM device in polling mode.

Parameters:

uwStartAddress,:	Write start address
pData,:	Pointer to data to be written
uwDataSize,:	Size of written data from the memory

Return values:

SRAM status

Definition at line **219** of file **stm324x9i_eval_sram.c**.

References **SRAM_ERROR**, **SRAM_OK**, and **sramHandle**.

uint8_t BSP_SRAM_WriteData_DMA (uint32_t uwStartAddress,
uint16_t * pData,
uint32_t uwDataSize
)

Writes an amount of data from the SRAM device in DMA mode.

Parameters:

uwStartAddress,:	Write start address
pData,:	Pointer to data to be written
uwDataSize,:	Size of written data from the memory

Return values:

SRAM status

Definition at line **238** of file **stm324x9i_eval_sram.c**.

References **SRAM_ERROR**, **SRAM_OK**, and **sramHandle**.

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP
User Manual by  1.7.6.1

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL
SRAM Private Functions

STM324x9I EVAL SRAM

Functions

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

Function Documentation

void BSP_SRAM_DMA_IRQHandler (void)

Handles SRAM DMA transfer interrupt request.

Definition at line 253 of file stm324x9i_eval_sram.c.

References sramHandle.

uint8_t BSP_SRAM_Init (void)

Initializes the SRAM device.

Return values:

SRAM status

Definition at line 133 of file stm324x9i_eval_sram.c.

References CONTINUOUSCLOCK_FEATURE, SRAM_BURSTACCESS, SRAM_ERROR, SRAM_MEMORY_WIDTH, SRAM_MspInit(), SRAM_OK, SRAM_WRITEBURST, sramHandle, and Timing.

uint8_t BSP_SRAM_ReadData (uint32_t uwStartAddress, uint16_t * pData, uint32_t uwDataSize)

Reads an amount of data from the SRAM device in polling mode.

Parameters:

uwStartAddress,: Read start address

pData,: Pointer to data to be read
uwDataSize,: Size of read data from the memory

Return values:
SRAM status

Definition at line **181** of file **stm324x9i_eval_sram.c**.

References **SRAM_ERROR**, **SRAM_OK**, and **sramHandle**.

uint8_t BSP_SRAM_ReadData_DMA (uint32_t uwStartAddress,
uint16_t * pData,
uint32_t uwDataSize
)

Reads an amount of data from the SRAM device in DMA mode.

Parameters:
uwStartAddress,: Read start address
pData,: Pointer to data to be read
uwDataSize,: Size of read data from the memory

Return values:
SRAM status

Definition at line **200** of file **stm324x9i_eval_sram.c**.

References **SRAM_ERROR**, **SRAM_OK**, and **sramHandle**.

uint8_t BSP_SRAM_WriteData (uint32_t uwStartAddress,
uint16_t * pData,
uint32_t uwDataSize
)

Writes an amount of data from the SRAM device in polling mode.

Parameters:

uwStartAddress,:	Write start address
pData,:	Pointer to data to be written
uwDataSize,:	Size of written data from the memory

Return values:

SRAM status

Definition at line **219** of file **stm324x9i_eval_sram.c**.

References **SRAM_ERROR**, **SRAM_OK**, and **sramHandle**.

uint8_t BSP_SRAM_WriteData_DMA (uint32_t uwStartAddress,
uint16_t * pData,
uint32_t uwDataSize
)

Writes an amount of data from the SRAM device in DMA mode.

Parameters:

uwStartAddress,:	Write start address
pData,:	Pointer to data to be written
uwDataSize,:	Size of written data from the memory

Return values:

SRAM status

Definition at line **238** of file **stm324x9i_eval_sram.c**.

References **SRAM_ERROR**, **SRAM_OK**, and **sramHandle**.

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP
User Manual by  1.7.6.1

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL TS
Private Functions

STM324x9I 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_ITConfig** (void) Configures and enables the touch screen interrupts.
uint8_t	**BSP_TS_ITGetStatus** (void) Gets the touch screen interrupt status.
uint8_t	**BSP_TS_GetState** (**TS_StateTypeDef** *TS_State) Returns status and positions of the touch screen.
void	**BSP_TS_ITClear** (void) Clears all touch screen interrupts.

Function Documentation

uint8_t BSP_TS_DeInit (void)

DeInitializes the TouchScreen.

Return values:

TS state

Definition at line **191** of file **stm324x9i_eval_ts.c**.

References **TS_OK**.

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 **230** of file **stm324x9i_eval_ts.c**.

References **I2C_Address**, **TS_StateTypeDef::TouchDetected**, **ts_driver**, **TS_OK**, **ts_orientation**, **TS_SWAP_X**, **TS_SWAP_XY**, **TS_SWAP_Y**, **ts_x_boundary**, **ts_y_boundary**, **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 **145** of file **stm324x9i_eval_ts.c**.

References **BSP_TS3510_IsDetected()**, **EXC7200_I2C_ADDRESS**, **I2C_Address**, **IOE_Init()**, **TS3510_I2C_ADDRESS**, **ts_driver**, **TS_I2C_ADDRESS**, **TS_OK**, **ts_orientation**, **TS_SWAP_NONE**, **TS_SWAP_Y**, **ts_x_boundary**, and **ts_y_boundary**.

void BSP_TS_ITClear (void)

Clears all touch screen interrupts.

Definition at line **277** of file **stm324x9i_eval_ts.c**.

References **BSP_IO_ITClear()**, **I2C_Address**, 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 **201** of file **stm324x9i_eval_ts.c**.

References **BSP_IO_ConfigPin()**, **BSP_IO_Init()**, **I2C_Address**,

ts_driver, **TS_INT_PIN**, 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 **219** of file **stm324x9i_eval_ts.c**.

References **I2C_Address**, and **ts_driver**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL TS Exported Functions

STM324x9I 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_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.

Function Documentation

uint8_t BSP_TS_DeInit (void)

DeInitializes the TouchScreen.

Return values:

TS state

Definition at line **191** of file **stm324x9i_eval_ts.c**.

References **TS_OK**.

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 **230** of file **stm324x9i_eval_ts.c**.

References **I2C_Address**, **TS_StateTypeDef::TouchDetected**, **ts_driver**, **TS_OK**, **ts_orientation**, **TS_SWAP_X**, **TS_SWAP_XY**, **TS_SWAP_Y**, **ts_x_boundary**, **ts_y_boundary**, **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 **145** of file **stm324x9i_eval_ts.c**.

References **BSP_TS3510_IsDetected()**, **EXC7200_I2C_ADDRESS**, **I2C_Address**, **IOE_Init()**, **TS3510_I2C_ADDRESS**, **ts_driver**, **TS_I2C_ADDRESS**, **TS_OK**, **ts_orientation**, **TS_SWAP_NONE**, **TS_SWAP_Y**, **ts_x_boundary**, and **ts_y_boundary**.

void BSP_TS_ITClear (void)

Clears all touch screen interrupts.

Definition at line **277** of file **stm324x9i_eval_ts.c**.

References **BSP_IO_ITClear()**, **I2C_Address**, 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 **201** of file **stm324x9i_eval_ts.c**.

References **BSP_IO_ConfigPin()**, **BSP_IO_Init()**, **I2C_Address**,

ts_driver, **TS_INT_PIN**, 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 **219** of file **stm324x9i_eval_ts.c**.

References **I2C_Address**, and **ts_driver**.

STM324x9I_EVAL BSP User Manual

Variables

STM324x9I EVAL LOW LEVEL Private Variables

STM324x9I EVAL LOW LEVEL

Variables

GPIO_TypeDef *	**GPIO_PORT** [**LEDn**]
const uint16_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 I2C_HandleTypeDef	**heval_I2c**

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 **116** of file **stm324x9i_eval.c**.

Referenced by **BSP_PB_Init()**.

const uint16_t BUTTON_PIN[BUTTONn]

Initial value:

```
 {WAKEUP_BUTTON_PIN,
                                    TAMPER_BUTT
ON_PIN,
                                    KEY_BUTTON_
PIN}
```

Definition at line **112** of file **stm324x9i_eval.c**.

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

GPIO_TypeDef* BUTTON_PORT[BUTTONn]

Initial value:

```
 {WAKEUP_BUTTON_GPIO_PORT,
                                    TAMPER_BUTT
ON_GPIO_PORT,
                                    KEY_BUTTON_
```

```
GPIO_PORT}
```

Definition at line **108** of file **stm324x9i_eval.c**.

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

const uint16_t COM_RX_AF[COMn] = {EVAL_COM1_RX_AF}

Definition at line **132** of file **stm324x9i_eval.c**.

Referenced by **BSP_COM_Init()**.

const uint16_t COM_RX_PIN[COMn] = {EVAL_COM1_RX_PIN}

Definition at line **128** of file **stm324x9i_eval.c**.

Referenced by **BSP_COM_Init()**.

GPIO_TypeDef* COM_RX_PORT[COMn] = {EVAL_COM1_RX_GPIO_

Definition at line **124** of file **stm324x9i_eval.c**.

Referenced by **BSP_COM_Init()**.

const uint16_t COM_TX_AF[COMn] = {EVAL_COM1_TX_AF}

Definition at line **130** of file **stm324x9i_eval.c**.

Referenced by **BSP_COM_Init()**.

const uint16_t COM_TX_PIN[COMn] = {EVAL_COM1_TX_PIN}

Definition at line **126** of file **stm324x9i_eval.c**.

Referenced by **BSP_COM_Init()**.

GPIO_TypeDef* COM_TX_PORT[COMn] = {EVAL_COM1_TX_GPIO_

Definition at line **122** of file **stm324x9i_eval.c**.

Referenced by **BSP_COM_Init()**.

USART_TypeDef* COM_USART[COMn] = {EVAL_COM1}

Definition at line **120** of file **stm324x9i_eval.c**.

Referenced by **BSP_COM_Init()**.

const uint16_t GPIO_PIN[LEDn]

Initial value:

```
{LED1_PIN,
                                    LED2_PIN,
                                    LED3_PIN,
                                    LED4_PIN}
```

Definition at line **103** of file **stm324x9i_eval.c**.

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

GPIO_TypeDef* GPIO_PORT[LEDn]

Initial value:

```
{LED1_GPIO_PORT,
```

```
                                        LED2_GPIO_PORT,
                                        LED3_GPIO_PORT,
                                        LED4_GPIO_PORT}
```

Definition at line **98** of file **stm324x9i_eval.c**.

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

I2C_HandleTypeDef heval_I2c `[static]`

Definition at line **134** of file **stm324x9i_eval.c**.

Referenced by **BSP_TS3510_IsDetected()**, **I2Cx_Error()**, **I2Cx_Init()**, **I2Cx_IsDeviceReady()**, **I2Cx_Read()**, **I2Cx_ReadMultiple()**, **I2Cx_Write()**, and **I2Cx_WriteMultiple()**.

STM324x9I_EVAL BSP User Manual

Enumerations

STM324x9I EVAL LOW LEVEL Exported Types

STM324x9I EVAL LOW LEVEL

Enumerations

enum	**Led_TypeDef** { **LED1** = 0, **LED2** = 1, **LED3** = 2, **LED4** = 3 }
enum	**Button_TypeDef** { **BUTTON_WAKEUP** = 0, **BUTTON_TAMPER** = 1, **BUTTON_KEY** = 2 }
enum	**ButtonMode_TypeDef** { **BUTTON_MODE_GPIO** = 0, **BUTTON_MODE_EXTI** = 1 }
enum	**JOYMode_TypeDef** { **JOY_MODE_GPIO** = 0, **JOY_MODE_EXTI** = 1 }
enum	**JOYState_TypeDef** { **JOY_NONE** = 0, **JOY_SEL** = 1, **JOY_DOWN** = 2, **JOY_LEFT** = 3, **JOY_RIGHT** = 4, **JOY_UP** = 5 }
enum	**COM_TypeDef** { **COM1** = 0, **COM2** = 1 }

Enumeration Type Documentation

enum Button_TypeDef

Enumerator:

BUTTON_WAKEUP

BUTTON_TAMPER

BUTTON_KEY

Definition at line **73** of file **stm324x9i_eval.h**.

enum ButtonMode_TypeDef

Enumerator:

BUTTON_MODE_GPIO

BUTTON_MODE_EXTI

Definition at line **80** of file **stm324x9i_eval.h**.

enum COM_TypeDef

Enumerator:

COM1

COM2

Definition at line **102** of file **stm324x9i_eval.h**.

enum JOYMode_TypeDef

Enumerator:

JOY_MODE_GPIO

JOY_MODE_EXTI

Definition at line **86** of file **stm324x9i_eval.h**.

enum JOYState_TypeDef

Enumerator:

JOY_NONE
JOY_SEL
JOY_DOWN
JOY_LEFT
JOY_RIGHT
JOY_UP

Definition at line **92** of file **stm324x9i_eval.h**.

enum Led_TypeDef

Enumerator:

LED1
LED2
LED3
LED4

Definition at line **65** of file **stm324x9i_eval.h**.

STM324x9I_EVAL BSP User Manual

Defines

STM324x9I EVAL LOW LEVEL BUTTON

STM324x9I EVAL LOW LEVEL Exported Constants

Defines

#define	**BUTTONn** 3
#define	**WAKEUP_BUTTON_PIN** GPIO_PIN_0 Wakeup push-button.
#define	**WAKEUP_BUTTON_GPIO_PORT** GPIOA
#define	**WAKEUP_BUTTON_GPIO_CLK_ENABLE**() __GPIOA_CLK
#define	**WAKEUP_BUTTON_GPIO_CLK_DISABLE**() __GPIOA_CLK
#define	**WAKEUP_BUTTON_EXTI_IRQn** EXTI0_IRQn
#define	**TAMPER_BUTTON_PIN** GPIO_PIN_13 Tamper push-button.
#define	**TAMPER_BUTTON_GPIO_PORT** GPIOC
#define	**TAMPER_BUTTON_GPIO_CLK_ENABLE**() __GPIOC_CLK_
#define	**TAMPER_BUTTON_GPIO_CLK_DISABLE**() __GPIOC_CLK
#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**() __GPIOC_CLK_ENA
#define	**KEY_BUTTON_GPIO_CLK_DISABLE**() __GPIOC_CLK_DIS
#define	**KEY_BUTTON_EXTI_IRQn** EXTI15_10_IRQn
#define	**BUTTONx_GPIO_CLK_ENABLE**(__INDEX__)
#define	**BUTTONx_GPIO_CLK_DISABLE**(__INDEX__)

Define Documentation

#define BUTTONn 3

Definition at line 168 of file stm324x9i_eval.h.

#define BUTTONx_GPIO_CLK_DISABLE (__INDEX__)

Value:

```
do{if((__INDEX__) == 0) WAKEUP_BUTTON_GPIO_CLK_DI
SABLE(); else \
                                                  if
((__INDEX__) == 1) TAMPER_BUTTON_GPIO_CLK_DISABLE
(); else \
                                                  if
 ((__INDEX__) == 2) KEY_BUTTON_GPIO_CLK_DISABLE()
; \
                                                  }w
hile(0)
```

Definition at line 201 of file stm324x9i_eval.h.

#define BUTTONx_GPIO_CLK_ENABLE (__INDEX__)

Value:

```
do{if((__INDEX__) == 0) WAKEUP_BUTTON_GPIO_CLK_EN
ABLE(); else \
                                                  if
((__INDEX__) == 1) TAMPER_BUTTON_GPIO_CLK_ENABLE(
); else \
                                                  if
 ((__INDEX__) == 2) KEY_BUTTON_GPIO_CLK_ENABLE();
 \
                                                  }w
```

```
hile(0)
```

Definition at line **197** of file **stm324x9i_eval.h**.

Referenced by **BSP_PB_Init()**.

#define KEY_BUTTON_EXTI_IRQn EXTI15_10_IRQn

Definition at line **195** of file **stm324x9i_eval.h**.

#define KEY_BUTTON_GPIO_CLK_DISABLE ( ) __GPIOC_CLK_D

Definition at line **194** of file **stm324x9i_eval.h**.

#define KEY_BUTTON_GPIO_CLK_ENABLE ( ) __GPIOC_CLK_EI

Definition at line **193** of file **stm324x9i_eval.h**.

#define KEY_BUTTON_GPIO_PORT GPIOC

Definition at line **192** of file **stm324x9i_eval.h**.

#define KEY_BUTTON_PIN GPIO_PIN_13

Key push-button.

Definition at line **191** of file **stm324x9i_eval.h**.

#define TAMPER_BUTTON_EXTI_IRQn EXTI15_10_IRQn

Definition at line **186** of file **stm324x9i_eval.h**.

#define TAMPER_BUTTON_GPIO_CLK_DISABLE ( ) __GPIOC_CL

Definition at line 185 of file stm324x9i_eval.h.

#define TAMPER_BUTTON_GPIO_CLK_ENABLE ( ) __GPIOC_CL

Definition at line 184 of file stm324x9i_eval.h.

#define TAMPER_BUTTON_GPIO_PORT GPIOC

Definition at line 183 of file stm324x9i_eval.h.

#define TAMPER_BUTTON_PIN GPIO_PIN_13

Tamper push-button.

Definition at line 182 of file stm324x9i_eval.h.

#define WAKEUP_BUTTON_EXTI_IRQn EXTI0_IRQn

Definition at line 177 of file stm324x9i_eval.h.

#define WAKEUP_BUTTON_GPIO_CLK_DISABLE ( ) __GPIOA_CL

Definition at line 176 of file stm324x9i_eval.h.

#define WAKEUP_BUTTON_GPIO_CLK_ENABLE ( ) __GPIOA_CL

Definition at line 175 of file stm324x9i_eval.h.

#define WAKEUP_BUTTON_GPIO_PORT GPIOA

Definition at line 174 of file stm324x9i_eval.h.

#define WAKEUP_BUTTON_PIN GPIO_PIN_0

Wakeup push-button.

Definition at line 173 of file stm324x9i_eval.h.

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL CAMERA Private Variables

STM324x9I EVAL CAMERA

Variables

static DCMI_HandleTypeDef	**hdcmi_eval**
CAMERA_DrvTypeDef *	**camera_drv**
uint32_t	**current_resolution**

Variable Documentation

CAMERA_DrvTypeDef* camera_drv

Definition at line 106 of file stm324x9i_eval_camera.c.

Referenced by BSP_CAMERA_BlackWhiteConfig(), BSP_CAMERA_ColorEffectConfig(), BSP_CAMERA_ContrastBrightnessConfig(), and BSP_CAMERA_Init().

uint32_t current_resolution

Definition at line 107 of file stm324x9i_eval_camera.c.

Referenced by BSP_CAMERA_ContinuousStart(), BSP_CAMERA_Init(), and BSP_CAMERA_SnapshotStart().

DCMI_HandleTypeDef hdcmi_eval [static]

Definition at line 105 of file stm324x9i_eval_camera.c.

Referenced by BSP_CAMERA_ContinuousStart(), BSP_CAMERA_DMA_IRQHandler(), BSP_CAMERA_Init(), BSP_CAMERA_IRQHandler(), BSP_CAMERA_Resume(), BSP_CAMERA_SnapshotStart(), BSP_CAMERA_Stop(), BSP_CAMERA_Suspend(), and DCMI_MspInit().

STM324x9I_EVAL BSP User Manual

Defines | Enumerations

STM324x9I EVAL CAMERA Exported Types

STM324x9I 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 stm324x9i_eval_camera.h.

#define RESOLUTION_R320x240 CAMERA_R320x240 /* QVGA Re

Definition at line 81 of file stm324x9i_eval_camera.h.

#define RESOLUTION_R480x272 CAMERA_R480x272 /* 480x272 F

Definition at line 82 of file stm324x9i_eval_camera.h.

#define RESOLUTION_R640x480 CAMERA_R640x480 /* VGA Resc

Definition at line 83 of file stm324x9i_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 **stm324x9i_eval_camera.h**.

STM324x9I_EVAL BSP User Manual

Defines

CODEC AudioFrame SLOT TDMMode

STM324x9I 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 **stm324x9i_eval_audio.h**.

Referenced by **SAIx_Init()**.

#define CODEC_AUDIOFRAME_SLOT_02 SAI_SLOTACTIVE_0 | S

Definition at line **90** of file **stm324x9i_eval_audio.h**.

#define CODEC_AUDIOFRAME_SLOT_13 SAI_SLOTACTIVE_1 | S

Definition at line **92** of file **stm324x9i_eval_audio.h**.

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL SDRAM Private Variables

STM324x9I 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 stm324x9i_eval_sdram.c.

Referenced by BSP_SDRAM_Initialization_sequence().

SDRAM_HandleTypeDef sdramHandle [static]

Definition at line 119 of file stm324x9i_eval_sdram.c.

Referenced by BSP_SDRAM_DMA_IRQHandler(), BSP_SDRAM_Init(), BSP_SDRAM_Initialization_sequence(), BSP_SDRAM_ReadData(), BSP_SDRAM_ReadData_DMA(), BSP_SDRAM_Sendcmd(), BSP_SDRAM_WriteData(), BSP_SDRAM_WriteData_DMA(), and SDRAM_MspInit().

FMC_SDRAM_TimingTypeDef Timing [static]

Definition at line 120 of file stm324x9i_eval_sdram.c.

Referenced by BSP_SDRAM_Init().

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL CAMERA Private FunctionPrototypes

STM324x9I EVAL CAMERA

Functions

static void	**DCMI_MspInit** (void) Initializes the DCMI MSP.

Function Documentation

static void DCMI_MspInit (void) [static]

Initializes the DCMI MSP.

Definition at line **373** of file **stm324x9i_eval_camera.c**.

References **hdcmi_eval**.

Referenced by **BSP_CAMERA_Init()**.

STM324x9I_EVAL BSP User Manual

Defines

STM324x9I EVAL AUDIO Exported Macros

STM324x9I 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 **215** of file **stm324x9i_eval_audio.h**.

Referenced by **BSP_AUDIO_OUT_Play()**.

STM324x9I_EVAL BSP User Manual

Defines

STM324x9I EVAL
EEPROM Exported Constants

STM324x9I EVAL EEPROM

Defines

```
#define EEPROM_PAGESIZE 4
#define EEPROM_MAX_SIZE 0x2000 /* 64Kbit */
#define EEPROM_READ_TIMEOUT ((uint32_t)(1000))
#define EEPROM_WRITE_TIMEOUT ((uint32_t)(10))
#define EEPROM_MAX_TRIALS 3000
#define EEPROM_OK 0
#define EEPROM_FAIL 1
#define EEPROM_TIMEOUT 2
```

Define Documentation

#define EEPROM_FAIL 1

Definition at line 88 of file stm324x9i_eval_eeprom.h.

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

#define EEPROM_MAX_SIZE 0x2000 /* 64Kbit */

Definition at line 75 of file stm324x9i_eval_eeprom.h.

#define EEPROM_MAX_TRIALS 3000

Definition at line 85 of file stm324x9i_eval_eeprom.h.

Referenced by BSP_EEPROM_Init(), and BSP_EEPROM_WaitEepromStandbyState().

#define EEPROM_OK 0

Definition at line 87 of file stm324x9i_eval_eeprom.h.

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

#define EEPROM_PAGESIZE 4

Definition at line 74 of file stm324x9i_eval_eeprom.h.

Referenced by **BSP_EEPROM_WriteBuffer()**.

#define EEPROM_READ_TIMEOUT ((uint32_t)(1000))

Definition at line **80** of file **stm324x9i_eval_eeprom.h**.

#define EEPROM_TIMEOUT 2

Definition at line **89** of file **stm324x9i_eval_eeprom.h**.

Referenced by **BSP_EEPROM_WaitEepromStandbyState()**.

#define EEPROM_WRITE_TIMEOUT ((uint32_t)(10))

Definition at line **82** of file **stm324x9i_eval_eeprom.h**.

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL EEPROM Private Variables

STM324x9I EVAL EEPROM

Variables

__IO uint16_t	**EEPROMAddress** = 0
__IO uint32_t	**EEPROMTimeout** = **EEPROM_READ_TIMEOUT**
__IO uint16_t	**EEPROMDataRead**
__IO uint8_t	**EEPROMDataWrite**

Variable Documentation

__IO uint16_t EEPROMAddress = 0

Definition at line 128 of file stm324x9i_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 130 of file stm324x9i_eval_eeprom.c.

Referenced by BSP_EEPROM_ReadBuffer().

__IO uint8_t EEPROMDataWrite

Definition at line 131 of file stm324x9i_eval_eeprom.c.

Referenced by BSP_EEPROM_WritePage().

__IO uint32_t EEPROMTimeout = EEPROM_READ_TIMEOUT

Definition at line 129 of file stm324x9i_eval_eeprom.c.

STM324x9I_EVAL BSP User Manual

Variables

STM324x9I EVAL TS
Private Variables

STM324x9I EVAL TS

Variables

static TS_DrvTypeDef *	**ts_driver**
static uint16_t	**ts_x_boundary**
static uint16_t	**ts_y_boundary**
static uint8_t	**ts_orientation**
static uint8_t	**I2C_Address**

Variable Documentation

uint8_t I2C_Address [static]

Definition at line 122 of file stm324x9i_eval_ts.c.

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

TS_DrvTypeDef* ts_driver [static]

Definition at line 119 of file stm324x9i_eval_ts.c.

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

uint8_t ts_orientation [static]

Definition at line 121 of file stm324x9i_eval_ts.c.

Referenced by BSP_TS_GetState(), and BSP_TS_Init().

uint16_t ts_x_boundary [static]

Definition at line 120 of file stm324x9i_eval_ts.c.

Referenced by BSP_TS_GetState(), and BSP_TS_Init().

uint16_t ts_y_boundary [static]

Definition at line **120** of file **stm324x9i_eval_ts.c**.

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

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL
AUDIO Private Function Prototypes

STM324x9I EVAL AUDIO

Functions

static void	**SAIx_MspInit** (void) Initializes SAI MSP.
static void	**I2Sx_MspInit** (void) AUDIO IN I2S MSP Init.

Function Documentation

static void I2Sx_MspInit (void) [static]

AUDIO IN I2S MSP Init.

Definition at line 907 of file stm324x9i_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 haudio_in_i2s.

Referenced by I2Sx_Init().

static void SAIx_MspInit (void) [static]

Initializes SAI MSP.

Definition at line 554 of file stm324x9i_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_SD_FS_AF,

AUDIO_SAIx_MCLK_SCK_SD_FS_ENABLE, **AUDIO_SAIx_MCLK_SCK_SD_FS_GPIO_PORT**, **AUDIO_SAIx_SCK_PIN**, **AUDIO_SAIx_SD_PIN**, and **haudio_out_sai**.

Referenced by **SAIx_Init()**.

STM324x9I_EVAL BSP User Manual

Variables

STM324x9I EVAL IO
Private Variables

STM324x9I EVAL IO

Variables

static IO_DrvTypeDef *	**io_driver**

Variable Documentation

IO_DrvTypeDef* io_driver [static]

Definition at line 111 of file stm324x9i_eval_io.c.

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

STM324x9I_EVAL BSP User Manual

Defines

STM324x9I EVAL IO Exported Constants

STM324x9I EVAL IO

Defines

#define **IO_PIN_0**	0x0001
#define **IO_PIN_1**	0x0002
#define **IO_PIN_2**	0x0004
#define **IO_PIN_3**	0x0008
#define **IO_PIN_4**	0x0010
#define **IO_PIN_5**	0x0020
#define **IO_PIN_6**	0x0040
#define **IO_PIN_7**	0x0080
#define **IO_PIN_8**	0x0100
#define **IO_PIN_9**	0x0200
#define **IO_PIN_10**	0x0400
#define **IO_PIN_11**	0x0800
#define **IO_PIN_12**	0x1000
#define **IO_PIN_13**	0x2000
#define **IO_PIN_14**	0x4000
#define **IO_PIN_15**	0x8000
#define **IO_PIN_ALL**	0xFFFF

Define Documentation

#define IO_PIN_0 0x0001

Definition at line 88 of file stm324x9i_eval_io.h.

#define IO_PIN_1 0x0002

Definition at line 89 of file stm324x9i_eval_io.h.

#define IO_PIN_10 0x0400

Definition at line 98 of file stm324x9i_eval_io.h.

#define IO_PIN_11 0x0800

Definition at line 99 of file stm324x9i_eval_io.h.

#define IO_PIN_12 0x1000

Definition at line 100 of file stm324x9i_eval_io.h.

#define IO_PIN_13 0x2000

Definition at line 101 of file stm324x9i_eval_io.h.

#define IO_PIN_14 0x4000

Definition at line 102 of file stm324x9i_eval_io.h.

#define IO_PIN_15 0x8000

Definition at line 103 of file stm324x9i_eval_io.h.

#define IO_PIN_2 0x0004

Definition at line 90 of file stm324x9i_eval_io.h.

#define IO_PIN_3 0x0008

Definition at line 91 of file stm324x9i_eval_io.h.

#define IO_PIN_4 0x0010

Definition at line 92 of file stm324x9i_eval_io.h.

#define IO_PIN_5 0x0020

Definition at line 93 of file stm324x9i_eval_io.h.

#define IO_PIN_6 0x0040

Definition at line 94 of file stm324x9i_eval_io.h.

#define IO_PIN_7 0x0080

Definition at line 95 of file stm324x9i_eval_io.h.

#define IO_PIN_8 0x0100

Definition at line 96 of file stm324x9i_eval_io.h.

#define IO_PIN_9 0x0200

Definition at line 97 of file stm324x9i_eval_io.h.

#define IO_PIN_ALL 0xFFFF

Definition at line 104 of file stm324x9i_eval_io.h.

Referenced by BSP_IO_Init().

STM324x9I_EVAL BSP User Manual

Defines

STM324x9I EVAL LCD Exported Constants

STM324x9I EVAL LCD

Defines

#define	**MAX_LAYER_NUMBER** 2
#define	**LCD_LayerCfgTypeDef** LTDC_LayerCfgTypeDef
#define	**LCD_OK** 0x00 LCD status structure definition.
#define	**LCD_ERROR** 0x01
#define	**LCD_TIMEOUT** 0x02
#define	**LCD_FB_START_ADDRESS** ((uint32_t)0xC0000000) LCD FB_StartAddress.
#define	**LCD_MAX_PCLK** ((uint8_t)0x00)
#define	**LCD_MIN_PCLK** ((uint8_t)0x01)
#define	**LCD_COLOR_BLUE** 0xFF0000FF LCD color.
#define	**LCD_COLOR_GREEN** 0xFF00FF00
#define	**LCD_COLOR_RED** 0xFFFF0000
#define	**LCD_COLOR_CYAN** 0xFF00FFFF
#define	**LCD_COLOR_MAGENTA** 0xFFFF00FF
#define	**LCD_COLOR_YELLOW** 0xFFFFFF00
#define	**LCD_COLOR_LIGHTBLUE** 0xFF8080FF
#define	**LCD_COLOR_LIGHTGREEN** 0xFF80FF80
#define	**LCD_COLOR_LIGHTRED** 0xFFFF8080
#define	**LCD_COLOR_LIGHTCYAN** 0xFF80FFFF
#define	**LCD_COLOR_LIGHTMAGENTA** 0xFFFF80FF
#define	**LCD_COLOR_LIGHTYELLOW** 0xFFFFFF80
#define	**LCD_COLOR_DARKBLUE** 0xFF000080
#define	**LCD_COLOR_DARKGREEN** 0xFF008000
#define	**LCD_COLOR_DARKRED** 0xFF800000
#define	**LCD_COLOR_DARKCYAN** 0xFF008080
#define	**LCD_COLOR_DARKMAGENTA** 0xFF800080
#define	**LCD_COLOR_DARKYELLOW** 0xFF808000
#define	**LCD_COLOR_WHITE** 0xFFFFFFFF
#define	**LCD_COLOR_LIGHTGRAY** 0xFFD3D3D3

#define	**LCD_COLOR_GRAY** 0xFF808080
#define	**LCD_COLOR_DARKGRAY** 0xFF404040
#define	**LCD_COLOR_BLACK** 0xFF000000
#define	**LCD_COLOR_BROWN** 0xFFA52A2A
#define	**LCD_COLOR_ORANGE** 0xFFFFA500
#define	**LCD_COLOR_TRANSPARENT** 0xFF000000
#define	**LCD_DEFAULT_FONT** Font24 LCD default font.

Define Documentation

#define LCD_COLOR_BLACK 0xFF000000

Definition at line 156 of file stm324x9i_eval_lcd.h.

Referenced by BSP_LCD_LayerDefaultInit().

#define LCD_COLOR_BLUE 0xFF0000FF

LCD color.

Definition at line 134 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_BROWN 0xFFA52A2A

Definition at line 157 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_CYAN 0xFF00FFFF

Definition at line 137 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_DARKBLUE 0xFF000080

Definition at line 146 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_DARKCYAN 0xFF008080

Definition at line 149 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_DARKGRAY 0xFF404040

Definition at line 155 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_DARKGREEN 0xFF008000

Definition at line 147 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_DARKMAGENTA 0xFF800080

Definition at line 150 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_DARKRED 0xFF800000

Definition at line 148 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_DARKYELLOW 0xFF808000

Definition at line 151 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_GRAY 0xFF808080

Definition at line 154 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_GREEN 0xFF00FF00

Definition at line 135 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_LIGHTBLUE 0xFF8080FF

Definition at line 140 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_LIGHTCYAN 0xFF80FFFF

Definition at line 143 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_LIGHTGRAY 0xFFD3D3D3

Definition at line 153 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_LIGHTGREEN 0xFF80FF80

Definition at line 141 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_LIGHTMAGENTA 0xFFFF80FF

Definition at line 144 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_LIGHTRED 0xFFFF8080

Definition at line 142 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_LIGHTYELLOW 0xFFFFFF80

Definition at line 145 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_MAGENTA 0xFFFF00FF

Definition at line 138 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_ORANGE 0xFFFFA500

Definition at line 158 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_RED 0xFFFF0000

Definition at line 136 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_TRANSPARENT 0xFF000000

Definition at line 159 of file stm324x9i_eval_lcd.h.

#define LCD_COLOR_WHITE 0xFFFFFFFF

Definition at line 152 of file stm324x9i_eval_lcd.h.

Referenced by BSP_LCD_LayerDefaultInit().

#define LCD_COLOR_YELLOW 0xFFFFFF00

Definition at line 139 of file stm324x9i_eval_lcd.h.

#define LCD_DEFAULT_FONT Font24

LCD default font.

Definition at line 164 of file stm324x9i_eval_lcd.h.

Referenced by BSP_LCD_InitEx().

#define LCD_ERROR 0x01

Definition at line 116 of file stm324x9i_eval_lcd.h.

#define LCD_FB_START_ADDRESS ((uint32_t)0xC0000000)

LCD FB_StartAddress.

Definition at line 122 of file stm324x9i_eval_lcd.h.

#define LCD_LayerCfgTypeDef LTDC_LayerCfgTypeDef

Definition at line 110 of file stm324x9i_eval_lcd.h.

Referenced by BSP_LCD_LayerDefaultInit().

#define LCD_MAX_PCLK ((uint8_t)0x00)

Definition at line 128 of file stm324x9i_eval_lcd.h.

Referenced by BSP_LCD_ClockConfig(), and BSP_LCD_Init().

#define LCD_MIN_PCLK ((uint8_t)0x01)

Definition at line 129 of file stm324x9i_eval_lcd.h.

#define LCD_OK 0x00

LCD status structure definition.

Definition at line 115 of file stm324x9i_eval_lcd.h.

Referenced by **BSP_LCD_InitEx()**.

#define LCD_TIMEOUT 0x02

Definition at line **117** of file **stm324x9i_eval_lcd.h**.

#define MAX_LAYER_NUMBER 2

Definition at line **108** of file **stm324x9i_eval_lcd.h**.

STM324x9I_EVAL BSP User Manual

Defines

STM324x9I EVAL LOW LEVEL LED

STM324x9I EVAL LOW LEVEL Exported Constants

Define for STM324x9I_EVAL board. More...

Defines

#define	**LEDn** 4
#define	**LED1_PIN** GPIO_PIN_6
#define	**LED1_GPIO_PORT** GPIOG
#define	**LED1_GPIO_CLK_ENABLE**() __GPIOG_CLK_ENABLE()
#define	**LED1_GPIO_CLK_DISABLE**() __GPIOG_CLK_DISABLE()
#define	**LED2_PIN** GPIO_PIN_7
#define	**LED2_GPIO_PORT** GPIOG
#define	**LED2_GPIO_CLK_ENABLE**() __GPIOG_CLK_ENABLE()
#define	**LED2_GPIO_CLK_DISABLE**() __GPIOG_CLK_DISABLE()
#define	**LED3_PIN** GPIO_PIN_10
#define	**LED3_GPIO_PORT** GPIOG
#define	**LED3_GPIO_CLK_ENABLE**() __GPIOG_CLK_ENABLE()
#define	**LED3_GPIO_CLK_DISABLE**() __GPIOG_CLK_DISABLE()
#define	**LED4_PIN** GPIO_PIN_12
#define	**LED4_GPIO_PORT** GPIOG
#define	**LED4_GPIO_CLK_ENABLE**() __GPIOG_CLK_ENABLE()
#define	**LED4_GPIO_CLK_DISABLE**() __GPIOG_CLK_DISABLE()
#define	**LEDx_GPIO_CLK_ENABLE**(__INDEX__)
#define	**LEDx_GPIO_CLK_DISABLE**(__INDEX__)

Detailed Description

Define for STM324x9I_EVAL board.

Define Documentation

#define LED1_GPIO_CLK_DISABLE ( ) __GPIOG_CLK_DISABLE(

Definition at line **130** of file **stm324x9i_eval.h**.

#define LED1_GPIO_CLK_ENABLE ( ) __GPIOG_CLK_ENABLE()

Definition at line **129** of file **stm324x9i_eval.h**.

#define LED1_GPIO_PORT GPIOG

Definition at line **128** of file **stm324x9i_eval.h**.

#define LED1_PIN GPIO_PIN_6

Definition at line **127** of file **stm324x9i_eval.h**.

#define LED2_GPIO_CLK_DISABLE ( ) __GPIOG_CLK_DISABLE(

Definition at line **136** of file **stm324x9i_eval.h**.

#define LED2_GPIO_CLK_ENABLE ( ) __GPIOG_CLK_ENABLE()

Definition at line **135** of file **stm324x9i_eval.h**.

#define LED2_GPIO_PORT GPIOG

Definition at line **134** of file **stm324x9i_eval.h**.

#define LED2_PIN GPIO_PIN_7

Definition at line 133 of file stm324x9i_eval.h.

#define LED3_GPIO_CLK_DISABLE ( ) __GPIOG_CLK_DISABLE(

Definition at line 141 of file stm324x9i_eval.h.

