STM3210C_EVAL BSP User Manual

TFT_LCD_TypeDef Struct Reference

Private Types Definitions

Data Fields

__IO uint16_t	**LCD_REG_R**
__IO uint16_t	**LCD_RAM_R**
__IO uint16