#define LED3_GPIO_CLK_ENABLE ( ) __GPIOG_CLK_ENABLE()

Definition at line 140 of file stm324x9i_eval.h.

#define LED3_GPIO_PORT GPIOG

Definition at line 139 of file stm324x9i_eval.h.

#define LED3_PIN GPIO_PIN_10

Definition at line 138 of file stm324x9i_eval.h.

#define LED4_GPIO_CLK_DISABLE ( ) __GPIOG_CLK_DISABLE(

Definition at line 146 of file stm324x9i_eval.h.

#define LED4_GPIO_CLK_ENABLE ( ) __GPIOG_CLK_ENABLE()

Definition at line 145 of file stm324x9i_eval.h.

#define LED4_GPIO_PORT GPIOG

Definition at line 144 of file stm324x9i_eval.h.

#define LED4_PIN GPIO_PIN_12

Definition at line 143 of file stm324x9i_eval.h.

#define LEDn 4

Definition at line 125 of file stm324x9i_eval.h.

#define LEDx_GPIO_CLK_DISABLE (__INDEX__)

Value:

```
do{if((__INDEX__) == 0) LED1_GPIO_CLK_DISABLE();
else \
                                                 if((
__INDEX__) == 1) LED2_GPIO_CLK_DISABLE(); else \
                                                 if((
__INDEX__) == 2) LED3_GPIO_CLK_DISABLE(); else \
                                                 if((
__INDEX__) == 3) LED4_GPIO_CLK_DISABLE(); \
                                                 }whi
le(0)
```

Definition at line 154 of file stm324x9i_eval.h.

#define LEDx_GPIO_CLK_ENABLE (__INDEX__)

Value:

```
do{if((__INDEX__) == 0) LED1_GPIO_CLK_ENABLE(); e
lse \
```

```
                                                  if((_
_INDEX__) == 1) LED2_GPIO_CLK_ENABLE(); else \
                                                  if((_
_INDEX__) == 2) LED3_GPIO_CLK_ENABLE(); else \
                                                  if((_
_INDEX__) == 3) LED4_GPIO_CLK_ENABLE(); \
                                                  }whil
e(0)
```

Definition at line **148** of file **stm324x9i_eval.h**.

Referenced by **BSP_LED_Init()**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL LCD
Private FunctionPrototypes

STM324x9I EVAL LCD

Functions

static void	**MspInit** (void) Initializes the LTDC MSP.

Function Documentation

static void MspInit (void) [static]

Initializes the LTDC MSP.

Definition at line **1058** of file **stm324x9i_eval_lcd.c**.

Referenced by **BSP_LCD_InitEx()**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL NOR Private Functions

STM324x9I EVAL NOR

Functions

static void	**NOR_MspInit** (void) Initializes the NOR MSP.

Function Documentation

static void NOR_MspInit (void) [static]

Initializes the NOR MSP.

Definition at line **338** of file **stm324x9i_eval_nor.c**.

Referenced by **BSP_NOR_Init()**.

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL NOR Private Variables

STM324x9I EVAL NOR

Variables

static NOR_HandleTypeDef	**norHandle**
static FMC_NORSRAM_TimingTypeDef	**Timing**

Variable Documentation

NOR_HandleTypeDef norHandle [static]

Definition at line 122 of file stm324x9i_eval_nor.c.

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

FMC_NORSRAM_TimingTypeDef Timing [static]

Definition at line 123 of file stm324x9i_eval_nor.c.

Referenced by BSP_NOR_Init().

STM324x9I_EVAL BSP User Manual

Defines

STM324x9I EVAL LCD Private Defines

STM324x9I 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 **108** of file **stm324x9i_eval_lcd.c**.

Referenced by **BSP_LCD_FillPolygon()**.

#define POLY_Y (Z) ((int32_t)((Points + Z)->Y))

Definition at line **109** of file **stm324x9i_eval_lcd.c**.

Referenced by **BSP_LCD_FillPolygon()**.

STM324x9I_EVAL BSP User Manual

Defines

STM324x9I EVAL SD Exported Types

STM324x9I 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 **stm324x9i_eval_sd.h**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL SD
Private FunctionPrototypes

STM324x9I EVAL SD

Functions

static void	**SD_MspInit** (void) Initializes the SD MSP.

Function Documentation

static void SD_MspInit (void) [static]

Initializes the SD MSP.

Definition at line **372** of file **stm324x9i_eval_sd.c**.

References **__DMAx_TxRx_CLK_ENABLE**, **SD_DMAx_Rx_CHANNEL**, **SD_DMAx_Rx_IRQn**, **SD_DMAx_Rx_STREAM**, **SD_DMAx_Tx_CHANNEL**, **SD_DMAx_Tx_IRQn**, **SD_DMAx_Tx_STREAM**, and **uSdHandle**.

Referenced by **BSP_SD_Init()**.

STM324x9I_EVAL BSP User Manual

Defines

STM324x9I EVAL SDRAM Exported Types

STM324x9I EVAL SDRAM

Defines

#define	**SDRAM_OK** 0x00 SDRAM status structure definition.
#define	**SDRAM_ERROR** 0x01

Define Documentation

#define SDRAM_ERROR 0x01

Definition at line 70 of file stm324x9i_eval_sdram.h.

Referenced by BSP_SDRAM_Init(), BSP_SDRAM_ReadData(), BSP_SDRAM_ReadData_DMA(), BSP_SDRAM_Sendcmd(), BSP_SDRAM_WriteData(), and BSP_SDRAM_WriteData_DMA().

#define SDRAM_OK 0x00

SDRAM status structure definition.

Definition at line 69 of file stm324x9i_eval_sdram.h.

Referenced by BSP_SDRAM_Init(), BSP_SDRAM_ReadData(), BSP_SDRAM_ReadData_DMA(), BSP_SDRAM_Sendcmd(), BSP_SDRAM_WriteData(), and BSP_SDRAM_WriteData_DMA().

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL
SDRAM Private Function Prototypes

STM324x9I EVAL SDRAM

Functions

static void	**SDRAM_MspInit** (void) Initializes SDRAM MSP.

Function Documentation

static void SDRAM_MspInit (void) [static]

Initializes SDRAM MSP.

Definition at line **348** of file **stm324x9i_eval_sdram.c**.

References **__DMAx_CLK_ENABLE**, **SDRAM_DMAx_CHANNEL**, **SDRAM_DMAx_IRQn**, **SDRAM_DMAx_STREAM**, and **sdramHandle**.

Referenced by **BSP_SDRAM_Init()**.

STM324x9I_EVAL BSP User Manual

Functions

STM324x9I EVAL
SRAM Private Function Prototypes

STM324x9I EVAL SRAM

Functions

static void	**SRAM_MspInit** (void) Initializes SRAM MSP.

Function Documentation

static void SRAM_MspInit (void) [static]

Initializes SRAM MSP.

Definition at line **261** of file **stm324x9i_eval_sram.c**.

References **__SRAM_DMAx_CLK_ENABLE**, **SRAM_DMAx_CHANNEL**, **SRAM_DMAx_IRQn**, **SRAM_DMAx_STREAM**, and **sramHandle**.

Referenced by **BSP_SRAM_Init()**.

STM324x9I_EVAL BSP User Manual

Variables

STM324x9I EVAL
SRAM Private Variables

STM324x9I EVAL SRAM

Variables

static SRAM_HandleTypeDef	**sramHandle**
static FMC_NORSRAM_TimingTypeDef	**Timing**

Variable Documentation

SRAM_HandleTypeDef sramHandle [static]

Definition at line 111 of file stm324x9i_eval_sram.c.

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

FMC_NORSRAM_TimingTypeDef Timing [static]

Definition at line 112 of file stm324x9i_eval_sram.c.

Referenced by BSP_SRAM_Init().

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL TS Exported Constants

STM324x9I EVAL TS

Defines

#define **TS_SWAP_NONE** 0x00
#define **TS_SWAP_X** 0x01
#define **TS_SWAP_Y** 0x02
#define **TS_SWAP_XY** 0x04
#define **TS_INT_PIN** 0x0010

Enumerations

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

Define Documentation

#define TS_INT_PIN 0x0010

Definition at line 96 of file stm324x9i_eval_ts.h.

Referenced by BSP_TS_ITConfig().

#define TS_SWAP_NONE 0x00

Definition at line 83 of file stm324x9i_eval_ts.h.

Referenced by BSP_TS_Init().

#define TS_SWAP_X 0x01

Definition at line 84 of file stm324x9i_eval_ts.h.

Referenced by BSP_TS_GetState().

#define TS_SWAP_XY 0x04

Definition at line 86 of file stm324x9i_eval_ts.h.

Referenced by BSP_TS_GetState().

#define TS_SWAP_Y 0x02

Definition at line 85 of file stm324x9i_eval_ts.h.

Referenced by BSP_TS_GetState(), and BSP_TS_Init().

Enumeration Type Documentation

enum TS_StatusTypeDef

Enumerator:

TS_OK

TS_ERROR

TS_TIMEOUT

Definition at line **88** of file **stm324x9i_eval_ts.h**.

STM324x9I_EVAL BSP User Manual

Variables

STM324x9I EVAL SD Private Variables

STM324x9I EVAL SD

Variables

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

Variable Documentation

SD_CardInfo uSdCardInfo [static]

Definition at line 123 of file stm324x9i_eval_sd.c.

Referenced by BSP_SD_Init().

SD_HandleTypeDef uSdHandle [static]

Definition at line 122 of file stm324x9i_eval_sd.c.

Referenced by BSP_SD_DMA_Rx_IRQHandler(), BSP_SD_DMA_Tx_IRQHandler(), BSP_SD_Erase(), BSP_SD_GetCardInfo(), BSP_SD_GetStatus(), BSP_SD_Init(), BSP_SD_IRQHandler(), BSP_SD_ReadBlocks(), BSP_SD_ReadBlocks_DMA(), BSP_SD_WriteBlocks(), BSP_SD_WriteBlocks_DMA(), and SD_MspInit().

STM324x9I_EVAL BSP User Manual

Drivers Directory Reference

Directories

directory	**BSP**

STM324x9I_EVAL BSP User Manual

BSP Directory Reference

Directories

directory	**STM324x9I_EVAL**

STM324x9I_EVAL BSP User Manual

STM324x9I_EVAL Directory Reference

Files

file **stm324x9i_eval.c** [code]

This file provides a set of firmware functions to manage LEDs, push-buttons and COM ports available on STM324x9I-EVAL evaluation board(MB1045) RevB from STMicroelectronics.

file **stm324x9i_eval.h** [code]

This file contains definitions for STM324x9I_EVAL's LEDs, push-buttons and COM ports hardware resources.

file **stm324x9i_eval_audio.c** [code]

This file provides the Audio driver for the STM324x9I-EVAL evaluation board.

file **stm324x9i_eval_audio.h** [code]

This file contains the common defines and functions prototypes for the **stm324x9i_eval_audio.c** driver.

file **stm324x9i_eval_camera.c** [code]

This file includes the driver for Camera modules mounted on STM324x9I-EVAL evaluation board.

file **stm324x9i_eval_camera.h** [code]

This file contains the common defines and functions prototypes for the **stm324x9i_eval_camera.c** driver.

file **stm324x9i_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 **stm324x9i_eval_eeprom.h** [code]

This file contains all the functions prototypes for the **stm324x9i_eval_eeprom.c** firmware driver.

file **stm324x9i_eval_io.c** [code]

This file provides a set of functions needed to manage the IO pins on STM324x9I-EVAL evaluation board.

file **stm324x9i_eval_io.h** [code]

This file contains the common defines and functions prototypes for the **stm324x9i_eval_io.c** driver.

file **stm324x9i_eval_lcd.c** [code]

This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM324x9I-EVAL evaluation board.

file **stm324x9i_eval_lcd.h** [code]

This file contains the common defines and functions prototypes for the **stm324x9i_eval_lcd.c** driver.

file **stm324x9i_eval_nor.c** [code]

This file includes a standard driver for the M29W256GL70ZA6E NOR flash memory device mounted on STM324x9I-EVAL evaluation board.

file **stm324x9i_eval_nor.h** [code]

This file contains the common defines and functions prototypes for the **stm324x9i_eval_nor.c** driver.

file **stm324x9i_eval_sd.c** [code]

This file includes the uSD card driver mounted on STM324x9I-EVAL evaluation board.

file **stm324x9i_eval_sd.h** [code]

This file contains the common defines and functions prototypes for the **stm324x9i_eval_sd.c** driver.

file **stm324x9i_eval_sdram.c** [code]

This file includes the SDRAM driver for the MT48LC4M32B2B5-7 memory device mounted on STM324x9I-EVAL evaluation board.

file **stm324x9i_eval_sdram.h** [code]

This file contains the common defines and functions prototypes for the **stm324x9i_eval_sdram.c** driver.

file **stm324x9i_eval_sram.c** [code]

This file includes the SRAM driver for the IS61WV102416BLL-10M memory device mounted on STM324x9I-EVAL evaluation board.

file **stm324x9i_eval_sram.h** [code]

This file contains the common defines and functions prototypes for the **stm324x9i_eval_sram.c** driver.

file **stm324x9i_eval_ts.c** [code]

This file provides a set of functions needed to manage the Touch Screen on STM324x9I-EVAL evaluation board.

file **stm324x9i_eval_ts.h** [code]

This file contains the common defines and functions prototypes for the **stm324x9i_eval_ts.c** driver.

STM324x9I_EVAL BSP User Manual

stm324x9i_eval.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval.h
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file contains definitions for STM324x9I_EVAL's LEDs,
00008   *          push-buttons and COM ports hardware resources.
00009   ******************************************************************************
00010   * @attention
00011   *
00012   * <h2><center>&copy; COPYRIGHT(c) 2015 STMicroelectronics</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without modification,
00015   * are permitted provided that the following conditions are met:
```

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

```
00035   *
00036   ******************************************
*********************************
00037   */
00038
00039 /* Define to prevent recursive inclusion ---
-----------------------------------*/
00040 #ifndef __STM324X9I_EVAL_H
00041 #define __STM324X9I_EVAL_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 STM324x9I_EVAL
00055   * @{
00056   */
00057
00058 /** @addtogroup STM324x9I_EVAL_LOW_LEVEL
00059   * @{
00060   */
00061
00062 /** @defgroup STM324x9I_EVAL_LOW_LEVEL_Expor
ted_Types STM324x9I EVAL LOW LEVEL Exported Types
00063   * @{
00064   */
00065 typedef enum
00066 {
00067   LED1 = 0,
```

```
00068   LED2 = 1,
00069   LED3 = 2,
00070   LED4 = 3
00071 }Led_TypeDef;
00072
00073 typedef enum
00074 {
00075   BUTTON_WAKEUP = 0,
00076   BUTTON_TAMPER = 1,
00077   BUTTON_KEY = 2
00078 }Button_TypeDef;
00079
00080 typedef enum
00081 {
00082   BUTTON_MODE_GPIO = 0,
00083   BUTTON_MODE_EXTI = 1
00084 }ButtonMode_TypeDef;
00085
00086 typedef enum
00087 {
00088   JOY_MODE_GPIO = 0,
00089   JOY_MODE_EXTI = 1
00090 }JOYMode_TypeDef;
00091
00092 typedef enum
00093 {
00094   JOY_NONE  = 0,
00095   JOY_SEL   = 1,
00096   JOY_DOWN  = 2,
00097   JOY_LEFT  = 3,
00098   JOY_RIGHT = 4,
00099   JOY_UP    = 5
00100 }JOYState_TypeDef;
00101
00102 typedef enum
00103 {
00104   COM1 = 0,
```

```
00105   COM2 = 1
00106 }COM_TypeDef;
00107 /**
00108   * @}
00109   */
00110
00111 /** @defgroup STM324x9I_EVAL_LOW_LEVEL_Exported_Constants STM324x9I EVAL LOW LEVEL Exported Constants
00112   * @{
00113   */
00114
00115 /**
00116   * @brief  Define for STM324x9I_EVAL board
00117   */
00118 #if !defined (USE_STM324x9I_EVAL)
00119  #define USE_STM324x9I_EVAL
00120 #endif
00121
00122 /** @defgroup STM324x9I_EVAL_LOW_LEVEL_LED STM324x9I EVAL LOW LEVEL LED
00123   * @{
00124   */
00125 #define LEDn                                    4
00126
00127 #define LED1_PIN                                GPIO_PIN_6
00128 #define LED1_GPIO_PORT                          GPIOG
00129 #define LED1_GPIO_CLK_ENABLE()                  __GPIOG_CLK_ENABLE()
00130 #define LED1_GPIO_CLK_DISABLE()                 __GPIOG_CLK_DISABLE()
00131
00132
00133 #define LED2_PIN                                GPI
```

```
O_PIN_7
00134 #define LED2_GPIO_PORT                          GPI
OG
00135 #define LED2_GPIO_CLK_ENABLE()                  __G
PIOG_CLK_ENABLE()
00136 #define LED2_GPIO_CLK_DISABLE()                 __G
PIOG_CLK_DISABLE()
00137
00138 #define LED3_PIN                                GPI
O_PIN_10
00139 #define LED3_GPIO_PORT                          GPI
OG
00140 #define LED3_GPIO_CLK_ENABLE()                  __G
PIOG_CLK_ENABLE()
00141 #define LED3_GPIO_CLK_DISABLE()                 __G
PIOG_CLK_DISABLE()
00142
00143 #define LED4_PIN                                GPI
O_PIN_12
00144 #define LED4_GPIO_PORT                          GPI
OG
00145 #define LED4_GPIO_CLK_ENABLE()                  __G
PIOG_CLK_ENABLE()
00146 #define LED4_GPIO_CLK_DISABLE()                 __G
PIOG_CLK_DISABLE()
00147
00148 #define LEDx_GPIO_CLK_ENABLE(__INDEX__)  do{
if((__INDEX__) == 0) LED1_GPIO_CLK_ENABLE(); else \

00149
if((__INDEX__) == 1) LED2_GPIO_CLK_ENABLE(); else \

00150
if((__INDEX__) == 2) LED3_GPIO_CLK_ENABLE(); else \

00151
if((__INDEX__) == 3) LED4_GPIO_CLK_ENABLE(); \
```

```
00152
}while(0)
00153
00154 #define LEDx_GPIO_CLK_DISABLE(__INDEX__)  do
{if((__INDEX__) == 0) LED1_GPIO_CLK_DISABLE(); els
e \
00155
 if((__INDEX__) == 1) LED2_GPIO_CLK_DISABLE(); els
e \
00156
 if((__INDEX__) == 2) LED3_GPIO_CLK_DISABLE(); els
e \
00157
 if((__INDEX__) == 3) LED4_GPIO_CLK_DISABLE(); \
00158
 }while(0)
00159
00160 /**
00161   * @}
00162   */
00163
00164 /** @defgroup STM324x9I_EVAL_LOW_LEVEL_BUTTO
N STM324x9I EVAL LOW LEVEL BUTTON
00165   * @{
00166   */
00167 /* Joystick pins are connected to IO Expande
r (accessible through I2C1 interface) */
00168 #define BUTTONn
3
00169
00170 /**
00171   * @brief Wakeup push-button
00172   */
00173 #define WAKEUP_BUTTON_PIN
GPIO_PIN_0
00174 #define WAKEUP_BUTTON_GPIO_PORT
GPIOA
```

```
00175 #define WAKEUP_BUTTON_GPIO_CLK_ENABLE()
__GPIOA_CLK_ENABLE()
00176 #define WAKEUP_BUTTON_GPIO_CLK_DISABLE()
__GPIOA_CLK_DISABLE()
00177 #define WAKEUP_BUTTON_EXTI_IRQn
EXTI0_IRQn
00178
00179 /**
00180   * @brief Tamper push-button
00181   */
00182 #define TAMPER_BUTTON_PIN
 GPIO_PIN_13
00183 #define TAMPER_BUTTON_GPIO_PORT
 GPIOC
00184 #define TAMPER_BUTTON_GPIO_CLK_ENABLE()
 __GPIOC_CLK_ENABLE()
00185 #define TAMPER_BUTTON_GPIO_CLK_DISABLE()
 __GPIOC_CLK_DISABLE()
00186 #define TAMPER_BUTTON_EXTI_IRQn
 EXTI15_10_IRQn
00187
00188 /**
00189   * @brief Key push-button
00190   */
00191 #define KEY_BUTTON_PIN
 GPIO_PIN_13
00192 #define KEY_BUTTON_GPIO_PORT
 GPIOC
00193 #define KEY_BUTTON_GPIO_CLK_ENABLE()
 __GPIOC_CLK_ENABLE()
00194 #define KEY_BUTTON_GPIO_CLK_DISABLE()
 __GPIOC_CLK_DISABLE()
00195 #define KEY_BUTTON_EXTI_IRQn
 EXTI15_10_IRQn
00196
00197 #define BUTTONx_GPIO_CLK_ENABLE(__INDEX__)
do{if((__INDEX__) == 0) WAKEUP_BUTTON_GPIO_CLK_ENA
```

```
BLE(); else \
00198
   if((__INDEX__) == 1) TAMPER_BUTTON_GPIO_CLK_ENA
BLE(); else \
00199
   if ((__INDEX__) == 2) KEY_BUTTON_GPIO_CLK_ENABL
E(); \
00200
   }while(0)
00201 #define BUTTONx_GPIO_CLK_DISABLE(__INDEX__)
do{if((__INDEX__) == 0) WAKEUP_BUTTON_GPIO_CLK_DIS
ABLE(); else \
00202
   if((__INDEX__) == 1) TAMPER_BUTTON_GPIO_CLK_DIS
ABLE(); else \
00203
   if ((__INDEX__) == 2) KEY_BUTTON_GPIO_CLK_DISAB
LE(); \
00204
   }while(0)
00205 /**
00206   * @}
00207   */
00208
00209 /** @defgroup STM324x9I_EVAL_LOW_LEVEL_COM S
TM324x9I EVAL LOW LEVEL COM
00210   * @{
00211   */
00212 #define COMn                                1
00213
00214 /**
00215  * @brief Definition for COM port1, connecte
d to USART1
00216  */
00217 #define EVAL_COM1                             U
SART1
00218 #define EVAL_COM1_CLK_ENABLE()                _
```

```
_USART1_CLK_ENABLE()
00219 #define EVAL_COM1_CLK_DISABLE()             _
_USART1_CLK_DISABLE()
00220
00221 #define EVAL_COM1_TX_PIN                    G
PIO_PIN_9
00222 #define EVAL_COM1_TX_GPIO_PORT              G
PIOA
00223 #define EVAL_COM1_TX_GPIO_CLK_ENABLE()      _
_GPIOA_CLK_ENABLE()
00224 #define EVAL_COM1_TX_GPIO_CLK_DISABLE()     _
_GPIOA_CLK_DISABLE()
00225 #define EVAL_COM1_TX_AF                     G
PIO_AF7_USART1
00226
00227 #define EVAL_COM1_RX_PIN                    G
PIO_PIN_10
00228 #define EVAL_COM1_RX_GPIO_PORT              G
PIOA
00229 #define EVAL_COM1_RX_GPIO_CLK_ENABLE()      _
_GPIOA_CLK_ENABLE()
00230 #define EVAL_COM1_RX_GPIO_CLK_DISABLE()     _
_GPIOA_CLK_DISABLE()
00231 #define EVAL_COM1_RX_AF                     G
PIO_AF7_USART1
00232
00233 #define EVAL_COM1_IRQn                      U
SART1_IRQn
00234
00235 #define EVAL_COMx_CLK_ENABLE(__INDEX__)
         do{if((__INDEX__) == 0) EVAL_COM1_CLK_ENA
BLE(); \
00236
           }while(0)
00237 #define EVAL_COMx_CLK_DISABLE(__INDEX__)
         do{if((__INDEX__) == 0) EVAL_COM1_CLK_DIS
ABLE(); \
```

```
00238
          }while(0)
00239
00240 #define EVAL_COMx_TX_GPIO_CLK_ENABLE(__INDEX
__)      do{if((__INDEX__) == 0) EVAL_COM1_TX_GPIO
_CLK_ENABLE(); \
00241
         }while(0)
00242 #define EVAL_COMx_TX_GPIO_CLK_DISABLE(__INDE
X__)     do{if((__INDEX__) == 0) EVAL_COM1_TX_GPIO
_CLK_DISABLE(); \
00243
         }while(0)
00244
00245 #define EVAL_COMx_RX_GPIO_CLK_ENABLE(__INDEX
__)      do{if((__INDEX__) == 0) EVAL_COM1_RX_GPIO
_CLK_ENABLE(); \
00246
          }while(0)
00247 #define EVAL_COMx_RX_GPIO_CLK_DISABLE(__INDE
X__)     do{if((__INDEX__) == 0) EVAL_COM1_RX_GPIO
_CLK_DISABLE(); \
00248
          }while(0)
00249
00250 /**
00251   * @brief Joystick Pins definition
00252   */
00253 #define JOY_SEL_PIN                     IO_PI
N_14
00254 #define JOY_DOWN_PIN                    IO_PI
N_13
00255 #define JOY_LEFT_PIN                    IO_PI
N_12
00256 #define JOY_RIGHT_PIN                   IO_PI
N_11
00257 #define JOY_UP_PIN                      IO_PI
```

```
N_10
00258 #define JOY_NONE_PIN                    JOY_A
LL_PINS
00259 #define JOY_ALL_PINS                    (IO_P
IN_10 | IO_PIN_11 | IO_PIN_12 | IO_PIN_13 | IO_PIN
_14)
00260
00261 /**
00262   * @brief Eval Pins definition
00263   */
00264 #define XSDN_PIN                        IO_PI
N_0
00265 #define MII_INT_PIN                     IO_PI
N_1
00266 #define RSTI_PIN                        IO_PI
N_2
00267 #define CAM_PLUG_PIN                    IO_PI
N_3
00268 #define LCD_INT_PIN                     IO_PI
N_4
00269 #define AUDIO_INT_PIN                   IO_PI
N_5
00270 #define OTG_FS1_OVER_CURRENT_PIN        IO_PI
N_6
00271 #define OTG_FS1_POWER_SWITCH_PIN        IO_PI
N_7
00272 #define OTG_FS2_OVER_CURRENT_PIN        IO_PI
N_8
00273 #define OTG_FS2_POWER_SWITCH_PIN        IO_PI
N_9
00274 #define SD_DETECT_PIN                   IO_PI
N_15
00275
00276 /* Exported constant IO ------------------
-----------------------------------*/
00277 #define IO_I2C_ADDRESS                    0x8
4
```

```
00278 #define TS_I2C_ADDRESS                  0x82

00279 #define TS3510_I2C_ADDRESS              0x80

00280 #define EXC7200_I2C_ADDRESS             0x0
8

00281 #define CAMERA_I2C_ADDRESS              0x60

00282 #define AUDIO_I2C_ADDRESS               0x34

00283 #define EEPROM_I2C_ADDRESS_A01          0xA0

00284 #define EEPROM_I2C_ADDRESS_A02          0xA
6
00285 /* I2C clock speed configuration (in Hz)
00286    WARNING:
00287    Make sure that this define is not already
 declared in other files (ie.
00288    stm324x9I_eval.h file). It can be used in
 parallel by other modules. */
00289 #ifndef BSP_I2C_SPEED
00290  #define BSP_I2C_SPEED
  100000
00291 #endif /* BSP_I2C_SPEED */
00292
00293 /* User can use this section to tailor I2Cx/
I2Cx instance used and associated
00294    resources */
00295 /* Definition for I2Cx clock resources */
00296 #define EVAL_I2Cx
  I2C1
00297 #define EVAL_I2Cx_CLK_ENABLE()
  __I2C1_CLK_ENABLE()
00298 #define EVAL_DMAx_CLK_ENABLE()
  __DMA1_CLK_ENABLE()
00299 #define EVAL_I2Cx_SCL_SDA_GPIO_CLK_ENABLE()
```

```
  __GPIOB_CLK_ENABLE()
00300
00301 #define EVAL_I2Cx_FORCE_RESET()
  __I2C1_FORCE_RESET()
00302 #define EVAL_I2Cx_RELEASE_RESET()
  __I2C1_RELEASE_RESET()
00303
00304 /* Definition for I2Cx Pins */
00305 #define EVAL_I2Cx_SCL_PIN
  GPIO_PIN_6
00306 #define EVAL_I2Cx_SCL_SDA_GPIO_PORT
  GPIOB
00307 #define EVAL_I2Cx_SCL_SDA_AF
  GPIO_AF4_I2C1
00308 #define EVAL_I2Cx_SDA_PIN
  GPIO_PIN_9
00309
00310 /* I2C interrupt requests */
00311 #define EVAL_I2Cx_EV_IRQn
  I2C1_EV_IRQn
00312 #define EVAL_I2Cx_ER_IRQn
  I2C1_ER_IRQn
00313
00314 /**
00315   * @}
00316   */
00317
00318 /**
00319   * @}
00320   */
00321
00322 /** @defgroup STM324x9I_EVAL_LOW_LEVEL_Expor
ted_Macros STM324x9I EVAL LOW LEVEL Exported Macros

00323   * @{
00324   */
00325 /**
```

```
00326   * @}
00327   */
00328
00329 /** @defgroup STM324x9I_EVAL_LOW_LEVEL_Exported_Functions STM324x9I EVAL LOW LEVEL Exported Functions
00330   * @{
00331   */
00332 uint32_t         BSP_GetVersion(void);
00333 void             BSP_LED_Init(Led_TypeDef Led);
00334 void             BSP_LED_On(Led_TypeDef Led);
00335 void             BSP_LED_Off(Led_TypeDef Led);
00336 void             BSP_LED_Toggle(Led_TypeDef Led);
00337 void             BSP_PB_Init(Button_TypeDef Button, ButtonMode_TypeDef Button_Mode);
00338 uint32_t         BSP_PB_GetState(Button_TypeDef Button);
00339 void             BSP_COM_Init(COM_TypeDef COM, UART_HandleTypeDef *husart);
00340 uint8_t          BSP_JOY_Init(JOYMode_TypeDef Joy_Mode);
00341 JOYState_TypeDef BSP_JOY_GetState(void);
00342 uint8_t          BSP_TS3510_IsDetected(void);
00343
00344 /**
00345   * @}
00346   */
00347
00348 /**
00349   * @}
00350   */
00351
```

```
00352 /**
00353   * @}
00354   */
00355
00356 /**
00357   * @}
00358   */
00359
00360 #ifdef __cplusplus
00361 }
00362 #endif
00363
00364 #endif /* __STM324X9I_EVAL_H */
00365
00366 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM324x9I_EVAL BSP User Manual

stm324x9i_eval.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval.c
00004   * @author  MCD Application Team
00005   * @version V2.2.2
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 STM324x9I-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 stm324x9i_eval_i
o to manage the joystick
00044
00045 ------------------------------------------
-----------------------------------*/
00046
00047 /* Includes --------------------------------
-----------------------------------*/
00048 #include "stm324x9i_eval.h"
00049 #include "stm324x9i_eval_io.h"
00050
00051 /** @defgroup BSP BSP
00052   * @{
00053   */
00054
00055 /** @defgroup STM324x9I_EVAL STM324x9I EVAL
00056   * @{
00057   */
00058
00059 /** @defgroup STM324x9I_EVAL_LOW_LEVEL STM32
4x9I EVAL LOW LEVEL
00060   * @{
00061   */
00062
00063 /** @defgroup STM324x9I_EVAL_LOW_LEVEL_Priva
te_TypesDefinitions STM324x9I EVAL LOW LEVEL Priva
```

```
te TypesDefinitions
00064   * @{
00065   */
00066 /**
00067   * @}
00068   */
00069
00070 /** @defgroup STM324x9I_EVAL_LOW_LEVEL_Priva
te_Defines STM324x9I EVAL LOW LEVEL Private Defines

00071   * @{
00072   */
00073 /**
00074  * @brief STM324x9I EVAL BSP Driver version
number V2.2.2
00075    */
00076 #define __STM324x9I_EVAL_BSP_VERSION_MAIN
(0x02) /*!< [31:24] main version */
00077 #define __STM324x9I_EVAL_BSP_VERSION_SUB1
(0x02) /*!< [23:16] sub1 version */
00078 #define __STM324x9I_EVAL_BSP_VERSION_SUB2
(0x02) /*!< [15:8]  sub2 version */
00079 #define __STM324x9I_EVAL_BSP_VERSION_RC
(0x00) /*!< [7:0]  release candidate */
00080 #define __STM324x9I_EVAL_BSP_VERSION
 ((__STM324x9I_EVAL_BSP_VERSION_MAIN << 24)\
00081
 |(__STM324x9I_EVAL_BSP_VERSION_SUB1 << 16)\
00082
 |(__STM324x9I_EVAL_BSP_VERSION_SUB2 << 8 )\
00083
 |(__STM324x9I_EVAL_BSP_VERSION_RC))
00084 /**
00085   * @}
00086   */
00087
00088 /** @defgroup STM324x9I_EVAL_LOW_LEVEL_Priva
```

```
te_Macros STM324x9I EVAL LOW_LEVEL_Private_Macros
00089   * @{
00090   */
00091 /**
00092   * @}
00093   */
00094
00095 /** @defgroup STM324x9I_EVAL_LOW_LEVEL_Priva
te_Variables STM324x9I EVAL LOW LEVEL Private Vari
ables
00096   * @{
00097   */
00098 GPIO_TypeDef* GPIO_PORT[LEDn] = {LED1_GPIO_P
ORT,
00099                                  LED2_GPIO_P
ORT,
00100                                  LED3_GPIO_P
ORT,
00101                                  LED4_GPIO_P
ORT};
00102
00103 const uint16_t GPIO_PIN[LEDn] = {LED1_PIN,
00104                                  LED2_PIN,
00105                                  LED3_PIN,
00106                                  LED4_PIN};
00107
00108 GPIO_TypeDef* BUTTON_PORT[BUTTONn] = {WAKEUP
_BUTTON_GPIO_PORT,
00109                                       TAMPER
_BUTTON_GPIO_PORT,
00110                                       KEY_BU
TTON_GPIO_PORT};
00111
00112 const uint16_t BUTTON_PIN[BUTTONn] = {WAKEUP
_BUTTON_PIN,
00113                                       TAMPER
_BUTTON_PIN,
```

```
00114                                                   KEY_BU
TTON_PIN};
00115
00116 const uint16_t BUTTON_IRQn[BUTTONn] = {WAKEU
P_BUTTON_EXTI_IRQn,
00117                                                    TAMPE
R_BUTTON_EXTI_IRQn,
00118                                                    KEY_B
UTTON_EXTI_IRQn};
00119
00120 USART_TypeDef* COM_USART[COMn] = {EVAL_COM1}
;
00121
00122 GPIO_TypeDef* COM_TX_PORT[COMn] = {EVAL_COM1
_TX_GPIO_PORT};
00123
00124 GPIO_TypeDef* COM_RX_PORT[COMn] = {EVAL_COM1
_RX_GPIO_PORT};
00125
00126 const uint16_t COM_TX_PIN[COMn] = {EVAL_COM1
_TX_PIN};
00127
00128 const uint16_t COM_RX_PIN[COMn] = {EVAL_COM1
_RX_PIN};
00129
00130 const uint16_t COM_TX_AF[COMn] = {EVAL_COM1_
TX_AF};
00131
00132 const uint16_t COM_RX_AF[COMn] = {EVAL_COM1_
RX_AF};
00133
00134 static I2C_HandleTypeDef heval_I2c;
00135
00136 /**
00137   * @}
00138   */
00139
```

```
00140 /** @defgroup STM324x9I_EVAL_LOW_LEVEL_Private_FunctionPrototypes STM324x9I EVAL LOW LEVEL Private FunctionPrototypes
00141   * @{
00142   */
00143 static void     I2Cx_MspInit(void);
00144 static void     I2Cx_Init(void);
00145 static void     I2Cx_ITConfig(void);
00146 static void     I2Cx_Write(uint8_t Addr, uint8_t Reg, uint8_t Value);
00147 static uint8_t  I2Cx_Read(uint8_t Addr, uint8_t Reg);
00148 static HAL_StatusTypeDef I2Cx_ReadMultiple(uint8_t Addr, uint16_t Reg, uint16_t MemAddSize, uint8_t *Buffer, uint16_t Length);
00149 static HAL_StatusTypeDef I2Cx_WriteMultiple(uint8_t Addr, uint16_t Reg, uint16_t MemAddSize, uint8_t *Buffer, uint16_t Length);
00150 static HAL_StatusTypeDef I2Cx_IsDeviceReady(uint16_t DevAddress, uint32_t Trials);
00151 static void     I2Cx_Error(uint8_t Addr);
00152
00153 /* IOExpander IO functions */
00154 void            IOE_Init(void);
00155 void            IOE_ITConfig(void);
00156 void            IOE_Delay(uint32_t Delay);
00157 void            IOE_Write(uint8_t Addr, uint8_t Reg, uint8_t Value);
00158 uint8_t         IOE_Read(uint8_t Addr, uint8_t Reg);
00159 uint16_t        IOE_ReadMultiple(uint8_t Addr, uint8_t Reg, uint8_t *Buffer, uint16_t Length);
00160 void            IOE_WriteMultiple(uint8_t Addr, uint8_t Reg, uint8_t *Buffer, uint16_t Length);
00161
00162 /* AUDIO IO functions */
```

```
00163 void             AUDIO_IO_Init(void);
00164 void             AUDIO_IO_DeInit(void);
00165 void             AUDIO_IO_Write(uint8_t Addr,
 uint16_t Reg, uint16_t Value);
00166 uint16_t         AUDIO_IO_Read(uint8_t Addr,
uint16_t Reg);
00167 void             AUDIO_IO_Delay(uint32_t Dela
y);
00168
00169 /* CAMERA IO functions */
00170 void             CAMERA_IO_Init(void);
00171 void             CAMERA_Delay(uint32_t Delay)
;
00172 void             CAMERA_IO_Write(uint8_t Addr
, uint8_t Reg, uint8_t Value);
00173 uint8_t          CAMERA_IO_Read(uint8_t Addr,
 uint8_t Reg);
00174
00175 /* I2C EEPROM IO function */
00176 void                EEPROM_IO_Init(void);
00177 HAL_StatusTypeDef   EEPROM_IO_WriteData(uint
16_t DevAddress, uint16_t MemAddress, uint8_t* pBu
ffer, uint32_t BufferSize);
00178 HAL_StatusTypeDef   EEPROM_IO_ReadData(uint1
6_t DevAddress, uint16_t MemAddress, uint8_t* pBuf
fer, uint32_t BufferSize);
00179 HAL_StatusTypeDef   EEPROM_IO_IsDeviceReady(
uint16_t DevAddress, uint32_t Trials);
00180 /**
00181   * @}
00182   */
00183
00184 /** @defgroup STM324x9I_EVAL_LOW_LEVEL_Priva
te_Functions STM324x9I EVAL LOW LEVEL Private Func
tions
00185   * @{
00186   */
```

```
00187
00188   /**
00189   * @brief  This method returns the STM324x9
I EVAL BSP Driver revision
00190   * @retval version: 0xXYZR (8bits for each
decimal, R for RC)
00191   */
00192 uint32_t BSP_GetVersion(void)
00193 {
00194   return __STM324x9I_EVAL_BSP_VERSION;
00195 }
00196
00197 /**
00198   * @brief  Configures LED GPIO.
00199   * @param  Led: LED to be configured.
00200   *          This parameter can be one of th
e following values:
00201   *            @arg  LED1
00202   *            @arg  LED2
00203   *            @arg  LED3
00204   *            @arg  LED4
00205   */
00206 void BSP_LED_Init(Led_TypeDef Led)
00207 {
00208   GPIO_InitTypeDef  GPIO_InitStruct;
00209
00210   /* Enable the GPIO_LED clock */
00211   LEDx_GPIO_CLK_ENABLE(Led);
00212
00213   /* Configure the GPIO_LED pin */
00214   GPIO_InitStruct.Pin = GPIO_PIN[Led];
00215   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP
;
00216   GPIO_InitStruct.Pull = GPIO_PULLUP;
00217   GPIO_InitStruct.Speed = GPIO_SPEED_FAST;
00218
00219   HAL_GPIO_Init(GPIO_PORT[Led], &GPIO_InitSt
```

```
ruct);
00220
00221   HAL_GPIO_WritePin(GPIO_PORT[Led], GPIO_PIN
[Led], GPIO_PIN_SET);
00222 }
00223
00224 /**
00225   * @brief  Turns selected LED On.
00226   * @param  Led: LED to be set on
00227   *          This parameter can be one of th
e following values:
00228   *            @arg  LED1
00229   *            @arg  LED2
00230   *            @arg  LED3
00231   *            @arg  LED4
00232   */
00233 void BSP_LED_On(Led_TypeDef Led)
00234 {
00235   HAL_GPIO_WritePin(GPIO_PORT[Led], GPIO_PIN
[Led], GPIO_PIN_RESET);
00236 }
00237
00238 /**
00239   * @brief  Turns selected LED Off.
00240   * @param  Led: LED to be set off
00241   *          This parameter can be one of th
e following values:
00242   *            @arg  LED1
00243   *            @arg  LED2
00244   *            @arg  LED3
00245   *            @arg  LED4
00246   */
00247 void BSP_LED_Off(Led_TypeDef Led)
00248 {
00249   HAL_GPIO_WritePin(GPIO_PORT[Led], GPIO_PIN
[Led], GPIO_PIN_SET);
00250 }
```

```
00251
00252 /**
00253   * @brief  Toggles the selected LED.
00254   * @param  Led: LED to be toggled
00255   *          This parameter can be one of th
e following values:
00256   *            @arg  LED1
00257   *            @arg  LED2
00258   *            @arg  LED3
00259   *            @arg  LED4
00260   */
00261 void BSP_LED_Toggle(Led_TypeDef Led)
00262 {
00263   HAL_GPIO_TogglePin(GPIO_PORT[Led], GPIO_PIN
[Led]);
00264 }
00265
00266 /**
00267   * @brief  Configures button GPIO and EXTI
Line.
00268   * @param  Button: Button to be configured
00269   *          This parameter can be one of th
e following values:
00270   *            @arg  BUTTON_WAKEUP: Wakeup P
ush Button
00271   *            @arg  BUTTON_TAMPER: Tamper P
ush Button
00272   * @param  Button_Mode: Button mode
00273   *          This parameter can be one of th
e following values:
00274   *            @arg  BUTTON_MODE_GPIO: Butto
n will be used as simple IO
00275   *            @arg  BUTTON_MODE_EXTI: Butto
n will be connected to EXTI line
00276   *                                    with
interrupt generation capability
00277   */
```

```
00278 void BSP_PB_Init(Button_TypeDef Button, Butt
onMode_TypeDef Button_Mode)
00279 {
00280   GPIO_InitTypeDef GPIO_InitStruct;
00281
00282   /* Enable the BUTTON clock */
00283   BUTTONx_GPIO_CLK_ENABLE(Button);
00284
00285   if(Button_Mode == BUTTON_MODE_GPIO)
00286   {
00287     /* Configure Button pin as input */
00288     GPIO_InitStruct.Pin = BUTTON_PIN[Button]
;
00289     GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
00290     GPIO_InitStruct.Pull = GPIO_NOPULL;
00291     GPIO_InitStruct.Speed = GPIO_SPEED_FAST;
00292     HAL_GPIO_Init(BUTTON_PORT[Button], &GPIO
_InitStruct);
00293   }
00294
00295   if(Button_Mode == BUTTON_MODE_EXTI)
00296   {
00297     /* Configure Button pin as input with Ex
ternal interrupt */
00298     GPIO_InitStruct.Pin = BUTTON_PIN[Button]
;
00299     GPIO_InitStruct.Pull = GPIO_NOPULL;
00300     GPIO_InitStruct.Speed = GPIO_SPEED_FAST;
00301
00302     if(Button != BUTTON_WAKEUP)
00303     {
00304       GPIO_InitStruct.Mode = GPIO_MODE_IT_FA
LLING;
00305     }
00306     else
00307     {
00308       GPIO_InitStruct.Mode = GPIO_MODE_IT_RI
```

```
SING;
00309     }
00310
00311     HAL_GPIO_Init(BUTTON_PORT[Button], &GPIO
_InitStruct);
00312
00313     /* Enable and set Button EXTI Interrupt
to the lowest priority */
00314     HAL_NVIC_SetPriority((IRQn_Type)(BUTTON_
IRQn[Button]), 0x0F, 0x00);
00315     HAL_NVIC_EnableIRQ((IRQn_Type)(BUTTON_IR
Qn[Button]));
00316   }
00317 }
00318
00319 /**
00320   * @brief  Returns the selected button stat
e.
00321   * @param  Button: Button to be checked
00322   *          This parameter can be one of th
e following values:
00323   *            @arg  BUTTON_WAKEUP: Wakeup P
ush Button
00324   *            @arg  BUTTON_TAMPER: Tamper P
ush Button
00325   *            @arg  BUTTON_KEY: Key Push Bu
tton
00326   * @retval The Button GPIO pin value
00327   */
00328 uint32_t BSP_PB_GetState(Button_TypeDef Butt
on)
00329 {
00330   return HAL_GPIO_ReadPin(BUTTON_PORT[Button
], BUTTON_PIN[Button]);
00331 }
00332
00333 /**
```

```
00334   * @brief  Configures COM port.
00335   * @param  COM: COM port to be configured.
00336   *          This parameter can be one of th
e following values:
00337   *            @arg  COM1
00338   *            @arg  COM2
00339   * @param  huart: Pointer to a UART_HandleT
ypeDef structure that contains the
00340   *                configuration information
 for the specified USART peripheral.
00341   */
00342 void BSP_COM_Init(COM_TypeDef COM, UART_Hand
leTypeDef *huart)
00343 {
00344   GPIO_InitTypeDef GPIO_InitStruct;
00345
00346   /* Enable GPIO clock */
00347   EVAL_COMx_TX_GPIO_CLK_ENABLE(COM);
00348   EVAL_COMx_RX_GPIO_CLK_ENABLE(COM);
00349
00350   /* Enable USART clock */
00351   EVAL_COMx_CLK_ENABLE(COM);
00352
00353   /* Configure USART Tx as alternate functio
n */
00354   GPIO_InitStruct.Pin = COM_TX_PIN[COM];
00355   GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00356   GPIO_InitStruct.Speed = GPIO_SPEED_FAST;
00357   GPIO_InitStruct.Pull = GPIO_PULLUP;
00358   GPIO_InitStruct.Alternate = COM_TX_AF[COM]
;
00359   HAL_GPIO_Init(COM_TX_PORT[COM], &GPIO_Init
Struct);
00360
00361   /* Configure USART Rx as alternate functio
n */
00362   GPIO_InitStruct.Pin = COM_RX_PIN[COM];
```

```
00363   GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00364   GPIO_InitStruct.Alternate = COM_RX_AF[COM]
;
00365   HAL_GPIO_Init(COM_RX_PORT[COM], &GPIO_Init
Struct);
00366
00367   /* USART configuration */
00368   huart->Instance = COM_USART[COM];
00369   HAL_UART_Init(huart);
00370 }
00371
00372 /**
00373   * @brief  Configures joystick GPIO and EXT
I modes.
00374   * @param  Joy_Mode: Button mode.
00375   *          This parameter can be one of th
e following values:
00376   *            @arg  JOY_MODE_GPIO: Joystick
 pins will be used as simple IOs
00377   *            @arg  JOY_MODE_EXTI: Joystick
 pins will be connected to EXTI line
00378   *                                 with int
errupt generation capability
00379   * @retval IO_OK: if all initializations ar
e OK. Other value if error.
00380   */
00381 uint8_t BSP_JOY_Init(JOYMode_TypeDef Joy_Mod
e)
00382 {
00383   uint8_t ret = 0;
00384
00385   /* Initialize the IO functionalities */
00386   ret = BSP_IO_Init();
00387
00388   /* Configure joystick pins in IT mode */
00389   if(Joy_Mode == JOY_MODE_EXTI)
00390   {
```

```
00391     /* Configure IO interrupt acquisition mo
de */
00392     BSP_IO_ConfigPin(JOY_ALL_PINS, IO_MODE_I
T_FALLING_EDGE);
00393   }
00394
00395   return ret;
00396 }
00397
00398 /**
00399   * @brief  Returns the current joystick sta
tus.
00400   * @retval Code of the joystick key pressed
00401   *          This code can be one of the fol
lowing values:
00402   *            @arg  JOY_NONE
00403   *            @arg  JOY_SEL
00404   *            @arg  JOY_DOWN
00405   *            @arg  JOY_LEFT
00406   *            @arg  JOY_RIGHT
00407   *            @arg  JOY_UP
00408   */
00409 JOYState_TypeDef BSP_JOY_GetState(void)
00410 {
00411   uint16_t tmp = 0;
00412
00413   /* Read the status joystick pins */
00414   tmp = BSP_IO_ReadPin(JOY_ALL_PINS);
00415
00416   /* Check the pressed keys */
00417   if((tmp & JOY_NONE_PIN) == JOY_NONE)
00418   {
00419     return(JOYState_TypeDef) JOY_NONE;
00420   }
00421   else if(!(tmp & JOY_SEL_PIN))
00422   {
00423     return(JOYState_TypeDef) JOY_SEL;
```

```
00424   }
00425   else if(!(tmp & JOY_DOWN_PIN))
00426   {
00427     return(JOYState_TypeDef) JOY_DOWN;
00428   }
00429   else if(!(tmp & JOY_LEFT_PIN))
00430   {
00431     return(JOYState_TypeDef) JOY_LEFT;
00432   }
00433   else if(!(tmp & JOY_RIGHT_PIN))
00434   {
00435     return(JOYState_TypeDef) JOY_RIGHT;
00436   }
00437   else if(!(tmp & JOY_UP_PIN))
00438   {
00439     return(JOYState_TypeDef) JOY_UP;
00440   }
00441   else
00442   {
00443     return(JOYState_TypeDef) JOY_NONE;
00444   }
00445 }
00446
00447 /**
00448   * @brief  Check TS3510 touch screen presen
ce
00449   * @retval Return 0 if TS3510 is detected,
return 1 if not detected
00450   */
00451 uint8_t BSP_TS3510_IsDetected(void)
00452 {
00453   HAL_StatusTypeDef status = HAL_OK;
00454   uint32_t error = 0;
00455   uint8_t a_buffer;
00456
00457   uint8_t tmp_buffer[2] = {0x81, 0x08};
00458
```

```
00459   /* Prepare for LCD read data */
00460   IOE_WriteMultiple(TS3510_I2C_ADDRESS, 0x8A
, tmp_buffer, 2);
00461
00462   status = HAL_I2C_Mem_Read(&heval_I2c, TS35
10_I2C_ADDRESS, 0x8A, I2C_MEMADD_SIZE_8BIT, &a_buf
fer, 1, 1000);
00463
00464   /* Check the communication status */
00465   if(status != HAL_OK)
00466   {
00467     error = (uint32_t)HAL_I2C_GetError(&heva
l_I2c);
00468
00469     /* I2C error occured */
00470     I2Cx_Error(TS3510_I2C_ADDRESS);
00471
00472     if(error == HAL_I2C_ERROR_AF)
00473     {
00474       return 1;
00475     }
00476   }
00477   return 0;
00478 }
00479 /*************************************************
**********************************
00480                                     BUS OPERATIONS
00481 *************************************************
**********************************/
00482
00483 /****************************** I2C Routine
s ******************************/
00484 /**
00485   * @brief  Initializes I2C MSP.
00486   */
00487 static void I2Cx_MspInit(void)
00488 {
```

```
00489   GPIO_InitTypeDef  GPIO_InitStruct;
00490
00491   /*** Configure the GPIOs ***/
00492   /* Enable GPIO clock */
00493   EVAL_I2Cx_SCL_SDA_GPIO_CLK_ENABLE();
00494
00495   /* Configure I2C Tx as alternate function
*/
00496   GPIO_InitStruct.Pin = EVAL_I2Cx_SCL_PIN;
00497   GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
00498   GPIO_InitStruct.Pull = GPIO_NOPULL;
00499   GPIO_InitStruct.Speed = GPIO_SPEED_FAST;
00500   GPIO_InitStruct.Alternate = EVAL_I2Cx_SCL_
SDA_AF;
00501   HAL_GPIO_Init(EVAL_I2Cx_SCL_SDA_GPIO_PORT,
 &GPIO_InitStruct);
00502
00503   /* Configure I2C Rx as alternate function
*/
00504   GPIO_InitStruct.Pin = EVAL_I2Cx_SDA_PIN;
00505   HAL_GPIO_Init(EVAL_I2Cx_SCL_SDA_GPIO_PORT,
 &GPIO_InitStruct);
00506
00507   /*** Configure the I2C peripheral ***/
00508   /* Enable I2C clock */
00509   EVAL_I2Cx_CLK_ENABLE();
00510
00511   /* Force the I2C peripheral clock reset */

00512   EVAL_I2Cx_FORCE_RESET();
00513
00514   /* Release the I2C peripheral clock reset
*/
00515   EVAL_I2Cx_RELEASE_RESET();
00516
00517   /* Enable and set I2Cx Interrupt to a lowe
r priority */
```

```
00518   HAL_NVIC_SetPriority(EVAL_I2Cx_EV_IRQn, 0x
05, 0);
00519   HAL_NVIC_EnableIRQ(EVAL_I2Cx_EV_IRQn);
00520
00521   /* Enable and set I2Cx Interrupt to a lowe
r priority */
00522   HAL_NVIC_SetPriority(EVAL_I2Cx_ER_IRQn, 0x
05, 0);
00523   HAL_NVIC_EnableIRQ(EVAL_I2Cx_ER_IRQn);
00524 }
00525
00526 /**
00527   * @brief  Initializes I2C HAL.
00528   */
00529 static void I2Cx_Init(void)
00530 {
00531   if(HAL_I2C_GetState(&heval_I2c) == HAL_I2C
_STATE_RESET)
00532   {
00533     heval_I2c.Instance = I2C1;
00534     heval_I2c.Init.ClockSpeed      = BSP_I2C
_SPEED;
00535     heval_I2c.Init.DutyCycle       = I2C_DUT
YCYCLE_2;
00536     heval_I2c.Init.OwnAddress1     = 0;
00537     heval_I2c.Init.AddressingMode  = I2C_ADD
RESSINGMODE_7BIT;
00538     heval_I2c.Init.DualAddressMode = I2C_DUA
LADDRESS_DISABLED;
00539     heval_I2c.Init.OwnAddress2     = 0;
00540     heval_I2c.Init.GeneralCallMode = I2C_GEN
ERALCALL_DISABLED;
00541     heval_I2c.Init.NoStretchMode   = I2C_NOS
TRETCH_DISABLED;
00542
00543     /* Init the I2C */
00544     I2Cx_MspInit();
```

```
00545     HAL_I2C_Init(&heval_I2c);
00546   }
00547 }
00548
00549 /**
00550   * @brief  Configures I2C Interrupt.
00551   */
00552 static void I2Cx_ITConfig(void)
00553 {
00554   static uint8_t I2C_IT_Enabled = 0;
00555   GPIO_InitTypeDef  GPIO_InitStruct;
00556
00557   if(I2C_IT_Enabled == 0)
00558   {
00559     I2C_IT_Enabled = 1;
00560     /* Enable the GPIO EXTI clock */
00561     __GPIOI_CLK_ENABLE();
00562     __SYSCFG_CLK_ENABLE();
00563
00564     GPIO_InitStruct.Pin   = GPIO_PIN_8;
00565     GPIO_InitStruct.Pull  = GPIO_NOPULL;
00566     GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
00567     GPIO_InitStruct.Mode  = GPIO_MODE_IT_FAL
LING;
00568     HAL_GPIO_Init(GPIOI, &GPIO_InitStruct);
00569
00570     /* Enable and set GPIO EXTI Interrupt to
 the lowest priority */
00571     HAL_NVIC_SetPriority((IRQn_Type)(EXTI9_5
_IRQn), 0x0F, 0x0F);
00572     HAL_NVIC_EnableIRQ((IRQn_Type)(EXTI9_5_I
RQn));
00573   }
00574 }
00575
00576 /**
00577   * @brief  Writes a single data.
```

```
00578   * @param  Addr: I2C address
00579   * @param  Reg: Register address
00580   * @param  Value: Data to be written
00581   */
00582 static void I2Cx_Write(uint8_t Addr, uint8_t
 Reg, uint8_t Value)
00583 {
00584   HAL_StatusTypeDef status = HAL_OK;
00585
00586   status = HAL_I2C_Mem_Write(&heval_I2c, Add
r, (uint16_t)Reg, I2C_MEMADD_SIZE_8BIT, &Value, 1,
 100);
00587
00588   /* Check the communication status */
00589   if(status != HAL_OK)
00590   {
00591     /* Execute user timeout callback */
00592     I2Cx_Error(Addr);
00593   }
00594 }
00595
00596 /**
00597   * @brief  Reads a single data.
00598   * @param  Addr: I2C address
00599   * @param  Reg: Register address
00600   * @retval Read data
00601   */
00602 static uint8_t I2Cx_Read(uint8_t Addr, uint8
_t Reg)
00603 {
00604   HAL_StatusTypeDef status = HAL_OK;
00605   uint8_t Value = 0;
00606
00607   status = HAL_I2C_Mem_Read(&heval_I2c, Addr
, Reg, I2C_MEMADD_SIZE_8BIT, &Value, 1, 1000);
00608
00609   /* Check the communication status */
```

```
00610   if(status != HAL_OK)
00611   {
00612     /* Execute user timeout callback */
00613     I2Cx_Error(Addr);
00614   }
00615   return Value;
00616 }
00617
00618 /**
00619   * @brief  Reads multiple data.
00620   * @param  Addr: I2C address
00621   * @param  Reg: Reg address
00622   * @param  MemAddress: Internal memory addr
ess
00623   * @param  Buffer: Pointer to data buffer
00624   * @param  Length: Length of the data
00625   * @retval Number of read data
00626   */
00627 static HAL_StatusTypeDef I2Cx_ReadMultiple(u
int8_t Addr, uint16_t Reg, uint16_t MemAddress, ui
nt8_t *Buffer, uint16_t Length)
00628 {
00629   HAL_StatusTypeDef status = HAL_OK;
00630
00631   if(Addr == EXC7200_I2C_ADDRESS)
00632   {
00633     status = HAL_I2C_Master_Receive(&heval_I
2c, Addr, Buffer, Length, 1000);
00634   }
00635   else
00636   {
00637     status = HAL_I2C_Mem_Read(&heval_I2c, Ad
dr, (uint16_t)Reg, MemAddress, Buffer, Length, 100
0);
00638   }
00639
00640   /* Check the communication status */
```

```
00641   if(status != HAL_OK)
00642   {
00643     /* I2C error occured */
00644     I2Cx_Error(Addr);
00645   }
00646   return status;
00647 }
00648
00649 /**
00650   * @brief  Writes a value in a register of
the device through BUS in using DMA mode.
00651   * @param  Addr: Device address on BUS Bus.

00652   * @param  Reg: The target register address
 to write
00653   * @param  MemAddress: Internal memory addr
ess
00654   * @param  Buffer: The target register valu
e to be written
00655   * @param  Length: buffer size to be written

00656   * @retval HAL status
00657   */
00658 static HAL_StatusTypeDef I2Cx_WriteMultiple(
uint8_t Addr, uint16_t Reg, uint16_t MemAddress, u
int8_t *Buffer, uint16_t Length)
00659 {
00660   HAL_StatusTypeDef status = HAL_OK;
00661
00662   status = HAL_I2C_Mem_Write(&heval_I2c, Add
r, (uint16_t)Reg, MemAddress, Buffer, Length, 1000
);
00663
00664   /* Check the communication status */
00665   if(status != HAL_OK)
00666   {
00667     /* Re-Initialize the I2C Bus */
```

```
00668     I2Cx_Error(Addr);
00669   }
00670   return status;
00671 }
00672
00673 /**
00674   * @brief  Checks if target device is ready
 for communication.
00675   * @note   This function is used with Memor
y devices
00676   * @param  DevAddress: Target device address

00677   * @param  Trials: Number of trials
00678   * @retval HAL status
00679   */
00680 static HAL_StatusTypeDef I2Cx_IsDeviceReady(
uint16_t DevAddress, uint32_t Trials)
00681 {
00682   return (HAL_I2C_IsDeviceReady(&heval_I2c,
DevAddress, Trials, 1000));
00683 }
00684
00685 /**
00686   * @brief  Manages error callback by re-ini
tializing I2C.
00687   * @param  Addr: I2C Address
00688   */
00689 static void I2Cx_Error(uint8_t Addr)
00690 {
00691   /* De-initialize the I2C communication bus
 */
00692   HAL_I2C_DeInit(&heval_I2c);
00693
00694   /* Re-Initialize the I2C communication bus
 */
00695   I2Cx_Init();
00696 }
```

```
00697
00698 /*****************************************
*********************************
00699                                     LINK OPERATIONS
00700 *****************************************
*********************************/
00701
00702 /********************************** LINK IOE
**********************************/
00703
00704 /**
00705   * @brief  Initializes IOE low level.
00706   */
00707 void IOE_Init(void)
00708 {
00709   I2Cx_Init();
00710 }
00711
00712 /**
00713   * @brief  Configures IOE low level interru
pt.
00714   */
00715 void IOE_ITConfig(void)
00716 {
00717   I2Cx_ITConfig();
00718 }
00719
00720 /**
00721   * @brief  IOE writes single data.
00722   * @param  Addr: I2C address
00723   * @param  Reg: Register address
00724   * @param  Value: Data to be written
00725   */
00726 void IOE_Write(uint8_t Addr, uint8_t Reg, ui
nt8_t Value)
00727 {
00728   I2Cx_Write(Addr, Reg, Value);
```

```
00729 }
00730
00731 /**
00732   * @brief  IOE reads single data.
00733   * @param  Addr: I2C address
00734   * @param  Reg: Register address
00735   * @retval Read data
00736   */
00737 uint8_t IOE_Read(uint8_t Addr, uint8_t Reg)
00738 {
00739   return I2Cx_Read(Addr, Reg);
00740 }
00741
00742 /**
00743   * @brief  IOE reads multiple data.
00744   * @param  Addr: I2C address
00745   * @param  Reg: Register address
00746   * @param  Buffer: Pointer to data buffer
00747   * @param  Length: Length of the data
00748   * @retval Number of read data
00749   */
00750 uint16_t IOE_ReadMultiple(uint8_t Addr, uint
8_t Reg, uint8_t *Buffer, uint16_t Length)
00751 {
00752  return I2Cx_ReadMultiple(Addr, (uint16_t)Re
g, I2C_MEMADD_SIZE_8BIT, Buffer, Length);
00753 }
00754
00755 /**
00756   * @brief  IOE writes multiple data.
00757   * @param  Addr: I2C address
00758   * @param  Reg: Register address
00759   * @param  Buffer: Pointer to data buffer
00760   * @param  Length: Length of the data
00761   */
00762 void IOE_WriteMultiple(uint8_t Addr, uint8_t
 Reg, uint8_t *Buffer, uint16_t Length)
```

```
00763 {
00764   I2Cx_WriteMultiple(Addr, (uint16_t)Reg, I2
C_MEMADD_SIZE_8BIT, Buffer, Length);
00765 }
00766
00767 /**
00768   * @brief  IOE delay
00769   * @param  Delay: Delay in ms
00770   */
00771 void IOE_Delay(uint32_t Delay)
00772 {
00773   HAL_Delay(Delay);
00774 }
00775
00776 /******************************** LINK AUDI
O *********************************/
00777
00778 /**
00779   * @brief  Initializes Audio low level.
00780   */
00781 void AUDIO_IO_Init(void)
00782 {
00783   I2Cx_Init();
00784 }
00785
00786 /**
00787   * @brief  DeInitializes Audio low level.
00788   */
00789 void AUDIO_IO_DeInit(void)
00790 {
00791
00792 }
00793
00794 /**
00795   * @brief  Writes a single data.
00796   * @param  Addr: I2C address
00797   * @param  Reg: Reg address
```

```
00798   * @param  Value: Data to be written
00799   */
00800 void AUDIO_IO_Write(uint8_t Addr, uint16_t R
eg, uint16_t Value)
00801 {
00802   uint16_t tmp = Value;
00803
00804   Value = ((uint16_t)(tmp >> 8) & 0x00FF);
00805
00806   Value |= ((uint16_t)(tmp << 8)& 0xFF00);
00807
00808   I2Cx_WriteMultiple(Addr, Reg, I2C_MEMADD_S
IZE_16BIT,(uint8_t*)&Value, 2);
00809 }
00810
00811 /**
00812   * @brief  Reads a single data.
00813   * @param  Addr: I2C address
00814   * @param  Reg: Reg address
00815   * @retval Data to be read
00816   */
00817 uint16_t AUDIO_IO_Read(uint8_t Addr, uint16_
t Reg)
00818 {
00819   uint16_t Read_Value = 0, tmp = 0;
00820
00821   I2Cx_ReadMultiple(Addr, Reg, I2C_MEMADD_SI
ZE_16BIT, (uint8_t*)&Read_Value, 2);
00822
00823   tmp = ((uint16_t)(Read_Value >> 8) & 0x00F
F);
00824
00825   tmp |= ((uint16_t)(Read_Value << 8)& 0xFF0
0);
00826
00827   Read_Value = tmp;
00828
```

```
00829   return Read_Value;
00830 }
00831
00832 /**
00833   * @brief  AUDIO Codec delay
00834   * @param  Delay: Delay in ms
00835   */
00836 void AUDIO_IO_Delay(uint32_t Delay)
00837 {
00838   HAL_Delay(Delay);
00839 }
00840
00841 /********************************* LINK CAME
RA ***************************/
00842
00843 /**
00844   * @brief  Initializes Camera low level.
00845   */
00846 void CAMERA_IO_Init(void)
00847 {
00848   I2Cx_Init();
00849 }
00850
00851 /**
00852   * @brief  Camera writes single data.
00853   * @param  Addr: I2C address
00854   * @param  Reg: Register address
00855   * @param  Value: Data to be written
00856   */
00857 void CAMERA_IO_Write(uint8_t Addr, uint8_t R
eg, uint8_t Value)
00858 {
00859   I2Cx_Write(Addr, Reg, Value);
00860 }
00861
00862 /**
00863   * @brief  Camera reads single data.
```

```
00864   * @param  Addr: I2C address
00865   * @param  Reg: Register address
00866   * @retval Read data
00867   */
00868 uint8_t CAMERA_IO_Read(uint8_t Addr, uint8_t
 Reg)
00869 {
00870   return I2Cx_Read(Addr, Reg);
00871 }
00872
00873 /**
00874   * @brief  Camera delay
00875   * @param  Delay: Delay in ms
00876   */
00877 void CAMERA_Delay(uint32_t Delay)
00878 {
00879   HAL_Delay(Delay);
00880 }
00881
00882 /****************************** LINK I2C E
EPROM ***************************/
00883
00884 /**
00885   * @brief  Initializes peripherals used by
the I2C EEPROM driver.
00886   */
00887 void EEPROM_IO_Init(void)
00888 {
00889   I2Cx_Init();
00890 }
00891
00892 /**
00893   * @brief  Write data to I2C EEPROM driver
in using DMA channel.
00894   * @param  DevAddress: Target device address

00895   * @param  MemAddress: Internal memory addr
```

```
ess
00896   * @param  pBuffer: Pointer to data buffer
00897   * @param  BufferSize: Amount of data to be
 sent
00898   * @retval HAL status
00899   */
00900 HAL_StatusTypeDef EEPROM_IO_WriteData(uint16
_t DevAddress, uint16_t MemAddress, uint8_t* pBuff
er, uint32_t BufferSize)
00901 {
00902   return (I2Cx_WriteMultiple(DevAddress, Mem
Address, I2C_MEMADD_SIZE_16BIT, pBuffer, BufferSiz
e));
00903 }
00904
00905 /**
00906   * @brief  Read data from I2C EEPROM driver
 in using DMA channel.
00907   * @param  DevAddress: Target device address

00908   * @param  MemAddress: Internal memory addr
ess
00909   * @param  pBuffer: Pointer to data buffer
00910   * @param  BufferSize: Amount of data to be
 read
00911   * @retval HAL status
00912   */
00913 HAL_StatusTypeDef EEPROM_IO_ReadData(uint16_
t DevAddress, uint16_t MemAddress, uint8_t* pBuffe
r, uint32_t BufferSize)
00914 {
00915   return (I2Cx_ReadMultiple(DevAddress, MemA
ddress, I2C_MEMADD_SIZE_16BIT, pBuffer, BufferSize
));
00916 }
00917
00918 /**
```

```
00919   * @brief  Checks if target device is ready
 for communication.
00920   * @note   This function is used with Memor
y devices
00921   * @param  DevAddress: Target device address

00922   * @param  Trials: Number of trials
00923   * @retval HAL status
00924   */
00925 HAL_StatusTypeDef EEPROM_IO_IsDeviceReady(ui
nt16_t DevAddress, uint32_t Trials)
00926 {
00927   return (I2Cx_IsDeviceReady(DevAddress, Tri
als));
00928 }
00929
00930 /**
00931   * @}
00932   */
00933
00934 /**
00935   * @}
00936   */
00937
00938 /**
00939   * @}
00940   */
00941
00942 /**
00943   * @}
00944   */
00945
00946 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

User Manual by  1.7.6.1

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_audio.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval_audio.h
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm324x9i_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 __STM324x9I_EVAL_AUDIO_H
00041 #define __STM324x9I_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 "stm324x9i_eval.h"
00051 #include "../../../Middlewares/ST/STM32_Audi
o/Addons/PDM/pdm_filter.h"
00052
00053 /** @addtogroup BSP
00054   * @{
00055   */
00056
00057 /** @addtogroup STM324x9I_EVAL
00058   * @{
00059   */
00060
00061 /** @addtogroup STM324x9I_EVAL_AUDIO
00062   * @{
00063   */
00064
00065 /** @defgroup STM324x9I_EVAL_AUDIO_Exported_
Types STM324x9I EVAL AUDIO Exported Types
00066   * @{
```

```
00067   */
00068 /**
00069   * @}
00070   */
00071
00072 /** @defgroup STM324x9I_EVAL_AUDIO_Exported_
Constants STM324x9I EVAL AUDIO Exported Constants
00073   * @{
00074   */
00075
00076 /*-----------------------------------------
-----------------------------------
00077                               USER SAI defines p
arameters
00078  -----------------------------------------
-----------------------------------*/
00079 /** @defgroup CODEC_AudioFrame_SLOT_TDMMode
CODEC AudioFrame SLOT TDMMode
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
 SAI1_Block_B
00099 #define AUDIO_SAIx_CLK_ENABLE()
 __SAI1_CLK_ENABLE()
00100 #define AUDIO_SAIx_MCLK_SCK_SD_FS_AF
 GPIO_AF6_SAI1
00101
00102 #define AUDIO_SAIx_MCLK_SCK_SD_FS_ENABLE()
 __GPIOF_CLK_ENABLE()
00103 #define AUDIO_SAIx_FS_PIN
 GPIO_PIN_9
00104 #define AUDIO_SAIx_SCK_PIN
 GPIO_PIN_8
00105 #define AUDIO_SAIx_SD_PIN
 GPIO_PIN_6
00106 #define AUDIO_SAIx_MCK_PIN
 GPIO_PIN_7
00107 #define AUDIO_SAIx_MCLK_SCK_SD_FS_GPIO_PORT
 GPIOF
00108
00109
00110 /* SAI DMA Stream definitions */
00111 #define AUDIO_SAIx_DMAx_CLK_ENABLE()
 __DMA2_CLK_ENABLE()
00112 #define AUDIO_SAIx_DMAx_STREAM
 DMA2_Stream5
```

```
00113 #define AUDIO_SAIx_DMAx_CHANNEL
 DMA_CHANNEL_0
00114 #define AUDIO_SAIx_DMAx_IRQ
 DMA2_Stream5_IRQn
00115 #define AUDIO_SAIx_DMAx_PERIPH_DATA_SIZE
 DMA_PDATAALIGN_HALFWORD
00116 #define AUDIO_SAIx_DMAx_MEM_DATA_SIZE
 DMA_MDATAALIGN_HALFWORD
00117 #define DMA_MAX_SZE
 0xFFFF
00118
00119 #define AUDIO_SAIx_DMAx_IRQHandler
 DMA2_Stream5_IRQHandler
00120
00121 /* Select the interrupt preemption priority
for the DMA interrupt */
00122 #define AUDIO_OUT_IRQ_PREPRIO           5
/* Select the preemption priority level(0 is the h
ighest) */
00123
00124 /*-----------------------------------------
-----------------------------------
00125                             AUDIO IN CONFIGURATI
ON
00126 -----------------------------------------
-----------------------------------*/
00127 /* SPI Configuration defines */
00128 #define AUDIO_I2Sx
SPI3
00129 #define AUDIO_I2Sx_CLK_ENABLE()
__SPI3_CLK_ENABLE()
00130 #define AUDIO_I2Sx_SCK_PIN
 GPIO_PIN_3
00131 #define AUDIO_I2Sx_SCK_GPIO_PORT
 GPIOB
00132 #define AUDIO_I2Sx_SCK_GPIO_CLK_ENABLE()
 __GPIOB_CLK_ENABLE()
```

```
00133 #define AUDIO_I2Sx_SCK_AF
 GPIO_AF6_SPI3
00134
00135 #define AUDIO_I2Sx_SD_PIN
GPIO_PIN_6
00136 #define AUDIO_I2Sx_SD_GPIO_PORT
GPIOD
00137 #define AUDIO_I2Sx_SD_GPIO_CLK_ENABLE()
__GPIOD_CLK_ENABLE()
00138 #define AUDIO_I2Sx_SD_AF
GPIO_AF5_I2S3ext
00139
00140 /* I2S DMA Stream Rx definitions */
00141 #define AUDIO_I2Sx_DMAx_CLK_ENABLE()
__DMA1_CLK_ENABLE()
00142 #define AUDIO_I2Sx_DMAx_STREAM
DMA1_Stream2
00143 #define AUDIO_I2Sx_DMAx_CHANNEL
DMA_CHANNEL_0
00144 #define AUDIO_I2Sx_DMAx_IRQ
DMA1_Stream2_IRQn
00145 #define AUDIO_I2Sx_DMAx_PERIPH_DATA_SIZE
DMA_PDATAALIGN_HALFWORD
00146 #define AUDIO_I2Sx_DMAx_MEM_DATA_SIZE
DMA_MDATAALIGN_HALFWORD
00147
00148 #define AUDIO_I2Sx_DMAx_IRQHandler
DMA1_Stream2_IRQHandler
00149
00150 /* Select the interrupt preemption priority
and subpriority for the IT/DMA interrupt */
00151 #define AUDIO_IN_IRQ_PREPRIO
6   /* Select the preemption priority level(0 is t
he highest) */
00152
00153
00154 /* Two channels are used:
```

```
00155    - one channel as input which is connected
 to I2S SCK in stereo mode
00156    - one channel as output which divides the
 frequency on the input
00157 */
00158
00159 #define AUDIO_TIMx_CLK_ENABLE()
__TIM3_CLK_ENABLE()
00160 #define AUDIO_TIMx_CLK_DISABLE()
__TIM3_CLK_DISABLE()
00161 #define AUDIO_TIMx
TIM3
00162 #define AUDIO_TIMx_IN_CHANNEL
TIM_CHANNEL_1
00163 #define AUDIO_TIMx_OUT_CHANNEL
TIM_CHANNEL_2 /* Select channel 2 as output */
00164 #define AUDIO_TIMx_GPIO_CLK_ENABLE()
__GPIOC_CLK_ENABLE()
00165 #define AUDIO_TIMx_GPIO
GPIOC
00166 #define AUDIO_TIMx_IN_GPIO_PIN
GPIO_PIN_6
00167 #define AUDIO_TIMx_OUT_GPIO_PIN
GPIO_PIN_7
00168 #define AUDIO_TIMx_AF
GPIO_AF2_TIM3
00169
00170 /*-----------------------------------------
-----------------------------------
00171              CONFIGURATION: Audio Driver Con
figuration parameters
00172 -----------------------------------------
-----------------------------------*/
00173
00174 #define AUDIODATA_SIZE
2   /* 16-bits audio data size */
00175
```

```
00176 /* Audio status definition */
00177 #define AUDIO_OK                         0

00178 #define AUDIO_ERROR                      1

00179 #define AUDIO_TIMEOUT                    2

00180
00181 /* AudioFreq * DataSize (2 bytes) * NumChann
els (Stereo: 2) */
00182 #define DEFAULT_AUDIO_IN_FREQ
I2S_AUDIOFREQ_16K
00183 #define DEFAULT_AUDIO_IN_BIT_RESOLUTION
16
00184 #define DEFAULT_AUDIO_IN_CHANNEL_NBR
2 /* Mono = 1, Stereo = 2 */
00185 #define DEFAULT_AUDIO_IN_VOLUME
64
00186
00187 /* PDM buffer input size */
00188 #define INTERNAL_BUFF_SIZE
128*DEFAULT_AUDIO_IN_FREQ/16000*DEFAULT_AUDIO_IN_C
HANNEL_NBR
00189 /* PCM buffer output size */
00190 #define PCM_OUT_SIZE
DEFAULT_AUDIO_IN_FREQ/1000*2
00191 #define CHANNEL_DEMUX_MASK
0x55
00192
00193 /*-----------------------------------------
-----------------------------------
00194                        OPTIONAL Configuration d
efines parameters
00195 -----------------------------------------
-----------------------------------*/
00196
00197 /* Delay for the Codec to be correctly reset
```

```
 */
00198 #define CODEC_RESET_DELAY           5
00199
00200 /**
00201   * @}
00202   */
00203
00204 /** @defgroup STM324x9I_EVAL_AUDIO_Exported_
Variables STM324x9I EVAL AUDIO Exported Variables
00205   * @{
00206   */
00207 extern __IO uint16_t AudioInVolume;
00208  /**
00209   * @}
00210   */
00211
00212 /** @defgroup STM324x9I_EVAL_AUDIO_Exported_
Macros STM324x9I EVAL AUDIO Exported Macros
00213   * @{
00214   */
00215 #define DMA_MAX(x)           (((x) <= DMA_MA
X_SZE)? (x):DMA_MAX_SZE)
00216 /**
00217   * @}
00218   */
00219
00220 /** @defgroup STM324x9I_EVAL_AUDIO_OUT_Expor
ted_Functions STM324x9I EVAL AUDIO OUT Exported Fu
nctions
00221   * @{
00222   */
00223 uint8_t BSP_AUDIO_OUT_Init(uint16_t OutputDe
vice, uint8_t Volume, uint32_t AudioFreq);
00224 uint8_t BSP_AUDIO_OUT_Play(uint16_t* pBuffer
, uint32_t Size);
00225 void    BSP_AUDIO_OUT_ChangeBuffer(uint16_t
*pData, uint16_t Size);
```

```
00226 uint8_t BSP_AUDIO_OUT_Pause(void);
00227 uint8_t BSP_AUDIO_OUT_Resume(void);
00228 uint8_t BSP_AUDIO_OUT_Stop(uint32_t Option);
00229 uint8_t BSP_AUDIO_OUT_SetVolume(uint8_t Volu
me);
00230 void    BSP_AUDIO_OUT_SetFrequency(uint32_t
AudioFreq);
00231 void    BSP_AUDIO_OUT_SetAudioFrameSlot(uint
32_t AudioFrameSlot);
00232 uint8_t BSP_AUDIO_OUT_SetMute(uint32_t Cmd);
00233 uint8_t BSP_AUDIO_OUT_SetOutputMode(uint8_t
Output);
00234
00235 /* User Callbacks: user has to implement the
se functions in his code if they are needed. */
00236 /* This function is called when the requeste
d data has been completely transferred.*/
00237 void    BSP_AUDIO_OUT_TransferComplete_CallB
ack(void);
00238
00239 /* This function is called when half of the
requested buffer has been transferred. */
00240 void    BSP_AUDIO_OUT_HalfTransfer_CallBack(
void);
00241
00242 /* This function is called when an Interrupt
 due to transfer error on or peripheral
00243    error occurs. */
00244 void    BSP_AUDIO_OUT_Error_CallBack(void);
00245
00246 /**
00247   * @}
00248   */
00249
00250 /** @defgroup STM324x9I_EVAL_AUDIO_IN_Export
ed_Functions STM324x9I EVAL AUDIO IN Exported Func
tions
```

```
00251   * @{
00252   */
00253 uint8_t BSP_AUDIO_IN_Init(uint32_t AudioFreq
, uint32_t BitRes, uint32_t ChnlNbr);
00254 uint8_t BSP_AUDIO_IN_Record(uint16_t *pData,
 uint32_t Size);
00255 uint8_t BSP_AUDIO_IN_Stop(void);
00256 uint8_t BSP_AUDIO_IN_Pause(void);
00257 uint8_t BSP_AUDIO_IN_Resume(void);
00258 uint8_t BSP_AUDIO_IN_SetVolume(uint8_t Volum
e);
00259 uint8_t BSP_AUDIO_IN_PDMToPCM(uint16_t* PDMB
uf, uint16_t* PCMBuf);
00260 /* User Callbacks: user has to implement the
se functions in his code if they are needed. */
00261 /* This function should be implemented by th
e user application.
00262    It is called into this driver when the cu
rrent buffer is filled to prepare the next
00263    buffer pointer and its size. */
00264 void    BSP_AUDIO_IN_TransferComplete_CallBa
ck(void);
00265 void    BSP_AUDIO_IN_HalfTransfer_CallBack(v
oid);
00266
00267 /* This function is called when an Interrupt
 due to transfer error on or peripheral
00268    error occurs. */
00269 void    BSP_AUDIO_IN_Error_Callback(void);
00270
00271 /**
00272   * @}
00273   */
00274
00275 /**
00276   * @}
00277   */
```

```
00278
00279 /**
00280   * @}
00281   */
00282
00283 /**
00284   * @}
00285   */
00286
00287 #ifdef __cplusplus
00288 }
00289 #endif
00290
00291 #endif /* __STM324x9I_EVAL_AUDIO_H */
00292
00293 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_audio.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval_audio.c
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file provides the Audio driver for the STM324x9I-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 STM324x9I-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 IOExpander has failed
 (try to 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    + Call the function BSP_EVAL_AUDIO_OUT_Pl
ay(
00060                                        pBuffer: p
ointer to the audio data file address
00061                                        Size   : s
ize of the buffer to be sent in Bytes
00062                                      )
00063       to start playing (for the first time)
from the audio file/stream.
00064    + Call the function BSP_AUDIO_OUT_Pause()
 to pause playing
00065    + Call the function BSP_AUDIO_OUT_Resume(
) to resume playing.
00066        Note. After calling BSP_AUDIO_OUT_Pau
se() function for pause, only BSP_AUDIO_OUT_Resume
() should be called
00067           for resume (it is not allowed to c
all BSP_AUDIO_OUT_Play() in this case).
00068        Note. This function should be called
only when the audio file is played or paused (not
stopped).
00069    + For each mode, you may need to implemen
t the relative callback functions into your code.
00070       The Callback functions are named AUDIO
_OUT_XXX_CallBack() and only their prototypes are
```

```
declared in
00071       the stm324x9i_eval_audio.h file. (refe
r to the example for more details on the callbacks
 implementations)
00072    + To Stop playing, to modify the volume l
evel, the frequency, the audio frame slot,
00073       the device output mode the mute or the
 stop, use the functions: BSP_AUDIO_OUT_SetVolume(
),
00074       AUDIO_OUT_SetFrequency(), BSP_AUDIO_OU
T_SetAudioFrameSlot(), BSP_AUDIO_OUT_SetOutputMode
(),
00075       BSP_AUDIO_OUT_SetMute() and BSP_AUDIO_
OUT_Stop().
00076    + The driver API and the callback functio
ns are at the end of the stm324x9i_eval_audio.h fi
le.
00077
00078 Driver architecture:
00079 --------------------
00080    + This driver provide the High Audio Laye
r: consists of the function API exported in the st
m324x9i_eval_audio.h file
00081      (BSP_AUDIO_OUT_Init(), BSP_AUDIO_OUT_Pl
ay() ...)
00082    + This driver provide also the Media Acce
ss Layer (MAL): which consists of functions allowi
ng to access the media containing/
00083      providing the audio file/stream. These
functions are also included as local functions into

00084      the stm324x9i_eval_audio_codec.c file (
SAIx_MspInit() and SAIx_Init())
00085
00086 Known Limitations:
00087 ------------------
00088    1- If the TDM Format used to paly in para
```

```
llel 2 audio Stream (the first Stream is configure
d in codec SLOT0 and second
00089       Stream in SLOT1) the Pause/Resume, vol
ume and mute feature will control the both streams.

00090    2- Parsing of audio file is not implement
ed (in order to determine audio file properties: M
ono/Stereo, Data size,
00091       File size, Audio Frequency, Audio Data
 header size ...). The configuration is fixed for
the given audio file.
00092    3- Supports only Stereo audio streaming.
00093    4- Supports only 16-bits audio data size.
00094 ==========================================
==================================*/
00095
00096 /* Includes ------------------------------
----------------------------------*/
00097 #include "stm324x9i_eval_audio.h"
00098
00099 /** @addtogroup BSP
00100   * @{
00101   */
00102
00103 /** @addtogroup STM324x9I_EVAL
00104   * @{
00105   */
00106
00107 /** @defgroup STM324x9I_EVAL_AUDIO STM324x9I
 EVAL AUDIO
00108   * @brief This file includes the low layer
driver for wm8994 Audio Codec
00109   *        available on STM324x9I-EVAL evalu
ation board(MB1045).
00110   * @{
00111   */
00112
```

```
00113 /** @defgroup STM324x9I_EVAL_AUDIO_Private_T
ypes STM324x9I EVAL AUDIO Private Types
00114   * @{
00115   */
00116 /**
00117   * @}
00118   */
00119
00120 /** @defgroup STM324x9I_EVAL_AUDIO_Private_D
efines STM324x9I EVAL AUDIO Private Defines
00121   * @{
00122   */
00123 /**
00124   * @}
00125   */
00126
00127 /** @defgroup STM324x9I_EVAL_AUDIO_Private_M
acros STM324x9I EVAL AUDIO Private Macros
00128   * @{
00129   */
00130 /**
00131   * @}
00132   */
00133
00134 /** @defgroup STM324x9I_EVAL_AUDIO_Private_V
ariables STM324x9I EVAL AUDIO Private Variables
00135   * @{
00136   */
00137 AUDIO_DrvTypeDef          *audio_drv;
00138 SAI_HandleTypeDef         haudio_out_sai;
00139 I2S_HandleTypeDef         haudio_in_i2s;
00140 TIM_HandleTypeDef         haudio_tim;
00141
00142 PDMFilter_InitStruct Filter[2];
00143 uint8_t Channel_Demux[128] = {
00144     0x00, 0x01, 0x00, 0x01, 0x02, 0x03, 0x02
, 0x03,
```

```
00145     0x00, 0x01, 0x00, 0x01, 0x02, 0x03, 0x02
, 0x03,
00146     0x04, 0x05, 0x04, 0x05, 0x06, 0x07, 0x06
, 0x07,
00147     0x04, 0x05, 0x04, 0x05, 0x06, 0x07, 0x06
, 0x07,
00148     0x00, 0x01, 0x00, 0x01, 0x02, 0x03, 0x02
, 0x03,
00149     0x00, 0x01, 0x00, 0x01, 0x02, 0x03, 0x02
, 0x03,
00150     0x04, 0x05, 0x04, 0x05, 0x06, 0x07, 0x06
, 0x07,
00151     0x04, 0x05, 0x04, 0x05, 0x06, 0x07, 0x06
, 0x07,
00152     0x08, 0x09, 0x08, 0x09, 0x0a, 0x0b, 0x0a
, 0x0b,
00153     0x08, 0x09, 0x08, 0x09, 0x0a, 0x0b, 0x0a
, 0x0b,
00154     0x0c, 0x0d, 0x0c, 0x0d, 0x0e, 0x0f, 0x0e
, 0x0f,
00155     0x0c, 0x0d, 0x0c, 0x0d, 0x0e, 0x0f, 0x0e
, 0x0f,
00156     0x08, 0x09, 0x08, 0x09, 0x0a, 0x0b, 0x0a
, 0x0b,
00157     0x08, 0x09, 0x08, 0x09, 0x0a, 0x0b, 0x0a
, 0x0b,
00158     0x0c, 0x0d, 0x0c, 0x0d, 0x0e, 0x0f, 0x0e
, 0x0f,
00159     0x0c, 0x0d, 0x0c, 0x0d, 0x0e, 0x0f, 0x0e
, 0x0f
00160 };
00161
00162 uint16_t __IO AudioInVolume = DEFAULT_AUDIO_
IN_VOLUME;
00163
00164 /**
00165   * @}
```

```
00166   */
00167
00168 /** @defgroup STM324x9I_EVAL_AUDIO_Private_F
unction_Prototypes STM324x9I EVAL AUDIO Private Fu
nction Prototypes
00169   * @{
00170   */
00171 static void SAIx_MspInit(void);
00172 static void SAIx_Init(uint32_t AudioFreq);
00173 static void I2Sx_MspInit(void);
00174 static void I2Sx_Init(uint32_t AudioFreq);
00175 static void TIMx_IC_MspInit(TIM_HandleTypeDe
f *htim);
00176 static void TIMx_Init(void);
00177 static void PDMDecoder_Init(uint32_t AudioFr
eq, uint32_t ChnlNbr);
00178 /**
00179   * @}
00180   */
00181
00182 /** @defgroup STM324x9I_EVAL_AUDIO_out_Priva
te_Functions STM324x9I EVAL AUDIO OUT Private Func
tions
00183   * @{
00184   */
00185
00186 /**
00187   * @brief  Configures the audio peripherals.

00188   * @param  OutputDevice: OUTPUT_DEVICE_SPEA
KER, OUTPUT_DEVICE_HEADPHONE,
00189   *                       or OUTPUT_DEVICE_B
OTH.
00190   * @param  Volume: Initial volume level (fr
om 0 (Mute) to 100 (Max))
00191   * @param  AudioFreq: Audio frequency used
to play the audio stream.
```

```
00192   * @note   The I2S PLL input clock must be
done in the user application.
00193   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00194   */
00195 uint8_t BSP_AUDIO_OUT_Init(uint16_t OutputDe
vice, uint8_t Volume, uint32_t AudioFreq)
00196 {
00197   uint8_t ret = AUDIO_ERROR;
00198   uint32_t deviceid = 0x00;
00199   RCC_PeriphCLKInitTypeDef RCC_ExCLKInitStru
ct;
00200
00201   HAL_RCCEx_GetPeriphCLKConfig(&RCC_ExCLKIni
tStruct);
00202   if((AudioFreq == AUDIO_FREQUENCY_11K) || (
AudioFreq == AUDIO_FREQUENCY_22K) || (AudioFreq ==
 AUDIO_FREQUENCY_44K))
00203   {
00204     /* Configure PLLSAI prescalers */
00205     /* PLLI2S_VCO: VCO_429M
00206     SAI_CLK(first level) = PLLI2S_VCO/PLLI2S
Q = 429/2 = 214.5 Mhz
00207     SAI_CLK_x = SAI_CLK(first level)/PLLI2SD
IVQ = 214.5/19 = 11.289 Mhz */
00208     RCC_ExCLKInitStruct.PeriphClockSelection
 = RCC_PERIPHCLK_SAI_PLLI2S;
00209     RCC_ExCLKInitStruct.PLLI2S.PLLI2SN = 429
;
00210     RCC_ExCLKInitStruct.PLLI2S.PLLI2SQ = 2;
00211     RCC_ExCLKInitStruct.PLLI2SDivQ = 19;
00212     HAL_RCCEx_PeriphCLKConfig(&RCC_ExCLKInit
Struct);
00213   }
00214   else /* AUDIO_FREQUENCY_8K, AUDIO_FREQUENC
Y_16K, AUDIO_FREQUENCY_48K), AUDIO_FREQUENCY_96K */
```

```
00215   {
00216     /* SAI clock config
00217     PLLI2S_VCO: VCO_344M
00218     SAI_CLK(first level) = PLLI2S_VCO/PLLI2S
Q = 344/7 = 49.142 Mhz
00219     SAI_CLK_x = SAI_CLK(first level)/PLLI2SD
IVQ = 49.142/1 = 49.142 Mhz */
00220     RCC_ExCLKInitStruct.PeriphClockSelection
 = RCC_PERIPHCLK_SAI_PLLI2S;
00221     RCC_ExCLKInitStruct.PLLI2S.PLLI2SN = 344
;
00222     RCC_ExCLKInitStruct.PLLI2S.PLLI2SQ = 7;
00223     RCC_ExCLKInitStruct.PLLI2SDivQ = 1;
00224     HAL_RCCEx_PeriphCLKConfig(&RCC_ExCLKInit
Struct);
00225   }
00226
00227   /* SAI data transfer preparation:
00228   Prepare the Media to be used for the audio
 transfer from memory to SAI peripheral */
00229   SAIx_Init(AudioFreq);
00230
00231   /* wm8994 codec initialization */
00232   deviceid = wm8994_drv.ReadID(AUDIO_I2C_ADD
RESS);
00233
00234   if((deviceid) == WM8994_ID)
00235   {
00236     /* Initialize the audio driver structure
 */
00237     audio_drv = &wm8994_drv;
00238     ret = AUDIO_OK;
00239   }
00240   else
00241   {
00242     ret = AUDIO_ERROR;
00243   }
```

```
00244
00245   if(ret == AUDIO_OK)
00246   {
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   RCC_PeriphCLKInitTypeDef RCC_ExCLKInitStru
ct;
00436
00437   HAL_RCCEx_GetPeriphCLKConfig(&RCC_ExCLKIni
tStruct);
00438
00439   /* Update the PLL configuration according
to the new frequency */
00440   if((AudioFreq == AUDIO_FREQUENCY_11K) || (
AudioFreq == AUDIO_FREQUENCY_22K) || (AudioFreq ==
 AUDIO_FREQUENCY_44K))
00441   {
00442     /* Configure PLLSAI prescalers */
00443     /* PLLSAI_VCO: VCO_429M
00444     SAI_CLK(first level) = PLLI2S_VCO/PLLI2S
Q = 429/2 = 214.5 Mhz
00445     SAI_CLK_x = SAI_CLK(first level)/PLLI2SD
IVQ = 214.5/19 = 11.289 Mhz */
00446     RCC_ExCLKInitStruct.PeriphClockSelection
 = RCC_PERIPHCLK_SAI_PLLI2S;
00447     RCC_ExCLKInitStruct.PLLI2S.PLLI2SN = 429
;
00448     RCC_ExCLKInitStruct.PLLI2S.PLLI2SQ = 2;
00449     RCC_ExCLKInitStruct.PLLI2SDivQ = 19;
00450     HAL_RCCEx_PeriphCLKConfig(&RCC_ExCLKInit
Struct);
00451   }
00452   else /* AUDIO_FREQUENCY_8K, AUDIO_FREQUENC
Y_16K, AUDIO_FREQUENCY_48K), AUDIO_FREQUENCY_96K */

00453   {
00454     /* SAI clock config
00455     PLLI2S_VCO: VCO_344M
00456     SAI_CLK(first level) = PLLI2S_VCO/PLLI2S
Q = 344/7 = 49.142 Mhz
00457     SAI_CLK_x = SAI_CLK(first level)/PLLI2SD
IVQ = 49.142/1 = 49.142 Mhz */
```

```
00458     RCC_ExCLKInitStruct.PeriphClockSelection
 = RCC_PERIPHCLK_SAI_PLLI2S;
00459     RCC_ExCLKInitStruct.PLLI2S.PLLI2SN = 344
;
00460     RCC_ExCLKInitStruct.PLLI2S.PLLI2SQ = 7;
00461     RCC_ExCLKInitStruct.PLLI2SDivQ = 1;
00462     HAL_RCCEx_PeriphCLKConfig(&RCC_ExCLKInit
Struct);
00463   }
00464   /* Disable SAI peripheral to allow access
to SAI internal registers */
00465   __HAL_SAI_DISABLE(&haudio_out_sai);
00466
00467   /* Update the SAI audio frequency configur
ation */
00468   haudio_out_sai.Init.AudioFrequency = Audio
Freq;
00469   HAL_SAI_Init(&haudio_out_sai);
00470
00471   /* Enable SAI peripheral to generate MCLK
*/
00472   __HAL_SAI_ENABLE(&haudio_out_sai);
00473 }
00474
00475 /**
00476   * @brief  Updates the Audio frame slot con
figuration.
00477   * @param  AudioFrameSlot: specifies the au
dio Frame slot
00478   *         This parameter can be any value
of @ref CODEC_AudioFrame_SLOT_TDMMode
00479   * @note   This API should be called after
the BSP_AUDIO_OUT_Init() to adjust the
00480   *         audio frame slot.
00481   */
00482 void BSP_AUDIO_OUT_SetAudioFrameSlot(uint32_
t AudioFrameSlot)
```

```
00483 {
00484   /* Disable SAI peripheral to allow access
to SAI internal registers */
00485   __HAL_SAI_DISABLE(&haudio_out_sai);
00486
00487   /* Update the SAI audio frame slot configu
ration */
00488   haudio_out_sai.SlotInit.SlotActive = Audio
FrameSlot;
00489   HAL_SAI_Init(&haudio_out_sai);
00490
00491   /* Enable SAI peripheral to generate MCLK
*/
00492   __HAL_SAI_ENABLE(&haudio_out_sai);
00493 }
00494
00495 /**
00496   * @brief  Tx Transfer completed callbacks.
00497   * @param  hsai: SAI handle
00498   */
00499 void HAL_SAI_TxCpltCallback(SAI_HandleTypeDe
f *hsai)
00500 {
00501   /* Manage the remaining file size and new
address offset: This function
00502      should be coded by user (its prototype
is already declared in stm324x9i_eval_audio.h) */
00503   BSP_AUDIO_OUT_TransferComplete_CallBack();
00504 }
00505
00506 /**
00507   * @brief  Tx Half Transfer completed callb
acks.
00508   * @param  hsai: SAI handle
00509   */
00510 void HAL_SAI_TxHalfCpltCallback(SAI_HandleTy
peDef *hsai)
```

```
00511 {
00512   /* Manage the remaining file size and new
address offset: This function
00513      should be coded by user (its prototype
is already declared in stm324x9i_eval_audio.h) */
00514   BSP_AUDIO_OUT_HalfTransfer_CallBack();
00515 }
00516
00517 /**
00518   * @brief  SAI error callbacks.
00519   * @param  hsai: SAI handle
00520   */
00521 void HAL_SAI_ErrorCallback(SAI_HandleTypeDef
 *hsai)
00522 {
00523   BSP_AUDIO_OUT_Error_CallBack();
00524 }
00525
00526 /**
00527   * @brief  Manages the DMA full Transfer co
mplete event.
00528   */
00529 __weak void BSP_AUDIO_OUT_TransferComplete_C
allBack(void)
00530 {
00531 }
00532
00533 /**
00534   * @brief  Manages the DMA Half Transfer co
mplete event.
00535   */
00536 __weak void BSP_AUDIO_OUT_HalfTransfer_CallB
ack(void)
00537 {
00538 }
00539
00540 /**
```

```
00541   * @brief  Manages the DMA FIFO error event.

00542   */
00543 __weak void BSP_AUDIO_OUT_Error_CallBack(void
)
00544 {
00545 }
00546
00547 /*******************************************
**********************************
00548                                   Static Functions
00549 *******************************************
**********************************/
00550
00551 /**
00552   * @brief  Initializes SAI MSP.
00553   */
00554 static void SAIx_MspInit(void)
00555 {
00556   static DMA_HandleTypeDef hdma_saiTx;
00557   GPIO_InitTypeDef  GPIO_InitStruct;
00558   SAI_HandleTypeDef *hsai = &haudio_out_sai;
00559
00560   /* Enable SAI clock */
00561   AUDIO_SAIx_CLK_ENABLE();
00562
00563   /* Enable GPIO clock */
00564   AUDIO_SAIx_MCLK_SCK_SD_FS_ENABLE();
00565
00566   /* CODEC_SAI pins configuration: FS, SCK,
MCK and SD pins ------------------*/
00567   GPIO_InitStruct.Pin = AUDIO_SAIx_FS_PIN |
AUDIO_SAIx_SCK_PIN | AUDIO_SAIx_SD_PIN | AUDIO_SAI
x_MCK_PIN;
00568   GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00569   GPIO_InitStruct.Pull = GPIO_NOPULL;
00570   GPIO_InitStruct.Speed = GPIO_SPEED_HIGH;
```

```
00571   GPIO_InitStruct.Alternate = AUDIO_SAIx_MCL
K_SCK_SD_FS_AF;
00572   HAL_GPIO_Init(AUDIO_SAIx_MCLK_SCK_SD_FS_GP
IO_PORT, &GPIO_InitStruct);
00573
00574   /* Enable the DMA clock */
00575   AUDIO_SAIx_DMAx_CLK_ENABLE();
00576
00577   if(hsai->Instance == AUDIO_SAIx)
00578   {
00579     /* Configure the hdma_saiTx handle param
eters */
00580     hdma_saiTx.Init.Channel             = AU
DIO_SAIx_DMAx_CHANNEL;
00581     hdma_saiTx.Init.Direction           = DM
A_MEMORY_TO_PERIPH;
00582     hdma_saiTx.Init.PeriphInc           = DM
A_PINC_DISABLE;
00583     hdma_saiTx.Init.MemInc              = DM
A_MINC_ENABLE;
00584     hdma_saiTx.Init.PeriphDataAlignment = AU
DIO_SAIx_DMAx_PERIPH_DATA_SIZE;
00585     hdma_saiTx.Init.MemDataAlignment    = AU
DIO_SAIx_DMAx_MEM_DATA_SIZE;
00586     hdma_saiTx.Init.Mode                = DM
A_NORMAL;
00587     hdma_saiTx.Init.Priority            = DM
A_PRIORITY_HIGH;
00588     hdma_saiTx.Init.FIFOMode            = DM
A_FIFOMODE_ENABLE;
00589     hdma_saiTx.Init.FIFOThreshold       = DM
A_FIFO_THRESHOLD_FULL;
00590     hdma_saiTx.Init.MemBurst            = DM
A_MBURST_SINGLE;
00591     hdma_saiTx.Init.PeriphBurst         = DM
A_PBURST_SINGLE;
00592
```

```
00593     hdma_saiTx.Instance = AUDIO_SAIx_DMAx_ST
REAM;
00594
00595     /* Associate the DMA handle */
00596     __HAL_LINKDMA(hsai, hdmatx, hdma_saiTx);
00597
00598     /* Deinitialize the Stream for new trans
fer */
00599     HAL_DMA_DeInit(&hdma_saiTx);
00600
00601     /* Configure the DMA Stream */
00602     HAL_DMA_Init(&hdma_saiTx);
00603   }
00604
00605   /* SAI DMA IRQ Channel configuration */
00606   HAL_NVIC_SetPriority(AUDIO_SAIx_DMAx_IRQ,
AUDIO_OUT_IRQ_PREPRIO, 0);
00607   HAL_NVIC_EnableIRQ(AUDIO_SAIx_DMAx_IRQ);
00608 }
00609
00610 /**
00611   * @brief  Initializes the Audio Codec audi
o interface (SAI).
00612   * @param  AudioFreq: Audio frequency to be
 configured for the SAI peripheral.
00613   * @note   The default SlotActive configura
tion is set to CODEC_AUDIOFRAME_SLOT_0123
00614   *         and user can update this configu
ration using
00615   */
00616 static void SAIx_Init(uint32_t AudioFreq)
00617 {
00618   /* Initialize the haudio_out_sai Instance
parameter */
00619   haudio_out_sai.Instance = AUDIO_SAIx;
00620
00621   /* Disable SAI peripheral to allow access
```

```
to SAI internal registers */
00622   __HAL_SAI_DISABLE(&haudio_out_sai);
00623
00624   /* Configure SAI_Block_x
00625   LSBFirst: Disabled
00626   DataSize: 16 */
00627   haudio_out_sai.Init.AudioFrequency = Audio
Freq;
00628   haudio_out_sai.Init.ClockSource = SAI_CLKS
OURCE_PLLI2S;
00629   haudio_out_sai.Init.AudioMode = SAI_MODEMA
STER_TX;
00630   haudio_out_sai.Init.NoDivider = SAI_MASTER
DIVIDER_ENABLED;
00631   haudio_out_sai.Init.Protocol = SAI_FREE_PR
OTOCOL;
00632   haudio_out_sai.Init.DataSize = SAI_DATASIZ
E_16;
00633   haudio_out_sai.Init.FirstBit = SAI_FIRSTBI
T_MSB;
00634   haudio_out_sai.Init.ClockStrobing = SAI_CL
OCKSTROBING_RISINGEDGE;
00635   haudio_out_sai.Init.Synchro = SAI_ASYNCHRO
NOUS;
00636   haudio_out_sai.Init.OutputDrive = SAI_OUTP
UTDRIVE_ENABLED;
00637   haudio_out_sai.Init.FIFOThreshold = SAI_FI
FOTHRESHOLD_1QF;
00638
00639   /* Configure SAI_Block_x Frame
00640   Frame Length: 64
00641   Frame active Length: 32
00642   FS Definition: Start frame + Channel Side
identification
00643   FS Polarity: FS active Low
00644   FS Offset: FS asserted one bit before the
first bit of slot 0 */
```

```
00645   haudio_out_sai.FrameInit.FrameLength = 64;

00646   haudio_out_sai.FrameInit.ActiveFrameLength
 = 32;
00647   haudio_out_sai.FrameInit.FSDefinition = SA
I_FS_CHANNEL_IDENTIFICATION;
00648   haudio_out_sai.FrameInit.FSPolarity = SAI_
FS_ACTIVE_LOW;
00649   haudio_out_sai.FrameInit.FSOffset = SAI_FS
_BEFOREFIRSTBIT;
00650
00651   /* Configure SAI Block_x Slot
00652   Slot First Bit Offset: 0
00653   Slot Size  : 16
00654   Slot Number: 4
00655   Slot Active: All slot actives */
00656   haudio_out_sai.SlotInit.FirstBitOffset = 0
;
00657   haudio_out_sai.SlotInit.SlotSize = SAI_SLO
TSIZE_DATASIZE;
00658   haudio_out_sai.SlotInit.SlotNumber = 4;
00659   haudio_out_sai.SlotInit.SlotActive = CODEC
_AUDIOFRAME_SLOT_0123;
00660   if(HAL_SAI_GetState(&haudio_out_sai) == HA
L_SAI_STATE_RESET)
00661   {
00662     /* Init the SAI */
00663     SAIx_MspInit();
00664   }
00665   HAL_SAI_Init(&haudio_out_sai);
00666
00667   /* Enable SAI peripheral to generate MCLK
*/
00668   __HAL_SAI_ENABLE(&haudio_out_sai);
00669 }
00670
00671 /**
```

```
00672   * @brief  Initializes wave recording.
00673   * @note   This function assumes that the I
2S input clock (through PLL_R in
00674   *         Devices RevA/Z and through dedic
ated PLLI2S_R in Devices RevB/Y)
00675   *         is already configured and ready
to be used.
00676   * @param  AudioFreq: Audio frequency to be
 configured for the I2S peripheral.
00677   * @param  BitRes: Audio frequency to be co
nfigured for the I2S peripheral.
00678   * @param  ChnlNbr: Audio frequency to be c
onfigured for the I2S peripheral.
00679   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00680   */
00681 uint8_t BSP_AUDIO_IN_Init(uint32_t AudioFreq
, uint32_t BitRes, uint32_t ChnlNbr)
00682 {
00683   RCC_PeriphCLKInitTypeDef RCC_ExCLKInitStru
ct;
00684
00685   HAL_RCCEx_GetPeriphCLKConfig(&RCC_ExCLKIni
tStruct);
00686   RCC_ExCLKInitStruct.PeriphClockSelection =
 RCC_PERIPHCLK_I2S;
00687   RCC_ExCLKInitStruct.PLLI2S.PLLI2SN = 384;
00688   RCC_ExCLKInitStruct.PLLI2S.PLLI2SR = 2;
00689   HAL_RCCEx_PeriphCLKConfig(&RCC_ExCLKInitSt
ruct);
00690
00691   /* Configure the PDM library */
00692   PDMDecoder_Init(AudioFreq, ChnlNbr);
00693
00694   /* Configure the Timer which clocks the ME
MS */
00695   TIMx_Init();
```

```
00696
00697   /* Configure the I2S peripheral */
00698   I2Sx_Init(AudioFreq);
00699
00700   /* Return AUDIO_OK when all operations are
 correctly done */
00701   return AUDIO_OK;
00702 }
00703
00704 /**
00705   * @brief  Starts audio recording.
00706   * @param  pbuf: Main buffer pointer for th
e recorded data storing
00707   * @param  size: Current size of the record
ed buffer
00708   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00709   */
00710 uint8_t BSP_AUDIO_IN_Record(uint16_t* pbuf,
uint32_t size)
00711 {
00712   uint32_t ret = AUDIO_ERROR;
00713
00714   /* Start the process receive DMA */
00715   HAL_I2S_Receive_DMA(&haudio_in_i2s, pbuf,
size);
00716
00717   /* Return AUDIO_OK when all operations are
 correctly done */
00718   ret = AUDIO_OK;
00719
00720   return ret;
00721 }
00722
00723 /**
00724   * @brief  Stops audio recording.
00725   * @retval AUDIO_OK if correct communicatio
```

```
n, else wrong communication
00726   */
00727 uint8_t BSP_AUDIO_IN_Stop(void)
00728 {
00729   uint32_t ret = AUDIO_ERROR;
00730
00731   /* Call the Media layer pause function */
00732   HAL_I2S_DMAPause(&haudio_in_i2s);
00733
00734   /* TIMx Peripheral clock disable */
00735   AUDIO_TIMx_CLK_DISABLE();
00736
00737   /* Return AUDIO_OK when all operations are
 correctly done */
00738   ret = AUDIO_OK;
00739
00740   return ret;
00741 }
00742
00743 /**
00744   * @brief  Pauses the audio file stream.
00745   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00746   */
00747 uint8_t BSP_AUDIO_IN_Pause(void)
00748 {
00749   /* Call the Media layer pause function */
00750   HAL_I2S_DMAPause(&haudio_in_i2s);
00751
00752   /* Return AUDIO_OK when all operations are
 correctly done */
00753   return AUDIO_OK;
00754 }
00755
00756 /**
00757   * @brief  Resumes the audio file stream.
```

```
00758   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00759   */
00760 uint8_t BSP_AUDIO_IN_Resume(void)
00761 {
00762   /* Call the Media layer pause/resume funct
ion */
00763   HAL_I2S_DMAResume(&haudio_in_i2s);
00764
00765   /* Return AUDIO_OK when all operations are
 correctly done */
00766   return AUDIO_OK;
00767 }
00768
00769 /**
00770   * @brief  Controls the audio in volume lev
el.
00771   * @param  Volume: Volume level to be set i
n percentage from 0% to 100% (0 for
00772   *         Mute and 100 for Max volume leve
l).
00773   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00774   */
00775 uint8_t BSP_AUDIO_IN_SetVolume(uint8_t Volum
e)
00776 {
00777   /* Set the Global variable AudioInVolume
*/
00778   AudioInVolume = Volume;
00779
00780   /* Return AUDIO_OK when all operations are
 correctly done */
00781   return AUDIO_OK;
00782 }
00783
00784 /**
```

```
00785   * @brief  Converts audio format from PDM t
o PCM.
00786   * @param  PDMBuf: Pointer to data PDM buff
er
00787   * @param  PCMBuf: Pointer to data PCM buff
er
00788   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00789   */
00790 uint8_t BSP_AUDIO_IN_PDMToPCM(uint16_t* PDMB
uf, uint16_t* PCMBuf)
00791 {
00792   uint8_t AppPDM[INTERNAL_BUFF_SIZE*2];
00793   uint8_t byte1 = 0, byte2 = 0;
00794   uint32_t index = 0;
00795
00796   /* PDM Demux */
00797   for(index = 0; index<INTERNAL_BUFF_SIZE/2;
 index++)
00798   {
00799     byte2 = (PDMBuf[index] >> 8)& 0xFF;
00800     byte1 = (PDMBuf[index] & 0xFF);
00801     AppPDM[(index*2)+1] = Channel_Demux[byte
1 & CHANNEL_DEMUX_MASK] | Channel_Demux[byte2 & CH
ANNEL_DEMUX_MASK] << 4;
00802     AppPDM[(index*2)] = Channel_Demux[(byte1
 >> 1) & CHANNEL_DEMUX_MASK] | Channel_Demux[(byte
2 >> 1) & CHANNEL_DEMUX_MASK] << 4;
00803   }
00804
00805   for(index = 0; index < DEFAULT_AUDIO_IN_CH
ANNEL_NBR; index++)
00806   {
00807     /* PDM to PCM filter */
00808     PDM_Filter_64_LSB((uint8_t*)&AppPDM[inde
x], (uint16_t*)&(PCMBuf[index]), AudioInVolume , (
PDMFilter_InitStruct *)&Filter[index]);
```

```
00809   }
00810
00811   /* Return AUDIO_OK when all operations are
 correctly done */
00812   return AUDIO_OK;
00813 }
00814
00815  /**
00816   * @brief  Rx Transfer completed callbacks.
00817   * @param  hi2s: I2S handle
00818   */
00819 void HAL_I2S_RxCpltCallback(I2S_HandleTypeDe
f *hi2s)
00820 {
00821   /* Call the record update function to get
the next buffer to fill and its size (size is igno
red) */
00822   BSP_AUDIO_IN_TransferComplete_CallBack();
00823 }
00824
00825 /**
00826   * @brief  Rx Half Transfer completed callb
acks.
00827   * @param  hi2s: I2S handle
00828   */
00829 void HAL_I2S_RxHalfCpltCallback(I2S_HandleTy
peDef *hi2s)
00830 {
00831   /* Manage the remaining file size and new
address offset: This function
00832      should be coded by user (its prototype
is already declared in stm324x9i_eval_audio.h) */
00833   BSP_AUDIO_IN_HalfTransfer_CallBack();
00834 }
00835
00836 /**
00837   * @brief  I2S error callbacks.
```

```
00838   * @param  hi2s: I2S handle
00839   */
00840 void HAL_I2S_ErrorCallback(I2S_HandleTypeDef
 *hi2s)
00841 {
00842   /* Manage the error generated on DMA FIFO:
 This function
00843      should be coded by user (its prototype
is already declared in stm324x9i_eval_audio.h) */

00844   BSP_AUDIO_IN_Error_Callback();
00845 }
00846
00847 /**
00848   * @brief  User callback when record buffer
 is filled.
00849   */
00850 __weak void BSP_AUDIO_IN_TransferComplete_Ca
llBack(void)
00851 {
00852   /* This function should be implemented by
the user application.
00853      It is called into this driver when the
current buffer is filled
00854      to prepare the next buffer pointer and
its size. */
00855 }
00856
00857 /**
00858   * @brief  Manages the DMA Half Transfer co
mplete event.
00859   */
00860 __weak void BSP_AUDIO_IN_HalfTransfer_CallBa
ck(void)
00861 {
00862   /* This function should be implemented by
the user application.
```

```
00863      It is called into this driver when the
current buffer is filled
00864      to prepare the next buffer pointer and
its size. */
00865 }
00866
00867 /**
00868   * @brief  Audio IN Error callback function.

00869   */
00870 __weak void BSP_AUDIO_IN_Error_Callback(void
)
00871 {
00872   /* This function is called when an Interru
pt due to transfer error on or peripheral
00873      error occurs. */
00874 }
00875
00876 /*******************************************
**********************************
00877                                   Static Functions
00878 *******************************************
**********************************/
00879
00880 /**
00881   * @brief  Initializes the PDM library.
00882   * @param  AudioFreq: Audio sampling freque
ncy
00883   * @param  ChnlNbr: Number of audio channel
s (1: mono; 2: stereo)
00884   */
00885 static void PDMDecoder_Init(uint32_t AudioFr
eq, uint32_t ChnlNbr)
00886 {
00887   uint32_t i = 0;
00888
00889   /* Enable CRC peripheral to unlock the PDM
```

```
 library */
00890   __CRC_CLK_ENABLE();
00891
00892   for(i = 0; i < ChnlNbr; i++)
00893   {
00894     /* Filter LP & HP Init */
00895     Filter[i].LP_HZ = AudioFreq/2;
00896     Filter[i].HP_HZ = 10;
00897     Filter[i].Fs = AudioFreq;
00898     Filter[i].Out_MicChannels = ChnlNbr;
00899     Filter[i].In_MicChannels = ChnlNbr;
00900     PDM_Filter_Init((PDMFilter_InitStruct *)&
Filter[i]);
00901   }
00902 }
00903
00904 /**
00905   * @brief  AUDIO IN I2S MSP Init.
00906   */
00907 static void I2Sx_MspInit(void)
00908 {
00909   static DMA_HandleTypeDef hdma_i2sRx;
00910   GPIO_InitTypeDef  GPIO_InitStruct;
00911   I2S_HandleTypeDef *hi2s = &haudio_in_i2s;
00912
00913   /* Enable I2S clock */
00914   AUDIO_I2Sx_CLK_ENABLE();
00915
00916   /* Enable SCK and SD GPIO clock */
00917   AUDIO_I2Sx_SD_GPIO_CLK_ENABLE();
00918   AUDIO_I2Sx_SCK_GPIO_CLK_ENABLE();
00919   /* CODEC_I2S pins configuration: WS, SCK a
nd SD pins */
00920   GPIO_InitStruct.Pin = AUDIO_I2Sx_SCK_PIN;
00921   GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00922   GPIO_InitStruct.Pull = GPIO_NOPULL;
00923   GPIO_InitStruct.Speed = GPIO_SPEED_FAST;
```

```
00924   GPIO_InitStruct.Alternate = AUDIO_I2Sx_SCK
_AF;
00925   HAL_GPIO_Init(AUDIO_I2Sx_SCK_GPIO_PORT, &G
PIO_InitStruct);
00926
00927   GPIO_InitStruct.Pin = AUDIO_I2Sx_SD_PIN;
00928   GPIO_InitStruct.Alternate = AUDIO_I2Sx_SD_
AF;
00929   HAL_GPIO_Init(AUDIO_I2Sx_SD_GPIO_PORT, &GP
IO_InitStruct);
00930
00931   /* Enable the DMA clock */
00932   AUDIO_I2Sx_DMAx_CLK_ENABLE();
00933
00934   if(hi2s->Instance == AUDIO_I2Sx)
00935   {
00936     /* Configure the hdma_i2sRx handle param
eters */
00937     hdma_i2sRx.Init.Channel             = AU
DIO_I2Sx_DMAx_CHANNEL;
00938     hdma_i2sRx.Init.Direction           = DM
A_PERIPH_TO_MEMORY;
00939     hdma_i2sRx.Init.PeriphInc           = DM
A_PINC_DISABLE;
00940     hdma_i2sRx.Init.MemInc              = DM
A_MINC_ENABLE;
00941     hdma_i2sRx.Init.PeriphDataAlignment = AU
DIO_I2Sx_DMAx_PERIPH_DATA_SIZE;
00942     hdma_i2sRx.Init.MemDataAlignment    = AU
DIO_I2Sx_DMAx_MEM_DATA_SIZE;
00943     hdma_i2sRx.Init.Mode                = DM
A_CIRCULAR;
00944     hdma_i2sRx.Init.Priority            = DM
A_PRIORITY_HIGH;
00945     hdma_i2sRx.Init.FIFOMode            = DM
A_FIFOMODE_DISABLE;
00946     hdma_i2sRx.Init.FIFOThreshold       = DM
```

```
A_FIFO_THRESHOLD_FULL;
00947     hdma_i2sRx.Init.MemBurst            = DM
A_MBURST_SINGLE;
00948     hdma_i2sRx.Init.PeriphBurst         = DM
A_MBURST_SINGLE;
00949
00950     hdma_i2sRx.Instance = AUDIO_I2Sx_DMAx_ST
REAM;
00951
00952     /* Associate the DMA handle */
00953     __HAL_LINKDMA(hi2s, hdmarx, hdma_i2sRx);
00954
00955     /* Deinitialize the Stream for new trans
fer */
00956     HAL_DMA_DeInit(&hdma_i2sRx);
00957
00958     /* Configure the DMA Stream */
00959     HAL_DMA_Init(&hdma_i2sRx);
00960   }
00961
00962   /* I2S DMA IRQ Channel configuration */
00963   HAL_NVIC_SetPriority(AUDIO_I2Sx_DMAx_IRQ,
AUDIO_IN_IRQ_PREPRIO, 0);
00964   HAL_NVIC_EnableIRQ(AUDIO_I2Sx_DMAx_IRQ);
00965 }
00966
00967 /**
00968   * @brief  Initializes the Audio Codec audi
o interface (I2S)
00969   * @note   This function assumes that the I
2S input clock (through PLL_R in
00970   *         Devices RevA/Z and through dedic
ated PLLI2S_R in Devices RevB/Y)
00971   *         is already configured and ready
to be used.
00972   * @param  AudioFreq: Audio frequency to be
 configured for the I2S peripheral.
```

```
00973   */
00974 static void I2Sx_Init(uint32_t AudioFreq)
00975 {
00976   /* Initialize the haudio_in_i2s Instance p
arameter */
00977   haudio_in_i2s.Instance = AUDIO_I2Sx;
00978
00979  /* Disable I2S block */
00980   __HAL_I2S_DISABLE(&haudio_in_i2s);
00981
00982   /* I2S2 peripheral configuration */
00983   haudio_in_i2s.Init.AudioFreq = 4 * AudioFr
eq;
00984   haudio_in_i2s.Init.ClockSource = I2S_CLOCK
_PLL;
00985   haudio_in_i2s.Init.CPOL = I2S_CPOL_HIGH;
00986   haudio_in_i2s.Init.DataFormat = I2S_DATAFO
RMAT_16B;
00987   haudio_in_i2s.Init.MCLKOutput = I2S_MCLKOU
TPUT_DISABLE;
00988   haudio_in_i2s.Init.Mode = I2S_MODE_MASTER_
RX;
00989   haudio_in_i2s.Init.Standard = I2S_STANDARD
_LSB;
00990   if(HAL_I2S_GetState(&haudio_in_i2s) == HAL
_I2S_STATE_RESET)
00991   {
00992     /* Initialize the I2S peripheral with th
e structure above */
00993     I2Sx_MspInit();
00994   }
00995
00996   /* Init the I2S */
00997   HAL_I2S_Init(&haudio_in_i2s);
00998 }
00999
01000 /**
```

```
01001   * @brief  Initializes the TIM INput Captur
e MSP.
01002   * @param  htim: TIM handle
01003   */
01004 static void TIMx_IC_MspInit(TIM_HandleTypeDe
f *htim)
01005 {
01006   GPIO_InitTypeDef   GPIO_InitStruct;
01007
01008   /* Enable peripherals and GPIO Clocks ----
-----------------------------------*/
01009   /* TIMx Peripheral clock enable */
01010   AUDIO_TIMx_CLK_ENABLE();
01011
01012   /* Enable GPIO Channels Clock */
01013   AUDIO_TIMx_GPIO_CLK_ENABLE();
01014
01015   /* Configure I/Os ------------------------
-----------------------------------*/
01016   /* Common configuration for all channels */

01017   GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
01018   GPIO_InitStruct.Pull = GPIO_NOPULL;
01019   GPIO_InitStruct.Speed = GPIO_SPEED_HIGH;
01020   GPIO_InitStruct.Alternate = AUDIO_TIMx_AF;
01021
01022   /* Configure TIM input channel */
01023   GPIO_InitStruct.Pin = AUDIO_TIMx_IN_GPIO_P
IN;
01024   HAL_GPIO_Init(AUDIO_TIMx_GPIO, &GPIO_InitS
truct);
01025
01026   /* Configure TIM output channel */
01027   GPIO_InitStruct.Pin = AUDIO_TIMx_OUT_GPIO_
PIN;
01028   HAL_GPIO_Init(AUDIO_TIMx_GPIO, &GPIO_InitS
truct);
```

```
01029 }
01030
01031 /**
01032   * @brief  Configure TIM as a clock divider
 by 2.
01033   *         I2S_SCK is externally connected
to TIMx input channel
01034   */
01035 static void TIMx_Init(void)
01036 {
01037   TIM_IC_InitTypeDef     sICConfig;
01038   TIM_OC_InitTypeDef     sOCConfig;
01039   TIM_ClockConfigTypeDef sCLKSourceConfig;
01040   TIM_SlaveConfigTypeDef sSlaveConfig;
01041
01042   /* Configure the TIM peripheral ----------
----------------------------------*/
01043   /* Set TIMx instance */
01044   haudio_tim.Instance = AUDIO_TIMx;
01045   /* Timer Input Capture Configuration Struc
ture declaration */
01046    /* Initialize TIMx peripheral as follow:
01047        + Period = 0xFFFF
01048        + Prescaler = 0
01049        + ClockDivision = 0
01050        + Counter direction = Up
01051   */
01052   haudio_tim.Init.Period        = 1;
01053   haudio_tim.Init.Prescaler     = 0;
01054   haudio_tim.Init.ClockDivision = 0;
01055   haudio_tim.Init.CounterMode   = TIM_COUNTE
RMODE_UP;
01056
01057   /* Initialize the TIMx peripheral with the
 structure above */
01058   TIMx_IC_MspInit(&haudio_tim);
01059   HAL_TIM_IC_Init(&haudio_tim);
```

```
01060
01061   /* Configure the Input Capture channel ---
-----------------------------------*/
01062   /* Configure the Input Capture of channel
2 */
01063   sICConfig.ICPolarity  = TIM_ICPOLARITY_FAL
LING;
01064   sICConfig.ICSelection = TIM_ICSELECTION_DI
RECTTI;
01065   sICConfig.ICPrescaler = TIM_ICPSC_DIV1;
01066   sICConfig.ICFilter    = 0;
01067   HAL_TIM_IC_ConfigChannel(&haudio_tim, &sIC
Config, AUDIO_TIMx_IN_CHANNEL);
01068
01069   /* Select external clock mode 1 */
01070   sCLKSourceConfig.ClockSource = TIM_CLOCKSO
URCE_ETRMODE1;
01071   sCLKSourceConfig.ClockPolarity = TIM_CLOCK
POLARITY_NONINVERTED;
01072   sCLKSourceConfig.ClockPrescaler = TIM_CLOC
KPRESCALER_DIV1;
01073   sCLKSourceConfig.ClockFilter = 0;
01074   HAL_TIM_ConfigClockSource(&haudio_tim, &sC
LKSourceConfig);
01075
01076   /* Select Input Channel as input trigger */

01077   sSlaveConfig.InputTrigger = TIM_TS_TI1FP1;
01078   sSlaveConfig.SlaveMode = TIM_SLAVEMODE_EXT
ERNAL1;
01079   sSlaveConfig.TriggerPolarity = TIM_TRIGGER
POLARITY_NONINVERTED;
01080   sSlaveConfig.TriggerPrescaler = TIM_CLOCKP
RESCALER_DIV1;
01081   sSlaveConfig.TriggerFilter = 0;
01082   HAL_TIM_SlaveConfigSynchronization(&haudio
_tim, &sSlaveConfig);
```

```
01083
01084   /* Output Compare PWM Mode configuration:
Channel2 */
01085   sOCConfig.OCMode = TIM_OCMODE_PWM1;
01086   sOCConfig.OCIdleState = TIM_OCIDLESTATE_SE
T;
01087   sOCConfig.Pulse = 1;
01088   sOCConfig.OCPolarity = TIM_OCPOLARITY_HIGH
;
01089   sOCConfig.OCNPolarity = TIM_OCNPOLARITY_HI
GH;
01090   sOCConfig.OCFastMode = TIM_OCFAST_DISABLE;
01091   sOCConfig.OCNIdleState = TIM_OCNIDLESTATE_
SET;
01092
01093   /* Initialize the TIM3 Channel2 with the s
tructure above */
01094   HAL_TIM_PWM_ConfigChannel(&haudio_tim, &sO
CConfig, AUDIO_TIMx_OUT_CHANNEL);
01095
01096   /* Start the TIM3 Channel2 */
01097   HAL_TIM_PWM_Start(&haudio_tim, AUDIO_TIMx_
OUT_CHANNEL);
01098
01099   /* Start the TIM3 Channel1 */
01100   HAL_TIM_IC_Start(&haudio_tim, AUDIO_TIMx_I
N_CHANNEL);
01101 }
01102
01103 /**
01104   * @}
01105   */
01106
01107 /**
01108   * @}
01109   */
01110
```

```
01111 /**
01112   * @}
01113   */
01114
01115 /**
01116   * @}
01117   */
01118
01119 /**
01120   * @}
01121   */
01122
01123 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_camera.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval_camera.h
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm324x9i_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 __STM324x9I_EVAL_CAMERA_H
00041 #define __STM324x9I_EVAL_CAMERA_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------
-----------------------------------*/
00048 /* Include Camera component Driver */
00049 #include "../Components/ov2640/ov2640.h"
00050
00051 /* Include IO Driver */
00052 #include "stm324x9i_eval_io.h"
00053
00054 /** @addtogroup BSP
00055   * @{
00056   */
00057
00058 /** @addtogroup STM324x9I_EVAL
00059   * @{
00060   */
00061
00062 /** @addtogroup STM324x9I_EVAL_CAMERA
00063   * @{
00064   */
00065
00066 /** @defgroup STM324x9I_EVAL_CAMERA_Exported
_Types STM324x9I 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 STM324x9I_EVAL_CAMERA_Exported
_Constants STM324x9I EVAL CAMERA Exported Constants

00089   * @{
00090   */
00091 /**
00092   * @}
00093   */
00094
00095 /** @defgroup STM324x9I_EVAL_CAMERA_Exported
_Functions STM324x9I EVAL CAMERA Exported Functions
```

```
00096   * @{
00097   */
00098 uint8_t BSP_CAMERA_Init(uint32_t Resolution)
;
00099 void    BSP_CAMERA_ContinuousStart(uint8_t *
buff);
00100 void    BSP_CAMERA_SnapshotStart(uint8_t *bu
ff);
00101 void    BSP_CAMERA_Suspend(void);
00102 void    BSP_CAMERA_Resume(void);
00103 uint8_t BSP_CAMERA_Stop(void);
00104 void    BSP_CAMERA_LineEventCallback(void);
00105 void    BSP_CAMERA_VsyncEventCallback(void);
00106 void    BSP_CAMERA_FrameEventCallback(void);
00107 void    BSP_CAMERA_ErrorCallback(void);
00108
00109 /* Camera features functions prototype */
00110 void    BSP_CAMERA_ContrastBrightnessConfig(
uint32_t contrast_level, uint32_t brightness_level
);
00111 void    BSP_CAMERA_BlackWhiteConfig(uint32_t
 Mode);
00112 void    BSP_CAMERA_ColorEffectConfig(uint32_
t Effect);
00113
00114 /* To be called in DCMI_IRQHandler function
*/
00115 void    BSP_CAMERA_IRQHandler(void);
00116 /* To be called in DMA2_Stream1_IRQHandler f
unction */
00117 void    BSP_CAMERA_DMA_IRQHandler(void);
00118
00119 /**
00120   * @}
00121   */
00122
00123 /**
```

```
00124   * @}
00125   */
00126
00127 /**
00128   * @}
00129   */
00130
00131 /**
00132   * @}
00133   */
00134
00135 #ifdef __cplusplus
00136 }
00137 #endif
00138
00139 #endif /* __STM324x9I_EVAL_CAMERA_H */
00140
00141 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_camera.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval_camera.c
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file includes the driver for Camera modules mounted on
00008   *          STM324x9I-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 OV2640 component driver MUST be inc
luded 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 "stm324x9i_eval_camera.h"
00068
00069 /** @addtogroup BSP
00070   * @{
00071   */
00072
00073 /** @addtogroup STM324x9I_EVAL
00074   * @{
00075   */
00076
00077 /** @defgroup STM324x9I_EVAL_CAMERA STM324x9
I EVAL CAMERA
00078   * @{
00079   */
00080
00081 /** @defgroup STM324x9I_EVAL_CAMERA_Private_
TypesDefinitions STM324x9I EVAL CAMERA Private Typ
esDefinitions
00082   * @{
00083   */
00084 /**
00085   * @}
00086   */
00087
00088 /** @defgroup STM324x9I_EVAL_CAMERA_Private_
Defines STM324x9I EVAL CAMERA Private Defines
00089   * @{
00090   */
00091 /**
```

```
00092   * @}
00093   */
00094
00095 /** @defgroup STM324x9I_EVAL_CAMERA_Private_
Macros STM324x9I EVAL CAMERA Private Macros
00096   * @{
00097   */
00098 /**
00099   * @}
00100   */
00101
00102 /** @defgroup STM324x9I_EVAL_CAMERA_Private_
Variables STM324x9I EVAL CAMERA Private Variables
00103   * @{
00104   */
00105 static DCMI_HandleTypeDef  hdcmi_eval;
00106        CAMERA_DrvTypeDef   *camera_drv;
00107 uint32_t current_resolution;
00108 /**
00109   * @}
00110   */
00111
00112 /** @defgroup STM324x9I_EVAL_CAMERA_Private_
FunctionPrototypes STM324x9I EVAL CAMERA Private F
unctionPrototypes
00113   * @{
00114   */
00115 static void DCMI_MspInit(void);
00116 static uint32_t GetSize(uint32_t resolution);

00117 /**
00118   * @}
00119   */
00120
00121 /** @defgroup STM324x9I_EVAL_CAMERA_Private_
Functions STM324x9I EVAL CAMERA Private Functions
00122   * @{
```

```
00123   */
00124
00125 /**
00126   * @brief  Initializes the camera.
00127   * @param  Resolution: Camera Resolution
00128   * @retval Camera status
00129   */
00130 uint8_t BSP_CAMERA_Init(uint32_t Resolution)
00131 {
00132   DCMI_HandleTypeDef *phdcmi;
00133
00134   uint8_t ret = CAMERA_ERROR;
00135
00136   /* Get the DCMI handle structure */
00137   phdcmi = &hdcmi_eval;
00138
00139   /*** Configures the DCMI to interface with
 the camera module ***/
00140   /* DCMI configuration */
00141   phdcmi->Init.CaptureRate      = DCMI_CR_AL
L_FRAME;
00142   phdcmi->Init.HSPolarity       = DCMI_HSPOL
ARITY_LOW;
00143   phdcmi->Init.SynchroMode      = DCMI_SYNCH
RO_HARDWARE;
00144   phdcmi->Init.VSPolarity       = DCMI_VSPOL
ARITY_LOW;
00145   phdcmi->Init.ExtendedDataMode = DCMI_EXTEN
D_DATA_8B;
00146   phdcmi->Init.PCKPolarity      = DCMI_PCKPO
LARITY_RISING;
00147   phdcmi->Instance              = DCMI;
00148
00149   /* Configure IO functionalities for camera
 detect pin */
00150   BSP_IO_Init();
00151
```

```
00152   /* Set the camera STANDBY pin */
00153   BSP_IO_ConfigPin(XSDN_PIN, IO_MODE_OUTPUT)
;
00154   BSP_IO_WritePin(XSDN_PIN, SET);
00155
00156   /* Check if the camera is plugged */
00157   if(BSP_IO_ReadPin(CAM_PLUG_PIN))
00158   {
00159     return CAMERA_ERROR;
00160   }
00161
00162   /* DCMI Initialization */
00163   DCMI_MspInit();
00164   HAL_DCMI_Init(phdcmi);
00165
00166   if(ov2640_ReadID(CAMERA_I2C_ADDRESS) == OV
2640_ID)
00167   {
00168     /* Initialize the camera driver structur
e */
00169     camera_drv = &ov2640_drv;
00170
00171     /* Camera Init */
00172     camera_drv->Init(CAMERA_I2C_ADDRESS, Res
olution);
00173
00174     /* Return CAMERA_OK status */
00175     ret = CAMERA_OK;
00176   }
00177
00178   current_resolution = Resolution;
00179
00180   return ret;
00181 }
00182
00183 /**
00184   * @brief  Starts the camera capture in con
```

```
tinuous mode.
00185   * @param  buff: pointer to the camera outp
ut buffer
00186   */
00187 void BSP_CAMERA_ContinuousStart(uint8_t *buf
f)
00188 {
00189   /* Start the camera capture */
00190   HAL_DCMI_Start_DMA(&hdcmi_eval, DCMI_MODE_
CONTINUOUS, (uint32_t)buff, GetSize(current_resolu
tion));
00191 }
00192
00193 /**
00194   * @brief  Starts the camera capture in sna
pshot mode.
00195   * @param  buff: pointer to the camera outp
ut buffer
00196   */
00197 void BSP_CAMERA_SnapshotStart(uint8_t *buff)
00198 {
00199   /* Start the camera capture */
00200   HAL_DCMI_Start_DMA(&hdcmi_eval, DCMI_MODE_
SNAPSHOT, (uint32_t)buff, GetSize(current_resoluti
on));
00201 }
00202
00203 /**
00204   * @brief Suspend the CAMERA capture
00205   */
00206 void BSP_CAMERA_Suspend(void)
00207 {
00208   /* Disable the DMA */
00209   __HAL_DMA_DISABLE(hdcmi_eval.DMA_Handle);
00210   /* Disable the DCMI */
00211   __HAL_DCMI_DISABLE(&hdcmi_eval);
00212
```

```
00213 }
00214
00215 /**
00216   * @brief Resume the CAMERA capture
00217   */
00218 void BSP_CAMERA_Resume(void)
00219 {
00220   /* Enable the DCMI */
00221   __HAL_DCMI_ENABLE(&hdcmi_eval);
00222   /* Enable the DMA */
00223   __HAL_DMA_ENABLE(hdcmi_eval.DMA_Handle);
00224 }
00225
00226 /**
00227   * @brief  Stop the CAMERA capture
00228   * @retval Camera status
00229   */
00230 uint8_t BSP_CAMERA_Stop(void)
00231 {
00232   DCMI_HandleTypeDef *phdcmi;
00233
00234   uint8_t ret = CAMERA_ERROR;
00235
00236   /* Get the DCMI handle structure */
00237   phdcmi = &hdcmi_eval;
00238
00239   if(HAL_DCMI_Stop(phdcmi) == HAL_OK)
00240   {
00241     ret = CAMERA_OK;
00242   }
00243
00244   /* Initialize IO */
00245   BSP_IO_Init();
00246
00247   /* Reset the camera STANDBY pin */
00248   BSP_IO_ConfigPin(XSDN_PIN, IO_MODE_OUTPUT)
;
```

```
00249   BSP_IO_WritePin(XSDN_PIN, RESET);
00250
00251   return ret;
00252 }
00253
00254 /**
00255   * @brief  Configures the camera contrast a
nd brightness.
00256   * @param  contrast_level: Contrast level
00257   *          This parameter can be one of th
e following values:
00258   *            @arg  CAMERA_CONTRAST_LEVEL4:
 for contrast +2
00259   *            @arg  CAMERA_CONTRAST_LEVEL3:
 for contrast +1
00260   *            @arg  CAMERA_CONTRAST_LEVEL2:
 for contrast  0
00261   *            @arg  CAMERA_CONTRAST_LEVEL1:
 for contrast -1
00262   *            @arg  CAMERA_CONTRAST_LEVEL0:
 for contrast -2
00263   * @param  brightness_level: Contrast level
00264   *          This parameter can be one of th
e following values:
00265   *            @arg  CAMERA_BRIGHTNESS_LEVEL
4: for brightness +2
00266   *            @arg  CAMERA_BRIGHTNESS_LEVEL
3: for brightness +1
00267   *            @arg  CAMERA_BRIGHTNESS_LEVEL
2: for brightness  0
00268   *            @arg  CAMERA_BRIGHTNESS_LEVEL
1: for brightness -1
00269   *            @arg  CAMERA_BRIGHTNESS_LEVEL
0: for brightness -2
00270   */
00271 void BSP_CAMERA_ContrastBrightnessConfig(uin
t32_t contrast_level, uint32_t brightness_level)
```

```
00272 {
00273   if(camera_drv->Config != NULL)
00274   {
00275     camera_drv->Config(CAMERA_I2C_ADDRESS, C
AMERA_CONTRAST_BRIGHTNESS, contrast_level, brightn
ess_level);
00276   }
00277 }
00278
00279 /**
00280   * @brief  Configures the camera white bala
nce.
00281   * @param  Mode: black_white mode
00282   *           This parameter can be one of th
e following values:
00283   *             @arg  CAMERA_BLACK_WHITE_BW
00284   *             @arg  CAMERA_BLACK_WHITE_NEGA
TIVE
00285   *             @arg  CAMERA_BLACK_WHITE_BW_N
EGATIVE
00286   *             @arg  CAMERA_BLACK_WHITE_NORM
AL
00287   */
00288 void BSP_CAMERA_BlackWhiteConfig(uint32_t Mo
de)
00289 {
00290   if(camera_drv->Config != NULL)
00291   {
00292     camera_drv->Config(CAMERA_I2C_ADDRESS, C
AMERA_BLACK_WHITE, Mode, 0);
00293   }
00294 }
00295
00296 /**
00297   * @brief  Configures the camera color effe
ct.
00298   * @param  Effect: Color effect
```

```
00299   *           This parameter can be one of th
e following values:
00300   *             @arg  CAMERA_COLOR_EFFECT_ANT
IQUE
00301   *             @arg  CAMERA_COLOR_EFFECT_BLU
E
00302   *             @arg  CAMERA_COLOR_EFFECT_GRE
EN
00303   *             @arg  CAMERA_COLOR_EFFECT_RED

00304   */
00305 void BSP_CAMERA_ColorEffectConfig(uint32_t E
ffect)
00306 {
00307   if(camera_drv->Config != NULL)
00308   {
00309     camera_drv->Config(CAMERA_I2C_ADDRESS, C
AMERA_COLOR_EFFECT, Effect, 0);
00310   }
00311 }
00312
00313 /**
00314   * @brief  Handles DCMI interrupt request.
00315   */
00316 void BSP_CAMERA_IRQHandler(void)
00317 {
00318   HAL_DCMI_IRQHandler(&hdcmi_eval);
00319 }
00320
00321 /**
00322   * @brief  Handles DMA interrupt request.
00323   */
00324 void BSP_CAMERA_DMA_IRQHandler(void)
00325 {
00326   HAL_DMA_IRQHandler(hdcmi_eval.DMA_Handle);
00327 }
00328
```

```
00329 /**
00330   * @brief  Get the capture size.
00331   * @param  resolution: the current resoluti
on.
00332   * @retval capture size.
00333   */
00334 static uint32_t GetSize(uint32_t resolution)
00335 {
00336   uint32_t size = 0;
00337 
00338   /* Get capture size */
00339   switch (resolution)
00340   {
00341   case CAMERA_R160x120:
00342     {
00343       size =  0x2580;
00344     }
00345     break;
00346   case CAMERA_R320x240:
00347     {
00348       size =  0x9600;
00349     }
00350     break;
00351   case CAMERA_R480x272:
00352     {
00353       size =  0xFF00;
00354     }
00355     break;
00356   case CAMERA_R640x480:
00357     {
00358       size =  0x25800;
00359     }
00360     break;
00361   default:
00362     {
00363       break;
00364     }
```

```
00365   }
00366
00367   return size;
00368 }
00369
00370 /**
00371   * @brief  Initializes the DCMI MSP.
00372   */
00373 static void DCMI_MspInit(void)
00374 {
00375   static DMA_HandleTypeDef hdma_eval;
00376   GPIO_InitTypeDef GPIO_Init_Structure;
00377   DCMI_HandleTypeDef *hdcmi = &hdcmi_eval;
00378
00379   /*** Enable peripherals and GPIO clocks **
*/
00380   /* Enable DCMI clock */
00381   __DCMI_CLK_ENABLE();
00382
00383   /* Enable DMA2 clock */
00384   __DMA2_CLK_ENABLE();
00385
00386   /* Enable GPIO clocks */
00387   __GPIOA_CLK_ENABLE();
00388   __GPIOB_CLK_ENABLE();
00389   __GPIOC_CLK_ENABLE();
00390   __GPIOD_CLK_ENABLE();
00391   __GPIOE_CLK_ENABLE();
00392
00393   /*** Configure the GPIO ***/
00394   /* Configure DCMI GPIO as alternate functi
on */
00395   GPIO_Init_Structure.Pin       = GPIO_PIN_4
;
00396   GPIO_Init_Structure.Mode      = GPIO_MODE_
AF_PP;
00397   GPIO_Init_Structure.Pull      = GPIO_PULLU
```

```
P;
00398   GPIO_Init_Structure.Speed     = GPIO_SPEED
_HIGH;
00399   GPIO_Init_Structure.Alternate = GPIO_AF13_
DCMI;
00400   HAL_GPIO_Init(GPIOA, &GPIO_Init_Structure)
;
00401
00402   GPIO_Init_Structure.Pin       = GPIO_PIN_7
 | GPIO_PIN_8;
00403   GPIO_Init_Structure.Mode      = GPIO_MODE_
AF_PP;
00404   GPIO_Init_Structure.Pull      = GPIO_PULLU
P;
00405   GPIO_Init_Structure.Speed     = GPIO_SPEED
_HIGH;
00406   GPIO_Init_Structure.Alternate = GPIO_AF13_
DCMI;
00407   HAL_GPIO_Init(GPIOB, &GPIO_Init_Structure)
;
00408
00409   GPIO_Init_Structure.Pin       = GPIO_PIN_6
 | GPIO_PIN_7  | GPIO_PIN_8  |\
00410                                   GPIO_PIN_9
 | GPIO_PIN_10 | GPIO_PIN_11 |\
00411                                   GPIO_PIN_1
2;
00412   GPIO_Init_Structure.Mode      = GPIO_MODE_
AF_PP;
00413   GPIO_Init_Structure.Pull      = GPIO_PULLU
P;
00414   GPIO_Init_Structure.Speed     = GPIO_SPEED
_HIGH;
00415   GPIO_Init_Structure.Alternate = GPIO_AF13_
DCMI;
00416   HAL_GPIO_Init(GPIOC, &GPIO_Init_Structure)
;
```

```
00417
00418   GPIO_Init_Structure.Pin       = GPIO_PIN_2
 | GPIO_PIN_3 | GPIO_PIN_6;
00419   GPIO_Init_Structure.Mode      = GPIO_MODE_
AF_PP;
00420   GPIO_Init_Structure.Pull      = GPIO_PULLU
P;
00421   GPIO_Init_Structure.Speed     = GPIO_SPEED
_HIGH;
00422   GPIO_Init_Structure.Alternate = GPIO_AF13_
DCMI;
00423   HAL_GPIO_Init(GPIOD, &GPIO_Init_Structure)
;
00424
00425   GPIO_Init_Structure.Pin       = GPIO_PIN_6
;
00426   GPIO_Init_Structure.Mode      = GPIO_MODE_
AF_PP;
00427   GPIO_Init_Structure.Pull      = GPIO_PULLU
P;
00428   GPIO_Init_Structure.Speed     = GPIO_SPEED
_HIGH;
00429   GPIO_Init_Structure.Alternate = GPIO_AF13_
DCMI;
00430   HAL_GPIO_Init(GPIOE, &GPIO_Init_Structure)
;
00431
00432   GPIO_Init_Structure.Pin       = GPIO_PIN_6
;
00433   GPIO_Init_Structure.Mode      = GPIO_MODE_
AF_PP;
00434   GPIO_Init_Structure.Pull      = GPIO_PULLU
P;
00435   GPIO_Init_Structure.Speed     = GPIO_SPEED
_HIGH;
00436   GPIO_Init_Structure.Alternate = GPIO_AF13_
DCMI;
```

```
00437   HAL_GPIO_Init(GPIOA, &GPIO_Init_Structure)
;
00438
00439   /*** Configure the DMA ***/
00440   /* Set the parameters to be configured */
00441   hdma_eval.Init.Channel             = DMA_C
HANNEL_1;
00442   hdma_eval.Init.Direction           = DMA_P
ERIPH_TO_MEMORY;
00443   hdma_eval.Init.PeriphInc           = DMA_P
INC_DISABLE;
00444   hdma_eval.Init.MemInc              = DMA_M
INC_ENABLE;
00445   hdma_eval.Init.PeriphDataAlignment = DMA_P
DATAALIGN_WORD;
00446   hdma_eval.Init.MemDataAlignment    = DMA_M
DATAALIGN_WORD;
00447   hdma_eval.Init.Mode                = DMA_C
IRCULAR;
00448   hdma_eval.Init.Priority            = DMA_P
RIORITY_HIGH;
00449   hdma_eval.Init.FIFOMode            = DMA_F
IFOMODE_DISABLE;
00450   hdma_eval.Init.FIFOThreshold       = DMA_F
IFO_THRESHOLD_FULL;
00451   hdma_eval.Init.MemBurst            = DMA_M
BURST_SINGLE;
00452   hdma_eval.Init.PeriphBurst         = DMA_P
BURST_SINGLE;
00453
00454   hdma_eval.Instance = DMA2_Stream1;
00455
00456   /* Associate the initialized DMA handle to
 the DCMI handle */
00457   __HAL_LINKDMA(hdcmi, DMA_Handle, hdma_eval
);
00458
```

```
00459   /*** Configure the NVIC for DCMI and DMA *
**/
00460   /* NVIC configuration for DCMI transfer co
mplete interrupt */
00461   HAL_NVIC_SetPriority(DCMI_IRQn, 5, 0);
00462   HAL_NVIC_EnableIRQ(DCMI_IRQn);
00463
00464   /* NVIC configuration for DMA2D transfer c
omplete interrupt */
00465   HAL_NVIC_SetPriority(DMA2_Stream1_IRQn, 5,
 0);
00466   HAL_NVIC_EnableIRQ(DMA2_Stream1_IRQn);
00467
00468   /* Configure the DMA stream */
00469   HAL_DMA_Init(hdcmi->DMA_Handle);
00470 }
00471
00472 /**
00473   * @brief  Line event callback
00474   * @param  hdcmi: pointer to the DCMI handl
e
00475   */
00476 void HAL_DCMI_LineEventCallback(DCMI_HandleT
ypeDef *hdcmi)
00477 {
00478   BSP_CAMERA_LineEventCallback();
00479 }
00480
00481 /**
00482   * @brief  Line Event callback.
00483   */
00484 __weak void BSP_CAMERA_LineEventCallback(void
)
00485 {
00486   /* NOTE : This function Should not be modi
fied, when the callback is needed,
00487             the HAL_DCMI_LineEventCallback c
```

```
ould be implemented in the user file
00488    */
00489 }
00490
00491 /**
00492   * @brief  VSYNC event callback
00493   * @param  hdcmi: pointer to the DCMI handl
e
00494   */
00495 void HAL_DCMI_VsyncEventCallback(DCMI_Handle
TypeDef *hdcmi)
00496 {
00497   BSP_CAMERA_VsyncEventCallback();
00498 }
00499
00500 /**
00501   * @brief  VSYNC Event callback.
00502   */
00503 __weak void BSP_CAMERA_VsyncEventCallback(vo
id)
00504 {
00505   /* NOTE : This function Should not be modi
fied, when the callback is needed,
00506             the HAL_DCMI_VsyncEventCallback
could be implemented in the user file
00507    */
00508 }
00509
00510 /**
00511   * @brief  Frame event callback
00512   * @param  hdcmi: pointer to the DCMI handl
e
00513   */
00514 void HAL_DCMI_FrameEventCallback(DCMI_Handle
TypeDef *hdcmi)
00515 {
00516   BSP_CAMERA_FrameEventCallback();
```

```
00517 }
00518
00519 /**
00520   * @brief  Frame Event callback.
00521   */
00522 __weak void BSP_CAMERA_FrameEventCallback(vo
id)
00523 {
00524   /* NOTE : This function Should not be modi
fied, when the callback is needed,
00525             the HAL_DCMI_FrameEventCallback
could be implemented in the user file
00526    */
00527 }
00528
00529 /**
00530   * @brief  Error callback
00531   * @param  hdcmi: pointer to the DCMI handl
e
00532   */
00533 void HAL_DCMI_ErrorCallback(DCMI_HandleTypeD
ef *hdcmi)
00534 {
00535   BSP_CAMERA_ErrorCallback();
00536 }
00537
00538 /**
00539   * @brief  Error callback.
00540   */
00541 __weak void BSP_CAMERA_ErrorCallback(void)
00542 {
00543   /* NOTE : This function Should not be modi
fied, when the callback is needed,
00544             the HAL_DCMI_ErrorCallback could
 be implemented in the user file
00545    */
00546 }
```

```
00547
00548 /**
00549   * @}
00550   */
00551
00552 /**
00553   * @}
00554   */
00555
00556 /**
00557   * @}
00558   */
00559
00560 /**
00561   * @}
00562   */
00563
00564 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_eeprom.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval_eeprom.h
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file contains all the functions prototypes for
00008   *          the stm324x9i_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 __STM32F4x9I_EVAL_EEPROM_H
00041 #define __STM32F4x9I_EVAL_EEPROM_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------
-----------------------------------*/
00048 #include "stm324x9i_eval.h"
00049
00050 /** @addtogroup BSP
00051   * @{
00052   */
00053
00054 /** @addtogroup STM324x9I_EVAL
00055   * @{
00056   */
00057
00058 /** @addtogroup STM324x9I_EVAL_EEPROM
00059   * @brief This file includes the I2C EEPROM
 driver of STM324x9I-EVAL evaluation board.
00060   * @{
00061   */
00062
00063 /** @defgroup STM324x9I_EVAL_EEPROM_Exported
_Types STM324x9I EVAL EEPROM Exported Types
00064   * @{
00065   */
00066 /**
```

```
00067   * @}
00068   */
00069
00070 /** @defgroup STM324x9I_EVAL_EEPROM_Exported
_Constants STM324x9I EVAL EEPROM Exported Constants

00071   * @{
00072   */
00073 /* EEPROM hardware address and page size */
00074 #define EEPROM_PAGESIZE             4
00075 #define EEPROM_MAX_SIZE             0x2000 /
* 64Kbit */
00076
00077 /* Maximum Timeout values for flags and even
ts waiting loops.
00078 This timeout is based on systick set to 1ms*/

00079 /* Timeout for read based if read all the EE
PROM : EEPROM_MAX_SIZE * BSP_I2C_SPEED (640ms) */
00080 #define EEPROM_READ_TIMEOUT         ((uint32
_t)(1000))
00081 /* Timeout for write based on max write whic
h is EEPROM_PAGESIZE bytes: EEPROM_PAGESIZE * BSP_
I2C_SPEED (320us) */
00082 #define EEPROM_WRITE_TIMEOUT         ((uint3
2_t)(10))
00083
00084 /* Maximum number of trials for EEPROM_WaitE
epromStandbyState() function */
00085 #define EEPROM_MAX_TRIALS           3000
00086
00087 #define EEPROM_OK                   0
00088 #define EEPROM_FAIL                 1
00089 #define EEPROM_TIMEOUT              2
00090 /**
00091   * @}
00092   */
```

```
00093
00094 /** @defgroup STM324x9I_EVAL_EEPROM_Exported
_Macros STM324x9I EVAL EEPROM Exported Macros
00095   * @{
00096   */
00097 /**
00098   * @}
00099   */
00100
00101 /** @defgroup STM324x9I_EVAL_EEPROM_Exported
_Functions STM324x9I EVAL EEPROM Exported Functions

00102   * @{
00103   */
00104 uint32_t BSP_EEPROM_Init(void);
00105 uint32_t BSP_EEPROM_ReadBuffer(uint8_t* pBuf
fer, uint16_t ReadAddr, uint16_t* NumByteToRead);
00106 uint32_t BSP_EEPROM_WritePage(uint8_t* pBuff
er, uint16_t WriteAddr, uint8_t* NumByteToWrite);
00107 uint32_t BSP_EEPROM_WriteBuffer(uint8_t* pBu
ffer, uint16_t WriteAddr, uint16_t NumByteToWrite)
;
00108 uint32_t BSP_EEPROM_WaitEepromStandbyState(v
oid);
00109
00110 /* USER Callbacks: This function is declared
 as __weak in EEPROM driver and
00111    should be implemented into user applicati
on.
00112    BSP_EEPROM_TIMEOUT_UserCallback() functio
n is called whenever a timeout condition
00113    occurs during communication (waiting on a
n event that doesn't occur, bus
00114    errors, busy devices ...). */
00115 void     BSP_EEPROM_TIMEOUT_UserCallback(void
);
00116
```

```
00117 /* Link function for I2C EEPROM peripheral */

00118 void              EEPROM_IO_Init(void);
00119 HAL_StatusTypeDef EEPROM_IO_WriteData(uint16
_t DevAddress, uint16_t MemAddress, uint8_t *pBuff
er, uint32_t BufferSize);
00120 HAL_StatusTypeDef EEPROM_IO_ReadData(uint16_
t DevAddress, uint16_t MemAddress, uint8_t *pBuffe
r, uint32_t BufferSize);
00121 HAL_StatusTypeDef EEPROM_IO_IsDeviceReady(ui
nt16_t DevAddress, uint32_t Trials);
00122
00123 /**
00124   * @}
00125   */
00126
00127 /**
00128   * @}
00129   */
00130
00131 /**
00132   * @}
00133   */
00134
00135 /**
00136   * @}
00137   */
00138
00139 #ifdef __cplusplus
00140 }
00141 #endif
00142
00143 #endif /* __STM324x9I_EVAL_EEPROM_H */
00144
00145 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP
User Manual by  1.7.6.1

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_eeprom.c

Go to the documentation of this file.

```
00001 /**
00002   ********************************************************************************
00003   * @file    stm324x9i_eval_eeprom.c
00004   * @author  MCD Application Team
00005   * @version V2.2.2
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 STM324x9I_EVAL
00017   *              board.
00018   *              To use this driver you have
to connect the ANT7-M24LR-A to CN3
00019   *              connector of STM324x9I_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 stm324x9i_
eval.h file, and the initialization is
00025   *           performed in EEPROM_IO_Init() f
unction declared in stm324x9i_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
```

```
00077   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00078   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00079   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00080   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
00081   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00082   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00083   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00084   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00085   *
00086   ******************************************
**********************************
00087   */
00088 /* Includes ------------------------------
----------------------------------*/
00089 #include "stm324x9i_eval_eeprom.h"
00090
00091 /** @addtogroup BSP
00092   * @{
00093   */
00094
00095 /** @addtogroup STM324x9I_EVAL
00096   * @{
00097   */
00098
00099 /** @defgroup STM324x9I_EVAL_EEPROM STM324x9
I EVAL EEPROM
00100   * @brief This file includes the I2C EEPROM
 driver of STM324x9I-EVAL evaluation board.
00101   * @{
```

```
00102   */
00103
00104 /** @defgroup STM324x9I_EVAL_EEPROM_Private_
Types STM324x9I EVAL EEPROM Private Types
00105   * @{
00106   */
00107 /**
00108   * @}
00109   */
00110
00111 /** @defgroup STM324x9I_EVAL_EEPROM_Private_
Defines STM324x9I EVAL EEPROM Private Defines
00112   * @{
00113   */
00114 /**
00115   * @}
00116   */
00117
00118 /** @defgroup STM324x9I_EVAL_EEPROM_Private_
Macros STM324x9I EVAL EEPROM Private Macros
00119   * @{
00120   */
00121 /**
00122   * @}
00123   */
00124
00125 /** @defgroup STM324x9I_EVAL_EEPROM_Private_
Variables STM324x9I EVAL EEPROM Private Variables
00126   * @{
00127   */
00128 __IO uint16_t EEPROMAddress = 0;
00129 __IO uint32_t EEPROMTimeout = EEPROM_READ_TI
MEOUT;
00130 __IO uint16_t EEPROMDataRead;
00131 __IO uint8_t  EEPROMDataWrite;
00132 /**
00133   * @}
```

```
00134   */
00135
00136 /** @defgroup STM324x9I_EVAL_EEPROM_Private_
Function_Prototypes STM324x9I EVAL EEPROM Private
Function Prototypes
00137   * @{
00138   */
00139 /**
00140   * @}
00141   */
00142
00143 /** @defgroup STM324x9I_EVAL_EEPROM_Private_
Functions STM324x9I EVAL EEPROM Private Functions
00144   * @{
00145   */
00146
00147 /**
00148   * @brief  Initializes peripherals used by
the I2C EEPROM driver.
00149   *
00150   * @note   There are 2 different versions o
f M24LR64 (A01 & A02).
00151   *             Then try to connect on 1st o
ne (EEPROM_I2C_ADDRESS_A01)
00152   *             and if problem, check the 2n
d one (EEPROM_I2C_ADDRESS_A02)
00153   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00154   *         different from EEPROM_OK (0)
00155   */
00156 uint32_t BSP_EEPROM_Init(void)
00157 {
00158   /* I2C Initialization */
00159   EEPROM_IO_Init();
00160
00161   /* Select the EEPROM address for A01 and c
heck if OK */
```

```
00162   EEPROMAddress = EEPROM_I2C_ADDRESS_A01;
00163   if(EEPROM_IO_IsDeviceReady(EEPROMAddress,
EEPROM_MAX_TRIALS) != HAL_OK)
00164   {
00165     /* Select the EEPROM address for A02 and
 check if OK */
00166     EEPROMAddress = EEPROM_I2C_ADDRESS_A02;
00167     if(EEPROM_IO_IsDeviceReady(EEPROMAddress
, EEPROM_MAX_TRIALS) != HAL_OK)
00168     {
00169       return EEPROM_FAIL;
00170     }
00171   }
00172   return EEPROM_OK;
00173 }
00174
00175 /**
00176   * @brief  Reads a block of data from the E
EPROM.
00177   * @param  pBuffer: pointer to the buffer t
hat receives the data read from
00178   *         the EEPROM.
00179   * @param  ReadAddr: EEPROM's internal addr
ess to start reading from.
00180   * @param  NumByteToRead: pointer to the va
riable holding number of bytes to
00181   *         be read from the EEPROM.
00182   *
00183   *        @note The variable pointed by Num
ByteToRead is reset to 0 when all the
00184   *              data are read from the EEPR
OM. Application should monitor this
00185   *              variable in order know when
 the transfer is complete.
00186   *
00187   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
```

```
00188   *         different from EEPROM_OK (0) or
the timeout user callback.
00189   */
00190 uint32_t BSP_EEPROM_ReadBuffer(uint8_t* pBuf
fer, uint16_t ReadAddr, uint16_t* NumByteToRead)
00191 {
00192   uint32_t buffersize = *NumByteToRead;
00193
00194   /* Set the pointer to the Number of data t
o be read. This pointer will be used
00195      by the DMA Transfer Completer interrupt
 Handler in order to reset the
00196      variable to 0. User should check on thi
s variable in order to know if the
00197      DMA transfer has been complete or not.
*/
00198   EEPROMDataRead = *NumByteToRead;
00199
00200   if(EEPROM_IO_ReadData(EEPROMAddress, ReadA
ddr, pBuffer, buffersize) != HAL_OK)
00201   {
00202     BSP_EEPROM_TIMEOUT_UserCallback();
00203     return EEPROM_FAIL;
00204   }
00205
00206   /* If all operations OK, return EEPROM_OK
(0) */
00207   return EEPROM_OK;
00208 }
00209
00210 /**
00211   * @brief  Writes more than one byte to the
 EEPROM with a single WRITE cycle.
00212   *
00213   * @note   The number of bytes (combined to
 write start address) must not
00214   *         cross the EEPROM page boundary.
```

```
This function can only write into
00215   *         the boundaries of an EEPROM page.

00216   *         This function doesn't check on b
oundaries condition (in this driver
00217   *         the function BSP_EEPROM_WriteBuf
fer() which calls BSP_EEPROM_WritePage() is
00218   *         responsible of checking on Page
boundaries).
00219   *
00220   * @param  pBuffer: pointer to the buffer c
ontaining the data to be written to
00221   *         the EEPROM.
00222   * @param  WriteAddr: EEPROM's internal add
ress to write to.
00223   * @param  NumByteToWrite: pointer to the v
ariable holding number of bytes to
00224   *         be written into the EEPROM.
00225   *
00226   *       @note The variable pointed by Num
ByteToWrite is reset to 0 when all the
00227   *             data are written to the EEP
ROM. Application should monitor this
00228   *             variable in order know when
 the transfer is complete.
00229   *
00230   *       @note This function just configur
e the communication and enable the DMA
00231   *             channel to transfer data. M
eanwhile, the user application may perform
00232   *             other tasks in parallel.
00233   *
00234   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00235   *         different from EEPROM_OK (0) or
the timeout user callback.
00236   */
```

```
00237 uint32_t BSP_EEPROM_WritePage(uint8_t* pBuff
er, uint16_t WriteAddr, uint8_t* NumByteToWrite)
00238 {
00239   uint32_t buffersize = *NumByteToWrite;
00240   uint32_t status = EEPROM_OK;
00241
00242   /* Set the pointer to the Number of data t
o be written. This pointer will be used
00243       by the DMA Transfer Completer interrup
t Handler in order to reset the
00244       variable to 0. User should check on th
is variable in order to know if the
00245       DMA transfer has been complete or not.
 */
00246   EEPROMDataWrite = *NumByteToWrite;
00247
00248   if(EEPROM_IO_WriteData(EEPROMAddress, Writ
eAddr, pBuffer, buffersize) != HAL_OK)
00249   {
00250     BSP_EEPROM_TIMEOUT_UserCallback();
00251     status = EEPROM_FAIL;
00252   }
00253
00254   if(BSP_EEPROM_WaitEepromStandbyState() !=
EEPROM_OK)
00255   {
00256     return EEPROM_FAIL;
00257   }
00258
00259   /* If all operations OK, return EEPROM_OK
(0) */
00260   return status;
00261 }
00262
00263 /**
00264   * @brief  Writes buffer of data to the I2C
 EEPROM.
```

```
00265   * @param  pBuffer: pointer to the buffer
containing the data to be written
00266   *         to the EEPROM.
00267   * @param  WriteAddr: EEPROM's internal add
ress to write to.
00268   * @param  NumByteToWrite: number of bytes
to write to the EEPROM.
00269   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00270   *         different from EEPROM_OK (0) or
the timeout user callback.
00271   */
00272 uint32_t BSP_EEPROM_WriteBuffer(uint8_t *pBu
ffer, uint16_t WriteAddr, uint16_t NumByteToWrite)
00273 {
00274   uint16_t numofpage = 0, numofsingle = 0, c
ount = 0;
00275   uint16_t addr = 0;
00276   uint8_t  dataindex = 0;
00277   uint32_t status = EEPROM_OK;
00278
00279   addr = WriteAddr % EEPROM_PAGESIZE;
00280   count = EEPROM_PAGESIZE - addr;
00281   numofpage =  NumByteToWrite / EEPROM_PAGES
IZE;
00282   numofsingle = NumByteToWrite % EEPROM_PAGE
SIZE;
00283
00284   /* If WriteAddr is EEPROM_PAGESIZE aligned
 */
00285   if(addr == 0)
00286   {
00287     /* If NumByteToWrite < EEPROM_PAGESIZE */

00288     if(numofpage == 0)
00289     {
00290       /* Store the number of data to be writ
```

```
ten */
00291       dataindex = numofsingle;
00292       /* Start writing data */
00293       status = BSP_EEPROM_WritePage(pBuffer,
 WriteAddr, (uint8_t*)(&dataindex));
00294       if(status != EEPROM_OK)
00295       {
00296         return status;
00297       }
00298     }
00299     /* If NumByteToWrite > EEPROM_PAGESIZE */

00300     else
00301     {
00302       while(numofpage--)
00303       {
00304         /* Store the number of data to be wr
itten */
00305         dataindex = EEPROM_PAGESIZE;
00306         status = BSP_EEPROM_WritePage(pBuffe
r, WriteAddr, (uint8_t*)(&dataindex));
00307         if(status != EEPROM_OK)
00308         {
00309           return status;
00310         }
00311
00312         WriteAddr +=  EEPROM_PAGESIZE;
00313         pBuffer += EEPROM_PAGESIZE;
00314       }
00315
00316       if(numofsingle!=0)
00317       {
00318         /* Store the number of data to be wr
itten */
00319         dataindex = numofsingle;
00320         status = BSP_EEPROM_WritePage(pBuffe
r, WriteAddr, (uint8_t*)(&dataindex));
```

```
00321         if(status != EEPROM_OK)
00322         {
00323           return status;
00324         }
00325       }
00326     }
00327   }
00328   /* If WriteAddr is not EEPROM_PAGESIZE ali
gned */
00329   else
00330   {
00331     /* If NumByteToWrite < EEPROM_PAGESIZE */

00332     if(numofpage== 0)
00333     {
00334       /* If the number of data to be written
 is more than the remaining space
00335       in the current page: */
00336       if(NumByteToWrite > count)
00337       {
00338         /* Store the number of data to be wr
itten */
00339         dataindex = count;
00340         /* Write the data contained in same
page */
00341         status = BSP_EEPROM_WritePage(pBuffe
r, WriteAddr, (uint8_t*)(&dataindex));
00342         if(status != EEPROM_OK)
00343         {
00344           return status;
00345         }
00346
00347         /* Store the number of data to be wr
itten */
00348         dataindex = (NumByteToWrite - count)
;
00349         /* Write the remaining data in the f
```

```
ollowing page */
00350         status = BSP_EEPROM_WritePage((uint8
_t*)(pBuffer + count), (WriteAddr + count), (uint8
_t*)(&dataindex));
00351         if(status != EEPROM_OK)
00352         {
00353           return status;
00354         }
00355       }
00356       else
00357       {
00358         /* Store the number of data to be wr
itten */
00359         dataindex = numofsingle;
00360         status = BSP_EEPROM_WritePage(pBuffe
r, WriteAddr, (uint8_t*)(&dataindex));
00361         if(status != EEPROM_OK)
00362         {
00363           return status;
00364         }
00365       }
00366     }
00367     /* If NumByteToWrite > EEPROM_PAGESIZE */

00368     else
00369     {
00370       NumByteToWrite -= count;
00371       numofpage =  NumByteToWrite / EEPROM_P
AGESIZE;
00372       numofsingle = NumByteToWrite % EEPROM_
PAGESIZE;
00373
00374       if(count != 0)
00375       {
00376         /* Store the number of data to be wr
itten */
00377         dataindex = count;
```

```
00378           status = BSP_EEPROM_WritePage(pBuffe
r, WriteAddr, (uint8_t*)(&dataindex));
00379           if(status != EEPROM_OK)
00380           {
00381             return status;
00382           }
00383           WriteAddr += count;
00384           pBuffer += count;
00385         }
00386
00387         while(numofpage--)
00388         {
00389           /* Store the number of data to be wr
itten */
00390           dataindex = EEPROM_PAGESIZE;

00391           status = BSP_EEPROM_WritePage(pBuffe
r, WriteAddr, (uint8_t*)(&dataindex));
00392           if(status != EEPROM_OK)
00393           {
00394             return status;
00395           }
00396           WriteAddr +=  EEPROM_PAGESIZE;
00397           pBuffer += EEPROM_PAGESIZE;
00398         }
00399         if(numofsingle != 0)
00400         {
00401           /* Store the number of data to be wr
itten */
00402           dataindex = numofsingle;
00403           status = BSP_EEPROM_WritePage(pBuffe
r, WriteAddr, (uint8_t*)(&dataindex));
00404           if(status != EEPROM_OK)
00405           {
00406             return status;
00407           }
00408         }
```

```
00409     }
00410   }
00411
00412   /* If all operations OK, return EEPROM_OK
(0) */
00413   return EEPROM_OK;
00414 }
00415
00416 /**
00417   * @brief  Wait for EEPROM Standby state.
00418   *
00419   * @note  This function allows to wait and
check that EEPROM has finished the
00420   *        last operation. It is mostly used
 after Write operation: after receiving
00421   *        the buffer to be written, the EEP
ROM may need additional time to actually
00422   *        perform the write operation. Duri
ng this time, it doesn't answer to
00423   *        I2C packets addressed to it. Once
 the write operation is complete
00424   *        the EEPROM responds to its addres
s.
00425   *
00426   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00427   *         different from EEPROM_OK (0) or
the timeout user callback.
00428   */
00429 uint32_t BSP_EEPROM_WaitEepromStandbyState(v
oid)
00430 {
00431   /* Check if the maximum allowed number of
trials has bee reached */
00432   if(EEPROM_IO_IsDeviceReady(EEPROMAddress,
EEPROM_MAX_TRIALS) != HAL_OK)
00433   {
```

```
00434     /* If the maximum number of trials has b
een reached, exit the function */
00435     BSP_EEPROM_TIMEOUT_UserCallback();
00436     return EEPROM_TIMEOUT;
00437   }
00438   return EEPROM_OK;
00439 }
00440
00441 /**
00442   * @brief  Basic management of the timeout
situation.
00443   */
00444 __weak void BSP_EEPROM_TIMEOUT_UserCallback(
void)
00445 {
00446 }
00447
00448 /**
00449   * @}
00450   */
00451
00452 /**
00453   * @}
00454   */
00455
00456 /**
00457   * @}
00458   */
00459
00460 /**
00461   * @}
00462   */
00463
00464 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP
User Manual by  1.7.6.1

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_io.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval_io.c
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file provides a set of functions needed to manage the IO pins
00008   *          on STM324x9I-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 STM324x9I-EVAL evaluation
00044      board.
00045    - The STMPE1600 IO expander device compon
ent driver must be included with this
00046      driver in order to run the IO functiona
lities commanded by the IO expander
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 To get/set an IO pin combination stat
e you can use the functions
00067        BSP_IO_ReadPin()/BSP_IO_WritePin() or
 the function BSP_IO_TogglePin() to toggle the pin

00068        state.
00069
00070 ----------------------------------------
----------------------------------*/
00071
00072 /* Includes ------------------------------
----------------------------------*/
00073 #include "stm324x9i_eval_io.h"
00074
00075 /** @addtogroup BSP
00076   * @{
00077   */
00078
00079 /** @addtogroup STM324x9I_EVAL
00080   * @{
00081   */
00082
00083 /** @defgroup STM324x9I_EVAL_IO STM324x9I EV
AL IO
00084   * @{
00085   */
00086
```

```
00087 /** @defgroup STM324x9I_EVAL_IO_Private_Types_Definitions STM324x9I EVAL IO Private Types Definitions
00088   * @{
00089   */
00090 /**
00091   * @}
00092   */
00093
00094 /** @defgroup STM324x9I_EVAL_IO_Private_Defines STM324x9I EVAL IO Private Defines
00095   * @{
00096   */
00097 /**
00098   * @}
00099   */
00100
00101 /** @defgroup STM324x9I_EVAL_IO_Private_Macros STM324x9I EVAL IO Private Macros
00102   * @{
00103   */
00104 /**
00105   * @}
00106   */
00107
00108 /** @defgroup STM324x9I_EVAL_IO_Private_Variables STM324x9I EVAL IO Private Variables
00109   * @{
00110   */
00111 static IO_DrvTypeDef *io_driver;
00112 /**
00113   * @}
00114   */
00115
00116 /** @defgroup STM324x9I_EVAL_IO_Private_Function_Prototypes STM324x9I EVAL IO Private Function Prototypes
```

```
00117   * @{
00118   */
00119 /**
00120   * @}
00121   */
00122
00123 /** @defgroup STM324x9I_EVAL_IO_Private_Func
tions STM324x9I EVAL IO Private Functions
00124   * @{
00125   */
00126
00127 /**
00128   * @brief  Initializes and configures the I
O functionalities and configures all
00129   *         necessary hardware resources (GP
IOs, clocks..).
00130   * @note   BSP_IO_Init() is using HAL_Delay
() function to ensure that stmpe1600
00131   *         IO Expander is correctly reset.
HAL_Delay() function provides accurate
00132   *         delay (in milliseconds) based on
 variable incremented in SysTick ISR.
00133   *         This implies that if BSP_IO_Init
() is called from a peripheral ISR process,
00134   *         then the SysTick interrupt must
have higher priority (numerically lower)
00135   *         than the peripheral interrupt. O
therwise the caller ISR process will be blocked.
00136   * @retval IO_OK if all initializations are
 OK. Other value if error.
00137   */
00138 uint8_t BSP_IO_Init(void)
00139 {
00140   uint8_t ret = IO_ERROR;
00141
00142   /* Read ID and verify the IO expander is r
eady */
```

```
00143   if(stmpe1600_io_drv.ReadID(IO_I2C_ADDRESS)
 == STMPE1600_ID)
00144   {
00145     /* Initialize the IO driver structure */
00146     io_driver = &stmpe1600_io_drv;
00147     ret = IO_OK;
00148   }
00149
00150   if(ret == IO_OK)
00151   {
00152     io_driver->Init(IO_I2C_ADDRESS);
00153     io_driver->Start(IO_I2C_ADDRESS, IO_PIN_
ALL);
00154   }
00155   return ret;
00156 }
00157
00158 /**
00159   * @brief  Gets the selected pins IT status.

00160   * @param  IO_Pin: Selected pins to check t
he status.
00161   *          This parameter can be any combi
nation of the IO pins.
00162   * @retval IO_OK if read status OK. Other v
alue if error.
00163   */
00164 uint8_t BSP_IO_ITGetStatus(uint16_t IO_Pin)
00165 {
00166   /* Return the IO Pin IT status */
00167   return (io_driver->ITStatus(IO_I2C_ADDRESS
, IO_Pin));
00168 }
00169
00170 /**
00171   * @brief  Clears all the IO IT pending bit
s.
```

```
00172   */
00173 void BSP_IO_ITClear(void)
00174 {
00175   /* Clear all IO IT pending bits */
00176   io_driver->ClearIT(IO_I2C_ADDRESS, STMPE16
00_PIN_ALL);
00177 }
00178
00179 /**
00180   * @brief  Configures the IO pin(s) accordi
ng to IO mode structure value.
00181   * @param  IO_Pin: IO pin(s) to be configur
ed.
00182   *          This parameter can be one of th
e following values:
00183   *            @arg  STMPE1600_PIN_x: where
x can be from 0 to 15.
00184   * @param  IO_Mode: IO pin mode to configure

00185   *          This parameter can be one of th
e following values:
00186   *            @arg  IO_MODE_INPUT
00187   *            @arg  IO_MODE_OUTPUT
00188   *            @arg  IO_MODE_IT_RISING_EDGE
00189   *            @arg  IO_MODE_IT_FALLING_EDGE
00190   * @retval IO_OK if all initializations are
 OK. Other value if error.
00191   */
00192 uint8_t BSP_IO_ConfigPin(uint16_t IO_Pin, IO
_ModeTypedef IO_Mode)
00193 {
00194   /* Configure the selected IO pin(s) mode */

00195   io_driver->Config(IO_I2C_ADDRESS, (uint16_
t )IO_Pin, IO_Mode);
00196
00197   return IO_OK;
```

```
00198 }
00199
00200 /**
00201   * @brief  Sets the selected pins state.
00202   * @param  IO_Pin: Selected pins to write.
00203   *          This parameter can be any combi
nation of the IO pins.
00204   * @param  PinState: New pins state to writ
e
00205   */
00206 void BSP_IO_WritePin(uint16_t IO_Pin, uint8_
t PinState)
00207 {
00208   /* Set the Pin state */
00209   io_driver->WritePin(IO_I2C_ADDRESS, IO_Pin
, PinState);
00210 }
00211
00212 /**
00213   * @brief  Gets the selected pins current s
tate.
00214   * @param  IO_Pin: Selected pins to read.
00215   *          This parameter can be any combi
nation of the IO pins.
00216   * @retval The current pins state
00217   */
00218 uint16_t BSP_IO_ReadPin(uint16_t IO_Pin)
00219 {
00220  return(io_driver->ReadPin(IO_I2C_ADDRESS, I
O_Pin));
00221 }
00222
00223 /**
00224   * @brief  Toggles the selected pins state.
00225   * @param  IO_Pin: Selected pins to toggle.

00226   *          This parameter can be any combi
```

```
nation of the IO pins.
00227   */
00228 void BSP_IO_TogglePin(uint16_t IO_Pin)
00229 {
00230   /* Toggle the current pin state */
00231   if(io_driver->ReadPin(IO_I2C_ADDRESS, IO_P
in) == 1) /* Set */
00232   {
00233     io_driver->WritePin(IO_I2C_ADDRESS, IO_P
in, 0); /* Reset */
00234   }
00235   else
00236   {
00237     io_driver->WritePin(IO_I2C_ADDRESS, IO_P
in, 1); /* Set */
00238   }
00239 }
00240
00241 /**
00242   * @}
00243   */
00244
00245 /**
00246   * @}
00247   */
00248
00249 /**
00250   * @}
00251   */
00252
00253 /**
00254   * @}
00255   */
00256
00257 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP
User Manual by  1.7.6.1

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_sdram.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval_sdram.h
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm324x9i_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 __STM324x9I_EVAL_SDRAM_H
00041 #define __STM324x9I_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 STM324x9I_EVAL
00055   * @{
00056   */
00057
00058 /** @addtogroup STM324x9I_EVAL_SDRAM
00059   * @{
00060   */
00061
00062 /** @defgroup STM324x9I_EVAL_SDRAM_Exported_
Types STM324x9I EVAL SDRAM Exported Types
00063   * @{
00064   */
00065
00066 /**
00067   * @brief  SDRAM status structure definitio
```

```
n
00068   */
00069 #define   SDRAM_OK         0x00
00070 #define   SDRAM_ERROR      0x01
00071
00072 /**
00073   * @}
00074   */
00075
00076 /** @defgroup STM324x9I_EVAL_SDRAM_Exported_
Constants STM324x9I 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
_SDRAM_MEM_BUS_WIDTH_16 */
00084 #define SDRAM_MEMORY_WIDTH               FMC
_SDRAM_MEM_BUS_WIDTH_32
00085
00086 #define SDCLOCK_PERIOD                   FMC
_SDRAM_CLOCK_PERIOD_2
00087 /* #define SDCLOCK_PERIOD                FMC
_SDRAM_CLOCK_PERIOD_3 */
00088
00089 #define REFRESH_COUNT                    ((u
int32_t)0x0569)   /* SDRAM refresh counter (90Mhz
SD clock) */
00090
00091 #define SDRAM_TIMEOUT     ((uint32_t)0xFFFF)

00092
```

```
00093 /* DMA definitions for SDRAM DMA transfer */
00094 #define __DMAx_CLK_ENABLE                   __
DMA2_CLK_ENABLE
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   * @brief  FMC SDRAM Mode definition regist
er defines
00102   */
00103 #define SDRAM_MODEREG_BURST_LENGTH_1
     ((uint16_t)0x0000)
00104 #define SDRAM_MODEREG_BURST_LENGTH_2
     ((uint16_t)0x0001)
00105 #define SDRAM_MODEREG_BURST_LENGTH_4
     ((uint16_t)0x0002)
00106 #define SDRAM_MODEREG_BURST_LENGTH_8
     ((uint16_t)0x0004)
00107 #define SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL
     ((uint16_t)0x0000)
00108 #define SDRAM_MODEREG_BURST_TYPE_INTERLEAVED
     ((uint16_t)0x0008)
00109 #define SDRAM_MODEREG_CAS_LATENCY_2
     ((uint16_t)0x0020)
00110 #define SDRAM_MODEREG_CAS_LATENCY_3
     ((uint16_t)0x0030)
00111 #define SDRAM_MODEREG_OPERATING_MODE_STANDAR
D    ((uint16_t)0x0000)
00112 #define SDRAM_MODEREG_WRITEBURST_MODE_PROGRA
MMED ((uint16_t)0x0000)
00113 #define SDRAM_MODEREG_WRITEBURST_MODE_SINGLE
```

```
     ((uint16_t)0x0200)
00114 /**
00115   * @}
00116   */
00117
00118 /** @defgroup STM324x9I_EVAL_SDRAM_Exported_
Macro STM324x9I EVAL SDRAM Exported Macro
00119   * @{
00120   */
00121 /**
00122   * @}
00123   */
00124
00125 /** @defgroup STM324x9I_EVAL_SDRAM_Exported_
Functions STM324x9I EVAL SDRAM Exported Functions
00126   * @{
00127   */
00128 uint8_t BSP_SDRAM_Init(void);
00129 void    BSP_SDRAM_Initialization_sequence(ui
nt32_t RefreshCount);
00130 uint8_t BSP_SDRAM_ReadData(uint32_t uwStartA
ddress, uint32_t *pData, uint32_t uwDataSize);
00131 uint8_t BSP_SDRAM_ReadData_DMA(uint32_t uwSt
artAddress, uint32_t *pData, uint32_t uwDataSize);
00132 uint8_t BSP_SDRAM_WriteData(uint32_t uwStart
Address, uint32_t *pData, uint32_t uwDataSize);
00133 uint8_t BSP_SDRAM_WriteData_DMA(uint32_t uwS
tartAddress, uint32_t *pData, uint32_t uwDataSize)
;
00134 uint8_t BSP_SDRAM_Sendcmd(FMC_SDRAM_CommandT
ypeDef *SdramCmd);
00135 void    BSP_SDRAM_DMA_IRQHandler(void);
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 /* __STM324x9I_EVAL_SDRAM_H */
00158
00159 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_nor.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval_nor.h
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm324x9i_eval_nor.c driver.
00009   ******************************************************************************
00010   * @attention
00011   *
00012   * <h2><center>&copy; COPYRIGHT(c) 2015 STMicroelectronics</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without modification,
00015   * are permitted provided that the following conditions are met:
```

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

```
00035   *
00036   ******************************************
*********************************
00037   */
00038
00039 /* Define to prevent recursive inclusion ---
-----------------------------------*/
00040 #ifndef __STM324x9I_EVAL_NOR_H
00041 #define __STM324x9I_EVAL_NOR_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 STM324x9I_EVAL
00055   * @{
00056   */
00057
00058 /** @defgroup STM324x9I_EVAL_NOR STM324x9I E
VAL NOR
00059   * @{
00060   */
00061
00062 /** @defgroup STM324x9I_EVAL_NOR_Exported_Ty
pes STM324x9I EVAL NOR Exported Types
00063   * @{
00064   */
00065 /**
00066   * @}
```

```
00067   */
00068
00069 /**
00070   * @brief  NOR status structure definition

00071   */
00072 #define   NOR_STATUS_OK         0x00
00073 #define   NOR_STATUS_ERROR      0x01
00074
00075 /** @defgroup STM324x9I_EVAL_NOR_Exported_Co
nstants STM324x9I EVAL NOR Exported Constants
00076   * @{
00077   */
00078 #define NOR_DEVICE_ADDR  ((uint32_t)0x600000
00)
00079
00080 /* #define NOR_MEMORY_WIDTH    FMC_NORSRAM_M
EM_BUS_WIDTH_8  */
00081 #define NOR_MEMORY_WIDTH    FMC_NORSRAM_MEM_
BUS_WIDTH_16
00082
00083 #define NOR_BURSTACCESS    FMC_BURST_ACCESS_
MODE_DISABLE
00084 /* #define NOR_BURSTACCESS    FMC_BURST_ACCE
SS_MODE_ENABLE*/
00085
00086 #define NOR_WRITEBURST    FMC_WRITE_BURST_DI
SABLE
00087 /* #define NOR_WRITEBURST   FMC_WRITE_BURST_
ENABLE */
00088
00089 #define CONTINUOUSCLOCK_FEATURE    FMC_CONTI
NUOUS_CLOCK_SYNC_ONLY
00090 /* #define CONTINUOUSCLOCK_FEATURE     FMC_C
ONTINUOUS_CLOCK_SYNC_ASYNC */
00091
00092 /* NOR operations Timeout definitions */
```

```
00093 #define BLOCKERASE_TIMEOUT   ((uint32_t)0x00
A00000)  /* NOR block erase timeout */
00094 #define CHIPERASE_TIMEOUT    ((uint32_t)0x30
000000)  /* NOR chip erase timeout  */
00095 #define PROGRAM_TIMEOUT      ((uint32_t)0x00
004400)  /* NOR program timeout     */
00096
00097 /* NOR Ready/Busy signal GPIO definitions */
00098 #define NOR_READY_BUSY_PIN    GPIO_PIN_6
00099 #define NOR_READY_BUSY_GPIO   GPIOD
00100 #define NOR_READY_STATE       GPIO_PIN_SET
00101 #define NOR_BUSY_STATE        GPIO_PIN_RESET

00102 /**
00103   * @}
00104   */
00105
00106 /** @defgroup STM324x9I_EVAL_NOR_Exported_Ma
cro STM324x9I EVAL NOR Exported Macro
00107   * @{
00108   */
00109 /**
00110   * @}
00111   */
00112
00113 /** @defgroup STM324x9I_EVAL_NOR_Exported_Fu
nctions STM324x9I EVAL NOR Exported Functions
00114   * @{
00115   */
00116 uint8_t BSP_NOR_Init(void);
00117 uint8_t BSP_NOR_ReadData(uint32_t uwStartAdd
ress, uint16_t *pData, uint32_t uwDataSize);
00118 uint8_t BSP_NOR_WriteData(uint32_t uwStartAd
dress, uint16_t *pData, uint32_t uwDataSize);
00119 uint8_t BSP_NOR_ProgramData(uint32_t uwStart
Address, uint16_t *pData, uint32_t uwDataSize);
00120 uint8_t BSP_NOR_Erase_Block(uint32_t BlockAd
```

```
dress);
00121 uint8_t BSP_NOR_Erase_Chip(void);
00122 uint8_t BSP_NOR_Read_ID(NOR_IDTypeDef *pNOR_
ID);
00123 void BSP_NOR_ReturnToReadMode(void);
00124
00125 /**
00126   * @}
00127   */
00128
00129 /**
00130   * @}
00131   */
00132
00133 /**
00134   * @}
00135   */
00136
00137 /**
00138   * @}
00139   */
00140
00141 #ifdef __cplusplus
00142 }
00143 #endif
00144
00145 #endif /* __STM324x9I_EVAL_NOR_H */
00146
00147 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_nor.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval_nor.c
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file includes a standard driver for the M29W256GL70ZA6E NOR flash memory
00008   *          device mounted on STM324x9I-EVAL evaluation board.
00009   ******************************************************************************
00010   * @attention
00011   *
00012   * <h2><center>&copy; COPYRIGHT(c) 2015 STMicroelectronics</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without modification,
00015   * are permitted provided that the following conditions are met:
```

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

```
00035   *
00036   ******************************************
*********************************
00037   */
00038
00039 /* File Info : ----------------------------
-----------------------------------
00040                                       User NOTES

00041 1. How To use this driver:
00042 --------------------------
00043    - This driver is used to drive the M29W12
8GL NOR flash external memory mounted
00044      on STM324x9I-EVAL evaluation board.
00045    - This driver does not need a specific co
mponent driver for the NOR device
00046      to be included with.
00047
00048 2. Driver description:
00049 ---------------------
00050   + Initialization steps:
00051      o Initialize the NOR external memory us
ing the BSP_NOR_Init() function. This
00052        function includes the MSP layer hardw
are resources initialization and the
00053        FMC controller configuration to inter
face with the external NOR memory.
00054
00055   + NOR flash operations
00056      o NOR external memory can be accessed w
ith read/write operations once it is
00057        initialized.
00058        Read/write operation can be performed
 with AHB access using the functions
00059        BSP_NOR_ReadData()/BSP_NOR_WriteData(
). The BSP_NOR_WriteData() performs write operation
```

```
00060         of an amount of data by unit (halfwor
d). You can also perform a program data
00061         operation of an amount of data using
the function BSP_NOR_ProgramData().
00062       o The function BSP_NOR_Read_ID() return
s the chip IDs stored in the structure
00063         "NOR_IDTypeDef". (see the NOR IDs in
the memory data sheet)
00064       o Perform erase block operation using t
he function BSP_NOR_Erase_Block() and by
00065         specifying the block address. You can
 perform an erase operation of the whole
00066         chip by calling the function BSP_NOR_
Erase_Chip().
00067       o After other operations, the function
BSP_NOR_ReturnToReadMode() allows the NOR
00068         flash to return to read mode to perfo
rm read operations on it.
00069
00070 ------------------------------------------
-----------------------------------*/
00071
00072 /* Includes ---------------------------------
-----------------------------------*/
00073 #include "stm324x9i_eval_nor.h"
00074
00075 /** @addtogroup BSP
00076   * @{
00077   */
00078
00079 /** @addtogroup STM324x9I_EVAL
00080   * @{
00081   */
00082
00083 /** @defgroup STM324x9I_EVAL_NOR STM324x9I E
VAL NOR
00084   * @{
```

```
00085   */
00086
00087 /* Private typedef -----------------------
-----------------------------------*/
00088
00089 /** @defgroup STM324x9I_EVAL_NOR_Private_Typ
es_Definitions STM324x9I EVAL NOR Private Types De
finitions
00090   * @{
00091   */
00092
00093 /**
00094   * @}
00095   */
00096
00097 /* Private define ------------------------
-----------------------------------*/
00098
00099 /** @defgroup STM324x9I_EVAL_NOR_Private_Def
ines STM324x9I EVAL NOR Private Defines
00100   * @{
00101   */
00102
00103 /**
00104   * @}
00105   */
00106
00107 /* Private macro -------------------------
-----------------------------------*/
00108
00109 /** @defgroup STM324x9I_EVAL_NOR_Private_Mac
ros STM324x9I EVAL NOR Private Macros
00110   * @{
00111   */
00112
00113 /**
00114   * @}
```

```
00115   */
00116
00117 /* Private variables ---------------------
---------------------------------*/
00118
00119 /** @defgroup STM324x9I_EVAL_NOR_Private_Var
iables STM324x9I EVAL NOR Private Variables
00120   * @{
00121   */
00122 static NOR_HandleTypeDef norHandle;
00123 static FMC_NORSRAM_TimingTypeDef Timing;
00124
00125 /**
00126   * @}
00127   */
00128
00129 /* Private function prototypes -----------
---------------------------------*/
00130
00131 /** @defgroup STM324x9I_EVAL_NOR_Private_Fun
ction_Prototypes STM324x9I EVAL NOR Private Functi
on Prototypes
00132   * @{
00133   */
00134
00135 /**
00136   * @}
00137   */
00138
00139 /* Private functions ---------------------
---------------------------------*/
00140
00141 /** @defgroup STM324x9I_EVAL_NOR_Private_Fun
ctions STM324x9I EVAL NOR Private Functions
00142   * @{
00143   */
00144 static void NOR_MspInit(void);
```

```
00145
00146 /**
00147   * @}
00148   */
00149
00150 /**
00151   * @brief  Initializes the NOR device.
00152   * @retval NOR memory status
00153   */
00154 uint8_t BSP_NOR_Init(void)
00155 {
00156   norHandle.Instance  = FMC_NORSRAM_DEVICE;
00157   norHandle.Extended  = FMC_NORSRAM_EXTENDED
_DEVICE;
00158
00159   /* NOR device configuration */
00160   Timing.AddressSetupTime      = 4;
00161   Timing.AddressHoldTime       = 3;
00162   Timing.DataSetupTime         = 7;
00163   Timing.BusTurnAroundDuration = 1;
00164   Timing.CLKDivision           = 2;
00165   Timing.DataLatency           = 2;
00166   Timing.AccessMode            = FMC_ACCESS_
MODE_A;
00167
00168   norHandle.Init.NSBank             = FMC_NO
RSRAM_BANK1;
00169   norHandle.Init.DataAddressMux     = FMC_DA
TA_ADDRESS_MUX_DISABLE;
00170   norHandle.Init.MemoryType         = FMC_ME
MORY_TYPE_NOR;
00171   norHandle.Init.MemoryDataWidth    = NOR_ME
MORY_WIDTH;
00172   norHandle.Init.BurstAccessMode    = NOR_BU
RSTACCESS;
00173   norHandle.Init.WaitSignalPolarity = FMC_WA
IT_SIGNAL_POLARITY_LOW;
```

```
00174   norHandle.Init.WrapMode           = FMC_WR
AP_MODE_DISABLE;
00175   norHandle.Init.WaitSignalActive   = FMC_WA
IT_TIMING_BEFORE_WS;
00176   norHandle.Init.WriteOperation     = FMC_WR
ITE_OPERATION_ENABLE;
00177   norHandle.Init.WaitSignal         = FMC_WA
IT_SIGNAL_ENABLE;
00178   norHandle.Init.ExtendedMode       = FMC_EX
TENDED_MODE_DISABLE;
00179   norHandle.Init.AsynchronousWait   = FMC_AS
YNCHRONOUS_WAIT_ENABLE;
00180   norHandle.Init.WriteBurst         = NOR_WR
ITEBURST;
00181   norHandle.Init.ContinuousClock    = CONTIN
UOUSCLOCK_FEATURE;
00182
00183   /* NOR controller initialization */
00184   NOR_MspInit();
00185
00186   if(HAL_NOR_Init(&norHandle, &Timing, &Timi
ng) != HAL_OK)
00187   {
00188     return NOR_STATUS_ERROR;
00189   }
00190   else
00191   {
00192     return NOR_STATUS_OK;
00193   }
00194 }
00195
00196 /**
00197   * @brief  Reads an amount of data from the
 NOR device.
00198   * @param  uwStartAddress: Read start addre
ss
00199   * @param  pData: Pointer to data to be read
```

```
00200   * @param  uwDataSize: Size of data to read

00201   * @retval NOR memory status
00202   */
00203 uint8_t BSP_NOR_ReadData(uint32_t uwStartAdd
ress, uint16_t* pData, uint32_t uwDataSize)
00204 {
00205   if(HAL_NOR_ReadBuffer(&norHandle, NOR_DEVI
CE_ADDR + uwStartAddress, pData, uwDataSize) != HA
L_OK)
00206   {
00207     return NOR_STATUS_ERROR;
00208   }
00209   else
00210   {
00211     return NOR_STATUS_OK;
00212   }
00213 }
00214
00215 /**
00216   * @brief  Returns the NOR memory to read m
ode.
00217   */
00218 void BSP_NOR_ReturnToReadMode(void)
00219 {
00220    HAL_NOR_ReturnToReadMode(&norHandle);
00221 }
00222
00223 /**
00224   * @brief  Writes an amount of data to the
NOR device.
00225   * @param  uwStartAddress: Write start addr
ess
00226   * @param  pData: Pointer to data to be wri
tten
00227   * @param  uwDataSize: Size of data to writ
```

```
e
00228   * @retval NOR memory status
00229   */
00230 uint8_t BSP_NOR_WriteData(uint32_t uwStartAd
dress, uint16_t* pData, uint32_t uwDataSize)
00231 {
00232   uint32_t index = uwDataSize;
00233
00234   while(index > 0)
00235   {
00236     /* Write data to NOR */
00237     HAL_NOR_Program(&norHandle, (uint32_t *)(
NOR_DEVICE_ADDR + uwStartAddress), pData);
00238
00239     /* Read NOR device status */
00240     if(HAL_NOR_GetStatus(&norHandle, NOR_DEV
ICE_ADDR, PROGRAM_TIMEOUT) != NOR_SUCCESS)
00241     {
00242       return NOR_STATUS_ERROR;
00243     }
00244
00245     /* Update the counters */
00246     index--;
00247     uwStartAddress += 2;
00248     pData++;
00249   }
00250
00251   return NOR_STATUS_OK;
00252 }
00253
00254 /**
00255   * @brief  Programs an amount of data to th
e NOR device.
00256   * @param  uwStartAddress: Write start addr
ess
00257   * @param  pData: Pointer to data to be wri
tten
```

```
00258   * @param  uwDataSize: Size of data to writ
e
00259   * @retval NOR memory status
00260   */
00261 uint8_t BSP_NOR_ProgramData(uint32_t uwStart
Address, uint16_t* pData, uint32_t uwDataSize)
00262 {
00263   /* Send NOR program buffer operation */
00264   HAL_NOR_ProgramBuffer(&norHandle, uwStartA
ddress, pData, uwDataSize);
00265
00266   /* Return the NOR memory status */
00267   if(HAL_NOR_GetStatus(&norHandle, NOR_DEVIC
E_ADDR, PROGRAM_TIMEOUT) != NOR_SUCCESS)
00268   {
00269     return NOR_STATUS_ERROR;
00270   }
00271   else
00272   {
00273     return NOR_STATUS_OK;
00274   }
00275 }
00276
00277 /**
00278   * @brief  Erases the specified block of th
e NOR device.
00279   * @param  BlockAddress: Block address to e
rase
00280   * @retval NOR memory status
00281   */
00282 uint8_t BSP_NOR_Erase_Block(uint32_t BlockAd
dress)
00283 {
00284   /* Send NOR erase block operation */
00285   HAL_NOR_Erase_Block(&norHandle, BlockAddre
ss, NOR_DEVICE_ADDR);
00286
```

```
00287   /* Return the NOR memory status */
00288   if(HAL_NOR_GetStatus(&norHandle, NOR_DEVIC
E_ADDR, BLOCKERASE_TIMEOUT) != NOR_SUCCESS)
00289   {
00290     return NOR_STATUS_ERROR;
00291   }
00292   else
00293   {
00294     return NOR_STATUS_OK;
00295   }
00296 }
00297
00298 /**
00299   * @brief  Erases the entire NOR chip.
00300   * @retval NOR memory status
00301   */
00302 uint8_t BSP_NOR_Erase_Chip(void)
00303 {
00304   /* Send NOR Erase chip operation */
00305   HAL_NOR_Erase_Chip(&norHandle, NOR_DEVICE_
ADDR);
00306
00307   /* Return the NOR memory status */
00308   if(HAL_NOR_GetStatus(&norHandle, NOR_DEVIC
E_ADDR, CHIPERASE_TIMEOUT) != NOR_SUCCESS)
00309   {
00310     return NOR_STATUS_ERROR;
00311   }
00312   else
00313   {
00314     return NOR_STATUS_OK;
00315   }
00316 }
00317
00318 /**
00319   * @brief  Reads NOR flash IDs.
00320   * @param  pNOR_ID : Pointer to NOR ID stru
```

```
cture
00321   * @retval NOR memory status
00322   */
00323 uint8_t BSP_NOR_Read_ID(NOR_IDTypeDef *pNOR_
ID)
00324 {
00325   if(HAL_NOR_Read_ID(&norHandle, pNOR_ID) !=
 HAL_OK)
00326   {
00327     return NOR_STATUS_ERROR;
00328   }
00329   else
00330   {
00331     return NOR_STATUS_OK;
00332   }
00333 }
00334
00335 /**
00336   * @brief  Initializes the NOR MSP.
00337   */
00338 static void NOR_MspInit(void)
00339 {
00340   GPIO_InitTypeDef GPIO_Init_Structure;
00341
00342   /* Enable FMC clock */
00343   __FMC_CLK_ENABLE();
00344
00345   /* Enable GPIOs clock */
00346   __GPIOD_CLK_ENABLE();
00347   __GPIOE_CLK_ENABLE();
00348   __GPIOF_CLK_ENABLE();
00349   __GPIOG_CLK_ENABLE();
00350
00351   /* Common GPIO configuration */
00352   GPIO_Init_Structure.Mode      = GPIO_MODE_
AF_PP;
00353   GPIO_Init_Structure.Pull      = GPIO_PULLU
```

```
P;
00354   GPIO_Init_Structure.Speed     = GPIO_SPEED
_HIGH;
00355   GPIO_Init_Structure.Alternate = GPIO_AF12_
FMC;
00356
00357   /* GPIOD configuration */
00358   GPIO_Init_Structure.Pin   = GPIO_PIN_0 | G
PIO_PIN_1 | GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6
 |\
00359                               GPIO_PIN_7 | G
PIO_PIN_8 | GPIO_PIN_9 | GPIO_PIN_10 | GPIO_PIN_11
 |\
00360                               GPIO_PIN_12 |
GPIO_PIN_13 | GPIO_PIN_14 | GPIO_PIN_15;
00361   HAL_GPIO_Init(GPIOD, &GPIO_Init_Structure)
;
00362
00363   /* GPIOE configuration */
00364   GPIO_Init_Structure.Pin   = GPIO_PIN_2 | G
PIO_PIN_3 | GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6
 |\
00365                               GPIO_PIN_7 | G
PIO_PIN_8 | GPIO_PIN_9 | GPIO_PIN_10 | GPIO_PIN_11
 |\
00366                               GPIO_PIN_12 |
GPIO_PIN_13 | GPIO_PIN_14 | GPIO_PIN_15;
00367   HAL_GPIO_Init(GPIOE, &GPIO_Init_Structure)
;
00368
00369   /* GPIOF configuration */
00370   GPIO_Init_Structure.Pin   = GPIO_PIN_0 | G
PIO_PIN_1 | GPIO_PIN_2| GPIO_PIN_3 | GPIO_PIN_4
  |\
00371                               GPIO_PIN_5 | G
PIO_PIN_12 | GPIO_PIN_13 | GPIO_PIN_14 | GPIO_PIN_
15;
```

```
00372   HAL_GPIO_Init(GPIOF, &GPIO_Init_Structure)
;
00373
00374   /* GPIOG configuration */
00375   GPIO_Init_Structure.Pin   = GPIO_PIN_0 | G
PIO_PIN_1 | GPIO_PIN_2| GPIO_PIN_3 | GPIO_PIN_4
  |\
00376                                GPIO_PIN_5;
00377   HAL_GPIO_Init(GPIOG, &GPIO_Init_Structure)
;
00378 }
00379
00380 /**
00381   * @brief  NOR BSP Wait for Ready/Busy sign
al.
00382   * @param  hnor: Pointer to NOR handle
00383   * @param  Timeout: Timeout duration
00384   */
00385 void HAL_NOR_MspWait(NOR_HandleTypeDef *hnor
, uint32_t Timeout)
00386 {
00387   uint32_t timeout = Timeout;
00388
00389   /* Polling on Ready/Busy signal */
00390   while((HAL_GPIO_ReadPin(NOR_READY_BUSY_GPIO
, NOR_READY_BUSY_PIN) != NOR_BUSY_STATE) && (timeo
ut > 0))
00391   {
00392     timeout--;
00393   }
00394
00395   timeout = Timeout;
00396
00397   /* Polling on Ready/Busy signal */
00398   while((HAL_GPIO_ReadPin(NOR_READY_BUSY_GPIO
, NOR_READY_BUSY_PIN) != NOR_READY_STATE) && (time
out > 0))
```

```
00399   {
00400     timeout--;
00401   }
00402 }
00403
00404 /**
00405   * @}
00406   */
00407
00408 /**
00409   * @}
00410   */
00411
00412 /**
00413   * @}
00414   */
00415
00416 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_sd.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval_sd.h
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm324x9i_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 __STM324x9I_EVAL_SD_H
00041 #define __STM324x9I_EVAL_SD_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------------------------------------------*/
00048 #include "stm32f4xx_hal.h"
00049 #include "stm324x9i_eval_io.h"
00050
00051 /** @addtogroup BSP
00052   * @{
00053   */
00054
00055 /** @addtogroup STM324x9I_EVAL
00056   * @{
00057   */
00058
00059 /** @addtogroup STM324x9I_EVAL_SD
00060   * @{
00061   */
00062
00063 /** @defgroup STM324x9I_EVAL_SD_Exported_Types STM324x9I 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          0x00
00079 #define   MSD_ERROR       0x01
00080
00081 /** @defgroup STM324x9I_EVAL_SD_Exported_Con
stants STM324x9I EVAL SD Exported Constants
00082   * @{
00083   */
00084 #define SD_PRESENT               ((uint8_t)0
x01)
00085 #define SD_NOT_PRESENT           ((uint8_t)0
x00)
00086
00087 #define SD_DATATIMEOUT           ((uint32_t)
100000000)
00088
00089 /* DMA definitions for SD DMA transfer */
00090 #define __DMAx_TxRx_CLK_ENABLE             __
DMA2_CLK_ENABLE
00091 #define SD_DMAx_Tx_CHANNEL                 DM
A_CHANNEL_4
00092 #define SD_DMAx_Rx_CHANNEL                 DM
A_CHANNEL_4
00093 #define SD_DMAx_Tx_STREAM                  DM
A2_Stream6
00094 #define SD_DMAx_Rx_STREAM                  DM
A2_Stream3
00095 #define SD_DMAx_Tx_IRQn                    DM
A2_Stream6_IRQn
```

```
00096 #define SD_DMAx_Rx_IRQn                  DM
A2_Stream3_IRQn
00097 #define SD_DMAx_Tx_IRQHandler            DM
A2_Stream6_IRQHandler
00098 #define SD_DMAx_Rx_IRQHandler            DM
A2_Stream3_IRQHandler
00099 #define SD_DetectIRQHandler()            HA
L_GPIO_EXTI_IRQHandler(GPIO_PIN_8)
00100 /**
00101   * @}
00102   */
00103
00104 /** @defgroup STM324x9I_EVAL_SD_Exported_Mac
ro STM324x9I EVAL SD Exported Macro
00105   * @{
00106   */
00107 /**
00108   * @}
00109   */
00110
00111 /** @defgroup STM324x9I_EVAL_SD_Exported_Fun
ctions STM324x9I EVAL SD Exported Functions
00112   * @{
00113   */
00114 uint8_t BSP_SD_Init(void);
00115 uint8_t BSP_SD_ITConfig(void);
00116 void    BSP_SD_DetectIT(void);
00117 void    BSP_SD_DetectCallback(void);
00118 uint8_t BSP_SD_ReadBlocks(uint32_t *pData, u
int64_t ReadAddr, uint32_t BlockSize, uint32_t Num
OfBlocks);
00119 uint8_t BSP_SD_WriteBlocks(uint32_t *pData,
uint64_t WriteAddr, uint32_t BlockSize, uint32_t N
umOfBlocks);
00120 uint8_t BSP_SD_ReadBlocks_DMA(uint32_t *pDat
a, uint64_t ReadAddr, uint32_t BlockSize, uint32_t
 NumOfBlocks);
```

```
00121 uint8_t BSP_SD_WriteBlocks_DMA(uint32_t *pDa
ta, uint64_t WriteAddr, uint32_t BlockSize, uint32
_t NumOfBlocks);
00122 uint8_t BSP_SD_Erase(uint64_t StartAddr, uin
t64_t EndAddr);
00123 void    BSP_SD_IRQHandler(void);
00124 void    BSP_SD_DMA_Tx_IRQHandler(void);
00125 void    BSP_SD_DMA_Rx_IRQHandler(void);
00126 HAL_SD_TransferStateTypedef BSP_SD_GetStatus(
void);
00127 void    BSP_SD_GetCardInfo(HAL_SD_CardInfoTy
pedef *CardInfo);
00128 uint8_t BSP_SD_IsDetected(void);
00129
00130 /**
00131   * @}
00132   */
00133
00134 /**
00135   * @}
00136   */
00137
00138 /**
00139   * @}
00140   */
00141
00142 /**
00143   * @}
00144   */
00145
00146 #ifdef __cplusplus
00147 }
00148 #endif
00149
00150 #endif /* __STM324x9I_EVAL_SD_H */
00151
00152 /************************ (C) COPYRIGHT STMi
```

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

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_sd.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval_sd.c
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file includes the uSD card driver mounted on STM324x9I-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 STM324x9I-EVAL
00044      evaluation board.
00045    - This driver does not need a specific component 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 the BSP_SD_Init() function. This
00052        function includes the MSP layer hardware resources initialization and the
00053        SDIO interface configuration to interface with the external micro SD. It
00054        also includes the micro SD initialization 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 mode is desired, you must configure the
00058        SD detection interrupt mode by calling 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 "stm324x9i_eval_sd.h"
00084
00085 /** @addtogroup BSP
00086   * @{
00087   */
00088
00089 /** @addtogroup STM324x9I_EVAL
00090   * @{
00091   */
00092
00093 /** @defgroup STM324x9I_EVAL_SD STM324x9I EV
AL SD
00094   * @{
00095   */
00096
00097
00098 /** @defgroup STM324x9I_EVAL_SD_Private_Type
sDefinitions STM324x9I EVAL SD Private TypesDefini
tions
00099   * @{
00100   */
00101 /**
00102   * @}
00103   */
00104
00105 /** @defgroup STM324x9I_EVAL_SD_Private_Defi
nes STM324x9I EVAL SD Private Defines
00106   * @{
00107   */
00108 /**
00109   * @}
00110   */
```

```
00111
00112 /** @defgroup STM324x9I_EVAL_SD_Private_Macros STM324x9I EVAL SD Private Macros
00113   * @{
00114   */
00115 /**
00116   * @}
00117   */
00118
00119 /** @defgroup STM324x9I_EVAL_SD_Private_Variables STM324x9I EVAL SD Private Variables
00120   * @{
00121   */
00122 static SD_HandleTypeDef uSdHandle;
00123 static SD_CardInfo uSdCardInfo;
00124 /**
00125   * @}
00126   */
00127
00128 /** @defgroup STM324x9I_EVAL_SD_Private_FunctionPrototypes STM324x9I EVAL SD Private FunctionPrototypes
00129   * @{
00130   */
00131 static void SD_MspInit(void);
00132 /**
00133   * @}
00134   */
00135
00136 /** @defgroup STM324x9I_EVAL_SD_Private_Functions STM324x9I EVAL SD Private Functions
00137   * @{
00138   */
00139
00140 /**
00141   * @brief  Initializes the SD card device.
00142   * @retval SD status
```

```
00143   */
00144 uint8_t BSP_SD_Init(void)
00145 {
00146   uint8_t SD_state = MSD_OK;
00147
00148   /* uSD device interface configuration */
00149   uSdHandle.Instance = SDIO;
00150
00151   uSdHandle.Init.ClockEdge           = SDIO_
CLOCK_EDGE_RISING;
00152   uSdHandle.Init.ClockBypass         = SDIO_
CLOCK_BYPASS_DISABLE;
00153   uSdHandle.Init.ClockPowerSave      = SDIO_
CLOCK_POWER_SAVE_DISABLE;
00154   uSdHandle.Init.BusWide             = SDIO_
BUS_WIDE_1B;
00155   uSdHandle.Init.HardwareFlowControl = SDIO_
HARDWARE_FLOW_CONTROL_DISABLE;
00156   uSdHandle.Init.ClockDiv            = SDIO_
TRANSFER_CLK_DIV;
00157
00158   /* Configure IO functionalities for SD det
ect pin */
00159   BSP_IO_Init();
00160
00161   /* Check if the SD card is plugged in the
slot */
00162   if(BSP_SD_IsDetected() != SD_PRESENT)
00163   {
00164     return MSD_ERROR;
00165   }
00166
00167   /* HAL SD initialization */
00168   SD_MspInit();
00169   if(HAL_SD_Init(&uSdHandle, &uSdCardInfo) !
= SD_OK)
00170   {
```

```
00171     SD_state = MSD_ERROR;
00172   }
00173
00174   /* Configure SD Bus width */
00175   if(SD_state == MSD_OK)
00176   {
00177     /* Enable wide operation */
00178     if(HAL_SD_WideBusOperation_Config(&uSdHa
ndle, SDIO_BUS_WIDE_4B) != SD_OK)
00179     {
00180       SD_state = MSD_ERROR;
00181     }
00182     else
00183     {
00184       SD_state = MSD_OK;
00185     }
00186   }
00187
00188   return  SD_state;
00189 }
00190
00191 /**
00192   * @brief  Configures Interrupt mode for SD
 detection pin.
00193   * @retval Returns 0
00194   */
00195 uint8_t BSP_SD_ITConfig(void)
00196 {
00197   /* Configure Interrupt mode for SD detecti
on pin */
00198   BSP_IO_ConfigPin(SD_DETECT_PIN, IO_MODE_IT
_FALLING_EDGE);
00199
00200   return 0;
00201 }
00202
00203 /**
```

```
00204  * @brief  Detects if SD card is correctly p
lugged in the memory slot or not.
00205  * @retval Returns if SD is detected or not
00206  */
00207 uint8_t BSP_SD_IsDetected(void)
00208 {
00209   __IO uint8_t status = SD_PRESENT;
00210
00211   /* Check SD card detect pin */
00212   if(BSP_IO_ReadPin(SD_DETECT_PIN))
00213   {
00214     status = SD_NOT_PRESENT;
00215   }
00216
00217   return status;
00218 }
00219
00220 /** @brief  SD detect IT treatment.
00221   */
00222 void BSP_SD_DetectIT(void)
00223 {
00224   /* Clear all pending bits */
00225   BSP_IO_ITClear();
00226
00227   /* To re-enable IT */
00228   BSP_SD_ITConfig();
00229
00230   /* SD detect IT callback */
00231   BSP_SD_DetectCallback();
00232 }
00233
00234 /** @brief  SD detect IT detection callback
00235   */
00236 __weak void BSP_SD_DetectCallback(void)
00237 {
00238   /* NOTE: This function Should not be modif
ied, when the callback is needed,
```

```
00239       the BSP_SD_DetectCallback could be impl
emented in the user file
00240     */
00241 }
00242
00243 /**
00244     * @brief  Reads block(s) from a specified
address in an SD card, in polling mode.
00245     * @param  pData: Pointer to the buffer tha
t will contain the data to transmit
00246     * @param  ReadAddr: Address from where dat
a is to be read
00247     * @param  BlockSize: SD card data block si
ze, that should be 512
00248     * @param  NumOfBlocks: Number of SD blocks
 to read
00249     * @retval SD status
00250     */
00251 uint8_t BSP_SD_ReadBlocks(uint32_t *pData, u
int64_t ReadAddr, uint32_t BlockSize, uint32_t Num
OfBlocks)
00252 {
00253     if(HAL_SD_ReadBlocks(&uSdHandle, pData, Re
adAddr, BlockSize, NumOfBlocks) != SD_OK)
00254     {
00255       return MSD_ERROR;
00256     }
00257     else
00258     {
00259       return MSD_OK;
00260     }
00261 }
00262
00263 /**
00264     * @brief  Writes block(s) to a specified a
ddress in an SD card, in polling mode.
00265     * @param  pData: Pointer to the buffer tha
```

```
t will contain the data to transmit
00266   * @param  WriteAddr: Address from where da
ta is to be written
00267   * @param  BlockSize: SD card data block si
ze, that should be 512
00268   * @param  NumOfBlocks: Number of SD blocks
 to write
00269   * @retval SD status
00270   */
00271 uint8_t BSP_SD_WriteBlocks(uint32_t *pData,
uint64_t WriteAddr, uint32_t BlockSize, uint32_t N
umOfBlocks)
00272 {
00273   if(HAL_SD_WriteBlocks(&uSdHandle, pData, W
riteAddr, BlockSize, NumOfBlocks) != SD_OK)
00274   {
00275     return MSD_ERROR;
00276   }
00277   else
00278   {
00279     return MSD_OK;
00280   }
00281 }
00282
00283 /**
00284   * @brief  Reads block(s) from a specified
address in an SD card, in DMA mode.
00285   * @param  pData: Pointer to the buffer tha
t will contain the data to transmit
00286   * @param  ReadAddr: Address from where dat
a is to be read
00287   * @param  BlockSize: SD card data block si
ze, that should be 512
00288   * @param  NumOfBlocks: Number of SD blocks
 to read
00289   * @retval SD status
00290   */
```

```
00291 uint8_t BSP_SD_ReadBlocks_DMA(uint32_t *pDat
a, uint64_t ReadAddr, uint32_t BlockSize, uint32_t
 NumOfBlocks)
00292 {
00293   uint8_t SD_state = MSD_OK;
00294
00295   /* Read block(s) in DMA transfer mode */
00296   if(HAL_SD_ReadBlocks_DMA(&uSdHandle, pData
, ReadAddr, BlockSize, NumOfBlocks) != SD_OK)
00297   {
00298     SD_state = MSD_ERROR;
00299   }
00300
00301   /* Wait until transfer is complete */
00302   if(SD_state == MSD_OK)
00303   {
00304     if(HAL_SD_CheckReadOperation(&uSdHandle,
 (uint32_t)SD_DATATIMEOUT) != SD_OK)
00305     {
00306       SD_state = MSD_ERROR;
00307     }
00308     else
00309     {
00310       SD_state = MSD_OK;
00311     }
00312   }
00313
00314   return SD_state;
00315 }
00316
00317 /**
00318   * @brief  Writes block(s) to a specified a
ddress in an SD card, in DMA mode.
00319   * @param  pData: Pointer to the buffer tha
t will contain the data to transmit
00320   * @param  WriteAddr: Address from where da
ta is to be written
```

```
00321   * @param  BlockSize: SD card data block si
ze, that should be 512
00322   * @param  NumOfBlocks: Number of SD blocks
 to write
00323   * @retval SD status
00324   */
00325 uint8_t BSP_SD_WriteBlocks_DMA(uint32_t *pDa
ta, uint64_t WriteAddr, uint32_t BlockSize, uint32
_t NumOfBlocks)
00326 {
00327   uint8_t SD_state = MSD_OK;
00328
00329   /* Write block(s) in DMA transfer mode */
00330   if(HAL_SD_WriteBlocks_DMA(&uSdHandle, pDat
a, WriteAddr, BlockSize, NumOfBlocks) != SD_OK)
00331   {
00332     SD_state = MSD_ERROR;
00333   }
00334
00335   /* Wait until transfer is complete */
00336   if(SD_state == MSD_OK)
00337   {
00338     if(HAL_SD_CheckWriteOperation(&uSdHandle
, (uint32_t)SD_DATATIMEOUT) != SD_OK)
00339     {
00340       SD_state = MSD_ERROR;
00341     }
00342     else
00343     {
00344       SD_state = MSD_OK;
00345     }
00346   }
00347
00348   return SD_state;
00349 }
00350
00351 /**
```

```
00352   * @brief  Erases the specified memory area
 of the given SD card.
00353   * @param  StartAddr: Start byte address
00354   * @param  EndAddr: End byte address
00355   * @retval SD status
00356   */
00357 uint8_t BSP_SD_Erase(uint64_t StartAddr, uin
t64_t EndAddr)
00358 {
00359   if(HAL_SD_Erase(&uSdHandle, StartAddr, End
Addr) != SD_OK)
00360   {
00361     return MSD_ERROR;
00362   }
00363   else
00364   {
00365     return MSD_OK;
00366   }
00367 }
00368
00369 /**
00370   * @brief  Initializes the SD MSP.
00371   */
00372 static void SD_MspInit(void)
00373 {
00374   static DMA_HandleTypeDef dmaRxHandle;
00375   static DMA_HandleTypeDef dmaTxHandle;
00376   GPIO_InitTypeDef GPIO_Init_Structure;
00377   SD_HandleTypeDef *hsd = &uSdHandle;
00378
00379   /* Enable SDIO clock */
00380   __SDIO_CLK_ENABLE();
00381
00382   /* Enable DMA2 clocks */
00383   __DMAx_TxRx_CLK_ENABLE();
00384
00385   /* Enable GPIOs clock */
```

```
00386   __GPIOC_CLK_ENABLE();
00387   __GPIOD_CLK_ENABLE();
00388
00389   /* Common GPIO configuration */
00390   GPIO_Init_Structure.Mode      = GPIO_MODE_
AF_PP;
00391   GPIO_Init_Structure.Pull      = GPIO_PULLU
P;
00392   GPIO_Init_Structure.Speed     = GPIO_SPEED
_HIGH;
00393   GPIO_Init_Structure.Alternate = GPIO_AF12_
SDIO;
00394
00395   /* GPIOC configuration */
00396   GPIO_Init_Structure.Pin = GPIO_PIN_8 | GPI
O_PIN_9 | GPIO_PIN_10 | GPIO_PIN_11 | GPIO_PIN_12;
00397
00398   HAL_GPIO_Init(GPIOC, &GPIO_Init_Structure)
;
00399
00400   /* GPIOD configuration */
00401   GPIO_Init_Structure.Pin = GPIO_PIN_2;
00402   HAL_GPIO_Init(GPIOD, &GPIO_Init_Structure)
;
00403
00404   /* NVIC configuration for SDIO interrupts
*/
00405   HAL_NVIC_SetPriority(SDIO_IRQn, 5, 0);
00406   HAL_NVIC_EnableIRQ(SDIO_IRQn);
00407
00408   /* Configure DMA Rx parameters */
00409   dmaRxHandle.Init.Channel             = SD_
DMAx_Rx_CHANNEL;
00410   dmaRxHandle.Init.Direction           = DMA
_PERIPH_TO_MEMORY;
00411   dmaRxHandle.Init.PeriphInc           = DMA
_PINC_DISABLE;
```

```
00412   dmaRxHandle.Init.MemInc                = DMA
_MINC_ENABLE;
00413   dmaRxHandle.Init.PeriphDataAlignment = DMA
_PDATAALIGN_WORD;
00414   dmaRxHandle.Init.MemDataAlignment    = DMA
_MDATAALIGN_WORD;
00415   dmaRxHandle.Init.Mode                  = DMA
_PFCTRL;
00416   dmaRxHandle.Init.Priority              = DMA
_PRIORITY_VERY_HIGH;
00417   dmaRxHandle.Init.FIFOMode              = DMA
_FIFOMODE_ENABLE;
00418   dmaRxHandle.Init.FIFOThreshold         = DMA
_FIFO_THRESHOLD_FULL;
00419   dmaRxHandle.Init.MemBurst              = DMA
_MBURST_INC4;
00420   dmaRxHandle.Init.PeriphBurst           = DMA
_PBURST_INC4;
00421
00422   dmaRxHandle.Instance = SD_DMAx_Rx_STREAM;
00423
00424   /* Associate the DMA handle */
00425   __HAL_LINKDMA(hsd, hdmarx, dmaRxHandle);
00426
00427   /* Deinitialize the stream for new transfe
r */
00428   HAL_DMA_DeInit(&dmaRxHandle);
00429
00430   /* Configure the DMA stream */
00431   HAL_DMA_Init(&dmaRxHandle);
00432
00433   /* Configure DMA Tx parameters */
00434   dmaTxHandle.Init.Channel               = SD_
DMAx_Tx_CHANNEL;
00435   dmaTxHandle.Init.Direction             = DMA
_MEMORY_TO_PERIPH;
00436   dmaTxHandle.Init.PeriphInc             = DMA
```

```
_PINC_DISABLE;
00437   dmaTxHandle.Init.MemInc              = DMA
_MINC_ENABLE;
00438   dmaTxHandle.Init.PeriphDataAlignment = DMA
_PDATAALIGN_WORD;
00439   dmaTxHandle.Init.MemDataAlignment    = DMA
_MDATAALIGN_WORD;
00440   dmaTxHandle.Init.Mode                = DMA
_PFCTRL;
00441   dmaTxHandle.Init.Priority            = DMA
_PRIORITY_VERY_HIGH;
00442   dmaTxHandle.Init.FIFOMode            = DMA
_FIFOMODE_ENABLE;
00443   dmaTxHandle.Init.FIFOThreshold       = DMA
_FIFO_THRESHOLD_FULL;
00444   dmaTxHandle.Init.MemBurst            = DMA
_MBURST_INC4;
00445   dmaTxHandle.Init.PeriphBurst         = DMA
_PBURST_INC4;
00446
00447   dmaTxHandle.Instance = SD_DMAx_Tx_STREAM;
00448
00449   /* Associate the DMA handle */
00450   __HAL_LINKDMA(hsd, hdmatx, dmaTxHandle);
00451
00452   /* Deinitialize the stream for new transfe
r */
00453   HAL_DMA_DeInit(&dmaTxHandle);
00454
00455   /* Configure the DMA stream */
00456   HAL_DMA_Init(&dmaTxHandle);
00457
00458   /* NVIC configuration for DMA transfer com
plete interrupt */
00459   HAL_NVIC_SetPriority(SD_DMAx_Rx_IRQn, 6, 0
);
00460   HAL_NVIC_EnableIRQ(SD_DMAx_Rx_IRQn);
```

```
00461
00462   /* NVIC configuration for DMA transfer com
plete interrupt */
00463   HAL_NVIC_SetPriority(SD_DMAx_Tx_IRQn, 6, 0
);
00464   HAL_NVIC_EnableIRQ(SD_DMAx_Tx_IRQn);
00465 }
00466
00467 /**
00468   * @brief  Handles SD card interrupt reques
t.
00469   */
00470 void BSP_SD_IRQHandler(void)
00471 {
00472   HAL_SD_IRQHandler(&uSdHandle);
00473 }
00474
00475 /**
00476   * @brief  Handles SD DMA Tx transfer inter
rupt request.
00477   */
00478 void BSP_SD_DMA_Tx_IRQHandler(void)
00479 {
00480   HAL_DMA_IRQHandler(uSdHandle.hdmatx);
00481 }
00482
00483 /**
00484   * @brief  Handles SD DMA Rx transfer inter
rupt request.
00485   */
00486 void BSP_SD_DMA_Rx_IRQHandler(void)
00487 {
00488   HAL_DMA_IRQHandler(uSdHandle.hdmarx);
00489 }
00490
00491 /**
00492   * @brief  Gets the current SD card data st
```

```
atus.
00493   * @retval Data transfer state.
00494   *          This value can be one of the fo
llowing values:
00495   *            @arg  SD_TRANSFER_OK: No data
 transfer is acting
00496   *            @arg  SD_TRANSFER_BUSY: Data
transfer is acting
00497   *            @arg  SD_TRANSFER_ERROR: Data
 transfer error
00498   */
00499 HAL_SD_TransferStateTypedef BSP_SD_GetStatus(
void)
00500 {
00501   return(HAL_SD_GetStatus(&uSdHandle));
00502 }
00503
00504 /**
00505   * @brief  Get SD information about specifi
c SD card.
00506   * @param  CardInfo: Pointer to HAL_SD_Card
InfoTypedef structure
00507   */
00508 void BSP_SD_GetCardInfo(HAL_SD_CardInfoTyped
ef *CardInfo)
00509 {
00510   /* Get SD card Information */
00511   HAL_SD_Get_CardInfo(&uSdHandle, CardInfo);
00512 }
00513
00514 /**
00515   * @}
00516   */
00517
00518 /**
00519   * @}
00520   */
```

```
00521
00522 /**
00523   * @}
00524   */
00525
00526 /**
00527   * @}
00528   */
00529
00530 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_sdram.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval_sdram.c
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file includes the SDRAM driver for the MT48LC4M32B2B5-7 memory
00008   *          device mounted on STM324x9I-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 STM324x9I-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 32-b
it width transaction, the DMA transfer
00067         configuration is fixed at single (no
burst) word transfer (see the
00068         SDRAM_MspInit() static 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 "stm324x9i_eval_sdram.h"
00082
```

```
00083 /** @addtogroup BSP
00084   * @{
00085   */
00086
00087 /** @addtogroup STM324x9I_EVAL
00088   * @{
00089   */
00090
00091 /** @defgroup STM324x9I_EVAL_SDRAM STM324x9I
 EVAL SDRAM
00092   * @{
00093   */
00094
00095 /** @defgroup STM324x9I_EVAL_SDRAM_Private_T
ypes_Definitions STM324x9I EVAL SDRAM Private Type
s Definitions
00096   * @{
00097   */
00098 /**
00099   * @}
00100   */
00101
00102 /** @defgroup STM324x9I_EVAL_SDRAM_Private_D
efines STM324x9I EVAL SDRAM Private Defines
00103   * @{
00104   */
00105 /**
00106   * @}
00107   */
00108
00109 /** @defgroup STM324x9I_EVAL_SDRAM_Private_M
acros STM324x9I EVAL SDRAM Private Macros
00110   * @{
00111   */
00112 /**
00113   * @}
00114   */
```

```
00115
00116 /** @defgroup STM324x9I_EVAL_SDRAM_Private_V
ariables STM324x9I 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 STM324x9I_EVAL_SDRAM_Private_F
unction_Prototypes STM324x9I EVAL SDRAM Private Fu
nction Prototypes
00127   * @{
00128   */
00129 static void SDRAM_MspInit(void);
00130 /**
00131   * @}
00132   */
00133
00134 /** @defgroup STM324x9I_EVAL_SDRAM_Private_F
unctions STM324x9I EVAL SDRAM Private Functions
00135   * @{
00136   */
00137
00138 /**
00139   * @brief  Initializes the SDRAM device.
00140   * @retval SDRAM status
00141   */
00142 uint8_t BSP_SDRAM_Init(void)
00143 {
00144   static uint8_t sdramstatus = SDRAM_ERROR;
00145   /* SDRAM device configuration */
00146   sdramHandle.Instance = FMC_SDRAM_DEVICE;
00147
```

```
00148   /* Timing configuration for 90Mhz as SD cl
ock frequency (System clock is up to 180Mhz */
00149   Timing.LoadToActiveDelay    = 2;
00150   Timing.ExitSelfRefreshDelay = 7;
00151   Timing.SelfRefreshTime      = 4;
00152   Timing.RowCycleDelay        = 7;
00153   Timing.WriteRecoveryTime    = 2;
00154   Timing.RPDelay              = 2;
00155   Timing.RCDDelay             = 2;
00156
00157   sdramHandle.Init.SDBank             = FMC_
SDRAM_BANK1;
00158   sdramHandle.Init.ColumnBitsNumber   = FMC_
SDRAM_COLUMN_BITS_NUM_9;
00159   sdramHandle.Init.RowBitsNumber      = FMC_
SDRAM_ROW_BITS_NUM_12;
00160   sdramHandle.Init.MemoryDataWidth    = SDRA
M_MEMORY_WIDTH;
00161   sdramHandle.Init.InternalBankNumber = FMC_
SDRAM_INTERN_BANKS_NUM_4;
00162   sdramHandle.Init.CASLatency         = FMC_
SDRAM_CAS_LATENCY_3;
00163   sdramHandle.Init.WriteProtection    = FMC_
SDRAM_WRITE_PROTECTION_DISABLE;
00164   sdramHandle.Init.SDClockPeriod      = SDCL
OCK_PERIOD;
00165   sdramHandle.Init.ReadBurst          = FMC_
SDRAM_RBURST_ENABLE;
00166   sdramHandle.Init.ReadPipeDelay      = FMC_
SDRAM_RPIPE_DELAY_0;
00167
00168   /* SDRAM controller initialization */
00169   SDRAM_MspInit();
00170   if(HAL_SDRAM_Init(&sdramHandle, &Timing) !
= HAL_OK)
00171   {
00172     sdramstatus = SDRAM_ERROR;
```

```
00173   }
00174   else
00175   {
00176     sdramstatus = SDRAM_OK;
00177   }
00178
00179   /* SDRAM initialization sequence */
00180   BSP_SDRAM_Initialization_sequence(REFRESH_
COUNT);
00181
00182   return sdramstatus;
00183 }
00184
00185 /**
00186   * @brief  Programs the SDRAM device.
00187   * @param  RefreshCount: SDRAM refresh coun
ter value
00188   */
00189 void BSP_SDRAM_Initialization_sequence(uint3
2_t RefreshCount)
00190 {
00191   __IO uint32_t tmpmrd = 0;
00192
00193   /* Step 1: Configure a clock configuration
 enable command */
00194   Command.CommandMode            = FMC_SDRAM
_CMD_CLK_ENABLE;
00195   Command.CommandTarget          = FMC_SDRAM
_CMD_TARGET_BANK1;
00196   Command.AutoRefreshNumber      = 1;
00197   Command.ModeRegisterDefinition = 0;
00198
00199   /* Send the command */
00200   HAL_SDRAM_SendCommand(&sdramHandle, &Comma
nd, SDRAM_TIMEOUT);
00201
00202   /* Step 2: Insert 100 us minimum delay */
```

```
00203   /* Inserted delay is equal to 1 ms due to
systick time base unit (ms) */
00204   HAL_Delay(1);
00205
00206   /* Step 3: Configure a PALL (precharge all
) command */
00207   Command.CommandMode            = FMC_SDRAM
_CMD_PALL;
00208   Command.CommandTarget          = FMC_SDRAM
_CMD_TARGET_BANK1;
00209   Command.AutoRefreshNumber      = 1;
00210   Command.ModeRegisterDefinition = 0;
00211
00212   /* Send the command */
00213   HAL_SDRAM_SendCommand(&sdramHandle, &Comma
nd, SDRAM_TIMEOUT);
00214
00215   /* Step 4: Configure an Auto Refresh comma
nd */
00216   Command.CommandMode            = FMC_SDRAM
_CMD_AUTOREFRESH_MODE;
00217   Command.CommandTarget          = FMC_SDRAM
_CMD_TARGET_BANK1;
00218   Command.AutoRefreshNumber      = 8;
00219   Command.ModeRegisterDefinition = 0;
00220
00221   /* Send the command */
00222   HAL_SDRAM_SendCommand(&sdramHandle, &Comma
nd, SDRAM_TIMEOUT);
00223
00224   /* Step 5: Program the external memory mod
e register */
00225   tmpmrd = (uint32_t)SDRAM_MODEREG_BURST_LEN
GTH_1          |\
00226                      SDRAM_MODEREG_BURST_TYP
E_SEQUENTIAL   |\
00227                      SDRAM_MODEREG_CAS_LATEN
```

```
CY_3            |\
00228                       SDRAM_MODEREG_OPERATING
_MODE_STANDARD |\
00229                       SDRAM_MODEREG_WRITEBURS
T_MODE_SINGLE;
00230
00231   Command.CommandMode            = FMC_SDRAM
_CMD_LOAD_MODE;
00232   Command.CommandTarget          = FMC_SDRAM
_CMD_TARGET_BANK1;
00233   Command.AutoRefreshNumber      = 1;
00234   Command.ModeRegisterDefinition = tmpmrd;
00235
00236   /* Send the command */
00237   HAL_SDRAM_SendCommand(&sdramHandle, &Comma
nd, SDRAM_TIMEOUT);
00238
00239   /* Step 6: Set the refresh rate counter */
00240   /* Set the device refresh rate */
00241   HAL_SDRAM_ProgramRefreshRate(&sdramHandle,
 RefreshCount);
00242 }
00243
00244 /**
00245   * @brief  Reads an mount of data from the
SDRAM memory in polling mode.
00246   * @param  uwStartAddress: Read start addre
ss
00247   * @param  pData: Pointer to data to be rea
d
00248   * @param  uwDataSize: Size of read data fr
om the memory
00249   * @retval SDRAM status
00250   */
00251 uint8_t BSP_SDRAM_ReadData(uint32_t uwStartA
ddress, uint32_t *pData, uint32_t uwDataSize)
00252 {
```

```
00253   if(HAL_SDRAM_Read_32b(&sdramHandle, (uint3
2_t *)uwStartAddress, pData, uwDataSize) != HAL_OK
)
00254   {
00255     return SDRAM_ERROR;
00256   }
00257   else
00258   {
00259     return SDRAM_OK;
00260   }
00261 }
00262
00263 /**
00264   * @brief  Reads an mount of data from the
SDRAM memory in DMA mode.
00265   * @param  uwStartAddress: Read start addre
ss
00266   * @param  pData: Pointer to data to be rea
d
00267   * @param  uwDataSize: Size of read data fr
om the memory
00268   * @retval SDRAM status
00269   */
00270 uint8_t BSP_SDRAM_ReadData_DMA(uint32_t uwSt
artAddress, uint32_t *pData, uint32_t uwDataSize)
00271 {
00272   if(HAL_SDRAM_Read_DMA(&sdramHandle, (uint3
2_t *)uwStartAddress, pData, uwDataSize) != HAL_OK
)
00273   {
00274     return SDRAM_ERROR;
00275   }
00276   else
00277   {
00278     return SDRAM_OK;
00279   }
00280 }
```

```
00281
00282 /**
00283   * @brief  Writes an mount of data to the S
DRAM memory in polling mode.
00284   * @param  uwStartAddress: Write start addr
ess
00285   * @param  pData: Pointer to data to be wri
tten
00286   * @param  uwDataSize: Size of written data
 from the memory
00287   * @retval SDRAM status
00288   */
00289 uint8_t BSP_SDRAM_WriteData(uint32_t uwStart
Address, uint32_t *pData, uint32_t uwDataSize)
00290 {
00291   if(HAL_SDRAM_Write_32b(&sdramHandle, (uint
32_t *)uwStartAddress, pData, uwDataSize) != HAL_O
K)
00292   {
00293     return SDRAM_ERROR;
00294   }
00295   else
00296   {
00297     return SDRAM_OK;
00298   }
00299 }
00300
00301 /**
00302   * @brief  Writes an mount of data to the S
DRAM memory in DMA mode.
00303   * @param  uwStartAddress: Write start addr
ess
00304   * @param  pData: Pointer to data to be wri
tten
00305   * @param  uwDataSize: Size of written data
 from the memory
00306   * @retval SDRAM status
```

```
00307   */
00308 uint8_t BSP_SDRAM_WriteData_DMA(uint32_t uwS
tartAddress, uint32_t *pData, uint32_t uwDataSize)

00309 {
00310   if(HAL_SDRAM_Write_DMA(&sdramHandle, (uint
32_t *)uwStartAddress, pData, uwDataSize) != HAL_O
K)
00311   {
00312     return SDRAM_ERROR;
00313   }
00314   else
00315   {
00316     return SDRAM_OK;
00317   }
00318 }
00319
00320 /**
00321   * @brief  Sends command to the SDRAM bank.
00322   * @param  SdramCmd: Pointer to SDRAM comma
nd structure
00323   * @retval HAL status
00324   */
00325 uint8_t BSP_SDRAM_Sendcmd(FMC_SDRAM_CommandT
ypeDef *SdramCmd)
00326 {
00327   if(HAL_SDRAM_SendCommand(&sdramHandle, Sdr
amCmd, SDRAM_TIMEOUT) != HAL_OK)
00328   {
00329     return SDRAM_ERROR;
00330   }
00331   else
00332   {
00333     return SDRAM_OK;
00334   }
00335 }
00336
```

```
00337 /**
00338   * @brief  Handles SDRAM DMA transfer inter
rupt request.
00339   */
00340 void BSP_SDRAM_DMA_IRQHandler(void)
00341 {
00342   HAL_DMA_IRQHandler(sdramHandle.hdma);
00343 }
00344
00345 /**
00346   * @brief  Initializes SDRAM MSP.
00347   */
00348 static void SDRAM_MspInit(void)
00349 {
00350   static DMA_HandleTypeDef dmaHandle;
00351   GPIO_InitTypeDef GPIO_Init_Structure;
00352   SDRAM_HandleTypeDef  *hsdram = &sdramHandle
;
00353
00354   /* Enable FMC clock */
00355   __FMC_CLK_ENABLE();
00356
00357   /* Enable chosen DMAx clock */
00358   __DMAx_CLK_ENABLE();
00359
00360   /* Enable GPIOs clock */
00361   __GPIOD_CLK_ENABLE();
00362   __GPIOE_CLK_ENABLE();
00363   __GPIOF_CLK_ENABLE();
00364   __GPIOG_CLK_ENABLE();
00365   __GPIOH_CLK_ENABLE();
00366   __GPIOI_CLK_ENABLE();
00367
00368   /* Common GPIO configuration */
00369   GPIO_Init_Structure.Mode      = GPIO_MODE_
AF_PP;
00370   GPIO_Init_Structure.Pull      = GPIO_PULLU
```

```
P;
00371   GPIO_Init_Structure.Speed     = GPIO_SPEED
_FAST;
00372   GPIO_Init_Structure.Alternate = GPIO_AF12_
FMC;
00373
00374   /* GPIOD configuration */
00375   GPIO_Init_Structure.Pin   = GPIO_PIN_0 | G
PIO_PIN_1 | GPIO_PIN_8| GPIO_PIN_9 | GPIO_PIN_10 |
\
00376                               GPIO_PIN_14 |
GPIO_PIN_15;
00377
00378
00379   HAL_GPIO_Init(GPIOD, &GPIO_Init_Structure)
;
00380
00381   /* GPIOE configuration */
00382   GPIO_Init_Structure.Pin   = GPIO_PIN_0 | G
PIO_PIN_1 | GPIO_PIN_7| GPIO_PIN_8 | GPIO_PIN_9 |\
00383                               GPIO_PIN_10 |
GPIO_PIN_11 | GPIO_PIN_12 | GPIO_PIN_13 | GPIO_PIN
_14 |\
00384                               GPIO_PIN_15;
00385
00386   HAL_GPIO_Init(GPIOE, &GPIO_Init_Structure)
;
00387
00388   /* GPIOF configuration */
00389   GPIO_Init_Structure.Pin   = GPIO_PIN_0 | G
PIO_PIN_1 | GPIO_PIN_2| GPIO_PIN_3 | GPIO_PIN_4 |\
00390                               GPIO_PIN_5 | G
PIO_PIN_11 | GPIO_PIN_12 | GPIO_PIN_13 | GPIO_PIN_
14 |\
00391                               GPIO_PIN_15;
00392
00393   HAL_GPIO_Init(GPIOF, &GPIO_Init_Structure)
```

```
;
00394
00395   /* GPIOG configuration */
00396   GPIO_Init_Structure.Pin   = GPIO_PIN_0 | G
PIO_PIN_1 | GPIO_PIN_4| GPIO_PIN_5 | GPIO_PIN_8 |\
00397                               GPIO_PIN_15;
00398   HAL_GPIO_Init(GPIOG, &GPIO_Init_Structure)
;
00399
00400   /* GPIOH configuration */
00401   GPIO_Init_Structure.Pin   = GPIO_PIN_2 | G
PIO_PIN_3 | GPIO_PIN_5 | GPIO_PIN_8 | GPIO_PIN_9 |
\
00402                               GPIO_PIN_10 |
GPIO_PIN_11 | GPIO_PIN_12 | GPIO_PIN_13 | GPIO_PIN
_14 |\
00403                               GPIO_PIN_15;
00404   HAL_GPIO_Init(GPIOH, &GPIO_Init_Structure)
;
00405
00406   /* GPIOI configuration */
00407   GPIO_Init_Structure.Pin   = GPIO_PIN_0 | G
PIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4 |
\
00408                               GPIO_PIN_5 | G
PIO_PIN_6 | GPIO_PIN_7 | GPIO_PIN_9 | GPIO_PIN_10;
00409   HAL_GPIO_Init(GPIOI, &GPIO_Init_Structure)
;
00410
00411   /* Configure common DMA parameters */
00412   dmaHandle.Init.Channel             = SDRAM
_DMAx_CHANNEL;
00413   dmaHandle.Init.Direction           = DMA_M
EMORY_TO_MEMORY;
00414   dmaHandle.Init.PeriphInc           = DMA_P
INC_ENABLE;
00415   dmaHandle.Init.MemInc              = DMA_M
```

```
INC_ENABLE;
00416   dmaHandle.Init.PeriphDataAlignment = DMA_P
DATAALIGN_WORD;
00417   dmaHandle.Init.MemDataAlignment    = DMA_M
DATAALIGN_WORD;
00418   dmaHandle.Init.Mode                = DMA_N
ORMAL;
00419   dmaHandle.Init.Priority            = DMA_P
RIORITY_HIGH;
00420   dmaHandle.Init.FIFOMode            = DMA_F
IFOMODE_DISABLE;
00421   dmaHandle.Init.FIFOThreshold       = DMA_F
IFO_THRESHOLD_FULL;
00422   dmaHandle.Init.MemBurst            = DMA_M
BURST_SINGLE;
00423   dmaHandle.Init.PeriphBurst         = DMA_P
BURST_SINGLE;
00424
00425   dmaHandle.Instance = SDRAM_DMAx_STREAM;
00426
00427    /* Associate the DMA handle */
00428   __HAL_LINKDMA(hsdram, hdma, dmaHandle);
00429
00430   /* Deinitialize the stream for new transfe
r */
00431   HAL_DMA_DeInit(&dmaHandle);
00432
00433   /* Configure the DMA stream */
00434   HAL_DMA_Init(&dmaHandle);
00435
00436   /* NVIC configuration for DMA transfer com
plete interrupt */
00437   HAL_NVIC_SetPriority(SDRAM_DMAx_IRQn, 5, 0
);
00438   HAL_NVIC_EnableIRQ(SDRAM_DMAx_IRQn);
00439 }
00440
```

```
00441 /**
00442   * @}
00443   */
00444
00445 /**
00446   * @}
00447   */
00448
00449 /**
00450   * @}
00451   */
00452
00453 /**
00454   * @}
00455   */
00456
00457 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_sram.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm324x9i_eval_sram.h
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm324x9i_eval_sram.c driver.
00009   ******************************************************************************
00010   * @attention
00011   *
00012   * <h2><center>&copy; COPYRIGHT(c) 2015 STMicroelectronics</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without modification,
00015   * are permitted provided that the following conditions are met:
```

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

```
00035   *
00036   ******************************************
*********************************
00037   */
00038
00039 /* Define to prevent recursive inclusion ---
-----------------------------------*/
00040 #ifndef __STM324x9I_EVAL_SRAM_H
00041 #define __STM324x9I_EVAL_SRAM_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 STM324x9I_EVAL
00055   * @{
00056   */
00057
00058 /** @defgroup STM324x9I_EVAL_SRAM STM324x9I
EVAL SRAM
00059   * @{
00060   */
00061
00062 /** @defgroup STM324x9I_EVAL_SRAM_Exported_T
ypes STM324x9I EVAL SRAM Exported Types
00063   * @{
00064   */
00065 /**
00066   * @}
```

```
00067   */
00068
00069 /** @defgroup STM324x9I_EVAL_SRAM_Exported_C
onstants STM324x9I EVAL SRAM Exported Constants
00070   * @{
00071   */
00072
00073 /**
00074   * @brief  SRAM status structure definition

00075   */
00076 #define   SRAM_OK         0x00
00077 #define   SRAM_ERROR      0x01
00078
00079 #define SRAM_DEVICE_ADDR  ((uint32_t)0x64000
000)
00080 #define SRAM_DEVICE_SIZE  ((uint32_t)0x20000
0)  /* SRAM device size in MBytes */
00081
00082 /* #define SRAM_MEMORY_WIDTH    FMC_NORSRAM_
MEM_BUS_WIDTH_8  */
00083 #define SRAM_MEMORY_WIDTH    FMC_NORSRAM_MEM
_BUS_WIDTH_16
00084
00085 #define SRAM_BURSTACCESS    FMC_BURST_ACCESS
_MODE_DISABLE
00086 /* #define SRAM_BURSTACCESS    FMC_BURST_ACC
ESS_MODE_ENABLE*/
00087
00088 #define SRAM_WRITEBURST    FMC_WRITE_BURST_D
ISABLE
00089 /* #define SRAM_WRITEBURST   FMC_WRITE_BURST
_ENABLE */
00090
00091 #define CONTINUOUSCLOCK_FEATURE    FMC_CONTI
NUOUS_CLOCK_SYNC_ONLY
00092 /* #define CONTINUOUSCLOCK_FEATURE     FMC_C
```

```
ONTINUOUS_CLOCK_SYNC_ASYNC */
00093
00094 /* DMA definitions for SRAM DMA transfer */
00095 #define __SRAM_DMAx_CLK_ENABLE            __
DMA2_CLK_ENABLE
00096 #define SRAM_DMAx_CHANNEL                 DM
A_CHANNEL_0
00097 #define SRAM_DMAx_STREAM                  DM
A2_Stream0
00098 #define SRAM_DMAx_IRQn                    DM
A2_Stream0_IRQn
00099 #define SRAM_DMAx_IRQHandler              DM
A2_Stream0_IRQHandler
00100 /**
00101   * @}
00102   */
00103
00104 /** @defgroup STM324x9I_EVAL_SRAM_Exported_M
acro STM324x9I EVAL SRAM Exported Macro
00105   * @{
00106   */
00107 /**
00108   * @}
00109   */
00110
00111 /** @defgroup STM324x9I_EVAL_SRAM_Exported_F
unctions STM324x9I EVAL SRAM Exported Functions
00112   * @{
00113   */
00114 uint8_t BSP_SRAM_Init(void);
00115 uint8_t BSP_SRAM_ReadData(uint32_t uwStartAd
dress, uint16_t *pData, uint32_t uwDataSize);
00116 uint8_t BSP_SRAM_ReadData_DMA(uint32_t uwSta
rtAddress, uint16_t *pData, uint32_t uwDataSize);
00117 uint8_t BSP_SRAM_WriteData(uint32_t uwStartA
ddress, uint16_t *pData, uint32_t uwDataSize);
00118 uint8_t BSP_SRAM_WriteData_DMA(uint32_t uwSt
```

```
artAddress, uint16_t *pData, uint32_t uwDataSize);
00119 void    BSP_SRAM_DMA_IRQHandler(void);
00120
00121 /**
00122   * @}
00123   */
00124
00125 /**
00126   * @}
00127   */
00128
00129 /**
00130   * @}
00131   */
00132
00133 /**
00134   * @}
00135   */
00136
00137
00138 #ifdef __cplusplus
00139 }
00140 #endif
00141
00142 #endif /* __STM324x9I_EVAL_SRAM_H */
00143
00144 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM324x9I_EVAL BSP User Manual

stm324x9i_eval_sram.c

Go to the documentation of this file.

```
00001 /**
00002   ********************************************************************************
00003   * @file    stm324x9i_eval_sram.c
00004   * @author  MCD Application Team
00005   * @version V2.2.2
00006   * @date    13-January-2016
00007   * @brief   This file includes the SRAM driver for the IS61WV102416BLL-10M memory
00008   *          device mounted on STM324x9I-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 IS61WV
102416BLL-10M SRAM external memory mounted
00044      on STM324x9I-EVAL evaluation board.
00045    - This driver does not need a specific co
mponent driver for the SRAM device
00046      to be included with.
00047
00048 2. Driver description:
00049 ---------------------
00050   + Initialization steps:
00051      o Initialize the SRAM external memory u
sing the BSP_SRAM_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 SRAM memory.
00054
00055   + SRAM read/write operations
00056      o SRAM external memory can be accessed
with read/write operations once it is
00057        initialized.
00058        Read/write operation can be performed
 with AHB access using the functions
00059        BSP_SRAM_ReadData()/BSP_SRAM_WriteDat
a(), or by DMA transfer using the functions
00060        BSP_SRAM_ReadData_DMA()/BSP_SRAM_Writ
```

```
eData_DMA().
00061      o The AHB access is performed with 16-b
it width transaction, the DMA transfer
00062        configuration is fixed at single (no
burst) halfword transfer
00063        (see the SRAM_MspInit() static functi
on).
00064      o User can implement his own functions
for read/write access with his desired
00065        configurations.
00066      o If interrupt mode is used for DMA tra
nsfer, the function BSP_SRAM_DMA_IRQHandler()
00067        is called in IRQ handler file, to ser
ve the generated interrupt once the DMA
00068        transfer is complete.
00069
00070 ------------------------------------------
-----------------------------------*/
00071
00072 /* Includes ----------------------------------
-----------------------------------*/
00073 #include "stm324x9i_eval_sram.h"
00074
00075 /** @addtogroup BSP
00076   * @{
00077   */
00078
00079 /** @addtogroup STM324x9I_EVAL
00080   * @{
00081   */
00082
00083 /** @defgroup STM324x9I_EVAL_SRAM STM324x9I
EVAL SRAM
00084   * @{
00085   */
00086
00087 /** @defgroup STM324x9I_EVAL_SRAM_Private_Ty
```

```
pes_Definitions STM324x9I EVAL SRAM Private Types
Definitions
00088   * @{
00089   */
00090 /**
00091   * @}
00092   */
00093
00094 /** @defgroup STM324x9I_EVAL_SRAM_Private_De
fines STM324x9I EVAL SRAM Private Defines
00095   * @{
00096   */
00097 /**
00098   * @}
00099   */
00100
00101 /** @defgroup STM324x9I_EVAL_SRAM_Private_Ma
cros STM324x9I EVAL SRAM Private Macros
00102   * @{
00103   */
00104 /**
00105   * @}
00106   */
00107
00108 /** @defgroup STM324x9I_EVAL_SRAM_Private_Va
riables STM324x9I EVAL SRAM Private Variables
00109   * @{
00110   */
00111 static SRAM_HandleTypeDef sramHandle;
00112 static FMC_NORSRAM_TimingTypeDef Timing;
00113 /**
00114   * @}
00115   */
00116
00117 /** @defgroup STM324x9I_EVAL_SRAM_Private_Fu
nction_Prototypes STM324x9I EVAL SRAM Private Func
tion Prototypes
```

```
00118   * @{
00119   */
00120 static void SRAM_MspInit(void);
00121 /**
00122   * @}
00123   */
00124
00125 /** @defgroup STM324x9I_EVAL_SRAM_Private_Fu
nctions STM324x9I EVAL SRAM Private Functions
00126   * @{
00127   */
00128
00129 /**
00130   * @brief  Initializes the SRAM device.
00131   * @retval SRAM status
00132   */
00133 uint8_t BSP_SRAM_Init(void)
00134 {
00135   sramHandle.Instance = FMC_NORSRAM_DEVICE;
00136   sramHandle.Extended = FMC_NORSRAM_EXTENDED
_DEVICE;
00137
00138   /* SRAM device configuration */
00139   Timing.AddressSetupTime      = 2;
00140   Timing.AddressHoldTime       = 1;
00141   Timing.DataSetupTime         = 2;
00142   Timing.BusTurnAroundDuration = 1;
00143   Timing.CLKDivision           = 2;
00144   Timing.DataLatency           = 2;
00145   Timing.AccessMode            = FMC_ACCESS_
MODE_A;
00146
00147   sramHandle.Init.NSBank             = FMC_N
ORSRAM_BANK2;
00148   sramHandle.Init.DataAddressMux     = FMC_D
ATA_ADDRESS_MUX_DISABLE;
00149   sramHandle.Init.MemoryType         = FMC_M
```

```
EMORY_TYPE_SRAM;
00150   sramHandle.Init.MemoryDataWidth    = SRAM_
MEMORY_WIDTH;
00151   sramHandle.Init.BurstAccessMode    = SRAM_
BURSTACCESS;
00152   sramHandle.Init.WaitSignalPolarity = FMC_W
AIT_SIGNAL_POLARITY_LOW;
00153   sramHandle.Init.WrapMode           = FMC_W
RAP_MODE_DISABLE;
00154   sramHandle.Init.WaitSignalActive   = FMC_W
AIT_TIMING_BEFORE_WS;
00155   sramHandle.Init.WriteOperation     = FMC_W
RITE_OPERATION_ENABLE;
00156   sramHandle.Init.WaitSignal         = FMC_W
AIT_SIGNAL_DISABLE;
00157   sramHandle.Init.ExtendedMode       = FMC_E
XTENDED_MODE_DISABLE;
00158   sramHandle.Init.AsynchronousWait   = FMC_A
SYNCHRONOUS_WAIT_DISABLE;
00159   sramHandle.Init.WriteBurst         = SRAM_
WRITEBURST;
00160   sramHandle.Init.ContinuousClock    = CONTI
NUOUSCLOCK_FEATURE;
00161
00162   /* SRAM controller initialization */
00163   SRAM_MspInit();
00164   if(HAL_SRAM_Init(&sramHandle, &Timing, &Ti
ming) != HAL_OK)
00165   {
00166     return SRAM_ERROR;
00167   }
00168   else
00169   {
00170     return SRAM_OK;
00171   }
00172 }
00173
```

```
00174 /**
00175   * @brief  Reads an amount of data from the
 SRAM device in polling mode.
00176   * @param  uwStartAddress: Read start addre
ss
00177   * @param  pData: Pointer to data to be read

00178   * @param  uwDataSize: Size of read data fr
om the memory
00179   * @retval SRAM status
00180   */
00181 uint8_t BSP_SRAM_ReadData(uint32_t uwStartAd
dress, uint16_t *pData, uint32_t uwDataSize)
00182 {
00183   if(HAL_SRAM_Read_16b(&sramHandle, (uint32_
t *)uwStartAddress, pData, uwDataSize) != HAL_OK)
00184   {
00185     return SRAM_ERROR;
00186   }
00187   else
00188   {
00189     return SRAM_OK;
00190   }
00191 }
00192
00193 /**
00194   * @brief  Reads an amount of data from the
 SRAM device in DMA mode.
00195   * @param  uwStartAddress: Read start addre
ss
00196   * @param  pData: Pointer to data to be read

00197   * @param  uwDataSize: Size of read data fr
om the memory
00198   * @retval SRAM status
00199   */
00200 uint8_t BSP_SRAM_ReadData_DMA(uint32_t uwSta
```

```
rtAddress, uint16_t *pData, uint32_t uwDataSize)
00201 {
00202   if(HAL_SRAM_Read_DMA(&sramHandle, (uint32_
t *)uwStartAddress, (uint32_t *)pData, uwDataSize)
 != HAL_OK)
00203   {
00204     return SRAM_ERROR;
00205   }
00206   else
00207   {
00208     return SRAM_OK;
00209   }
00210 }
00211
00212 /**
00213   * @brief  Writes an amount of data from th
e SRAM device in polling mode.
00214   * @param  uwStartAddress: Write start addr
ess
00215   * @param  pData: Pointer to data to be wri
tten
00216   * @param  uwDataSize: Size of written data
 from the memory
00217   * @retval SRAM status
00218   */
00219 uint8_t BSP_SRAM_WriteData(uint32_t uwStartA
ddress, uint16_t *pData, uint32_t uwDataSize)
00220 {
00221   if(HAL_SRAM_Write_16b(&sramHandle, (uint32
_t *)uwStartAddress, pData, uwDataSize) != HAL_OK)
00222   {
00223     return SRAM_ERROR;
00224   }
00225   else
00226   {
00227     return SRAM_OK;
00228   }
```

```
00229 }
00230
00231 /**
00232   * @brief  Writes an amount of data from th
e SRAM device in DMA mode.
00233   * @param  uwStartAddress: Write start addr
ess
00234   * @param  pData: Pointer to data to be wri
tten
00235   * @param  uwDataSize: Size of written data
 from the memory
00236   * @retval SRAM status
00237   */
00238 uint8_t BSP_SRAM_WriteData_DMA(uint32_t uwSt
artAddress, uint16_t *pData, uint32_t uwDataSize)
00239 {
00240   if(HAL_SRAM_Write_DMA(&sramHandle, (uint32
_t *)uwStartAddress, (uint32_t *)pData, uwDataSize
) != HAL_OK)
00241   {
00242     return SRAM_ERROR;
00243   }
00244   else
00245   {
00246     return SRAM_OK;
00247   }
00248 }
00249
00250 /**
00251   * @brief  Handles SRAM DMA transfer interr
upt request.
00252   */
00253 void BSP_SRAM_DMA_IRQHandler(void)
00254 {
00255   HAL_DMA_IRQHandler(sramHandle.hdma);
00256 }
00257
```

```
00258 /**
00259   * @brief  Initializes SRAM MSP.
00260   */
00261 static void SRAM_MspInit(void)
00262 {
00263   static DMA_HandleTypeDef dmaHandle;
00264   GPIO_InitTypeDef GPIO_Init_Structure;
00265   SRAM_HandleTypeDef *hsram = &sramHandle;
00266
00267   /* Enable FMC clock */
00268   __FMC_CLK_ENABLE();
00269
00270   /* Enable chosen DMAx clock */
00271   __SRAM_DMAx_CLK_ENABLE();
00272
00273   /* Enable GPIOs clock */
00274   __GPIOD_CLK_ENABLE();
00275   __GPIOE_CLK_ENABLE();
00276   __GPIOF_CLK_ENABLE();
00277   __GPIOG_CLK_ENABLE();
00278
00279   /* Common GPIO configuration */
00280   GPIO_Init_Structure.Mode      = GPIO_MODE_
AF_PP;
00281   GPIO_Init_Structure.Pull      = GPIO_PULLU
P;
00282   GPIO_Init_Structure.Speed     = GPIO_SPEED
_HIGH;
00283   GPIO_Init_Structure.Alternate = GPIO_AF12_
FMC;
00284
00285   /* GPIOD configuration */
00286   GPIO_Init_Structure.Pin   = GPIO_PIN_0 | G
PIO_PIN_1 | GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_8
   |\
00287                               GPIO_PIN_9 | G
PIO_PIN_10 | GPIO_PIN_11 | GPIO_PIN_12 | GPIO_PIN_
```

```
13 |\
00288                                   GPIO_PIN_14 |
GPIO_PIN_15;
00289   HAL_GPIO_Init(GPIOD, &GPIO_Init_Structure)
;
00290
00291   /* GPIOE configuration */
00292   GPIO_Init_Structure.Pin   = GPIO_PIN_0 | G
PIO_PIN_1 | GPIO_PIN_3| GPIO_PIN_4 | GPIO_PIN_7
  |\
00293                                   GPIO_PIN_8 | G
PIO_PIN_9 | GPIO_PIN_10 | GPIO_PIN_11 | GPIO_PIN_1
2 |\
00294                                   GPIO_PIN_13 |
GPIO_PIN_14 | GPIO_PIN_15;
00295   HAL_GPIO_Init(GPIOE, &GPIO_Init_Structure)
;
00296
00297   /* GPIOF configuration */
00298   GPIO_Init_Structure.Pin   = GPIO_PIN_0 | G
PIO_PIN_1 | GPIO_PIN_2| GPIO_PIN_3 | GPIO_PIN_4
  |\
00299                                   GPIO_PIN_5 | G
PIO_PIN_12 | GPIO_PIN_13 | GPIO_PIN_14 | GPIO_PIN_
15;
00300   HAL_GPIO_Init(GPIOF, &GPIO_Init_Structure)
;
00301
00302   /* GPIOG configuration */
00303   GPIO_Init_Structure.Pin   = GPIO_PIN_0 | G
PIO_PIN_1 | GPIO_PIN_2| GPIO_PIN_3 | GPIO_PIN_4
  |\
00304                                   GPIO_PIN_5 | G
PIO_PIN_9;
00305   HAL_GPIO_Init(GPIOG, &GPIO_Init_Structure)
;
00306
```

```
00307   /* Configure common DMA parameters */
00308   dmaHandle.Init.Channel             = SRAM_
DMAx_CHANNEL;
00309   dmaHandle.Init.Direction           = DMA_M
EMORY_TO_MEMORY;
00310   dmaHandle.Init.PeriphInc           = DMA_P
INC_ENABLE;
00311   dmaHandle.Init.MemInc              = DMA_M
INC_ENABLE;
00312   dmaHandle.Init.PeriphDataAlignment = DMA_P
DATAALIGN_HALFWORD;
00313   dmaHandle.Init.MemDataAlignment    = DMA_M
DATAALIGN_HALFWORD;
00314   dmaHandle.Init.Mode                = DMA_N
ORMAL;
00315   dmaHandle.Init.Priority            = DMA_P
RIORITY_HIGH;
00316   dmaHandle.Init.FIFOMode            = DMA_F
IFOMODE_DISABLE;
00317   dmaHandle.Init.FIFOThreshold       = DMA_F
IFO_THRESHOLD_FULL;
00318   dmaHandle.Init.MemBurst            = DMA_M
BURST_SINGLE;
00319   dmaHandle.Init.PeriphBurst         = DMA_P
BURST_SINGLE;
00320
00321   dmaHandle.Instance = SRAM_DMAx_STREAM;
00322
00323    /* Associate the DMA handle */
00324   __HAL_LINKDMA(hsram, hdma, dmaHandle);
00325
00326   /* Deinitialize the Stream for new transfe
r */
00327   HAL_DMA_DeInit(&dmaHandle);
00328
00329   /* Configure the DMA Stream */
00330   HAL_DMA_Init(&dmaHandle);
```

```
00331
00332   /* NVIC configuration for DMA transfer com
plete interrupt */
00333   HAL_NVIC_SetPriority(SRAM_DMAx_IRQn, 5, 0)
;
00334   HAL_NVIC_EnableIRQ(SRAM_DMAx_IRQn);
00335 }
00336
00337 /**
00338   * @}
00339   */
00340
00341 /**
00342   * @}
00343   */
00344
00345 /**
00346   * @}
00347   */
00348
00349 /**
00350   * @}
00351   */
00352
00353 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM324x9I_EVAL BSP User Manual

Modules

BSP

Modules

STM324x9I EVAL

Generated on Wed Jan 13 2016 15:52:54 for STM324x9I_EVAL BSP
User Manual by doxygen 1.7.6.1

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL

BSP

Modules

- **STM324x9I EVAL LOW LEVEL**
- **STM324x9I EVAL AUDIO**

  This file includes the low layer driver for wm8994 Audio Codec available on STM324x9I-EVAL evaluation board(MB1045).

- **STM324x9I EVAL CAMERA**
- **STM324x9I EVAL EEPROM**

  This file includes the I2C EEPROM driver of STM324x9I-EVAL evaluation board.

- **STM324x9I EVAL IO**
- **STM324x9I EVAL LCD**
- **STM324x9I EVAL NOR**
- **STM324x9I EVAL SD**
- **STM324x9I EVAL SDRAM**
- **STM324x9I EVAL SRAM**
- **STM324x9I EVAL TS**

STM324x9I_EVAL BSP User Manual

STM324x9I EVAL LOW LEVEL Exported Constants

STM324x9I EVAL LOW LEVEL

Modules

STM324x9I EVAL LOW LEVEL LED

Define for STM324x9I_EVAL board.

STM324x9I EVAL LOW LEVEL BUTTON

STM324x9I EVAL LOW LEVEL COM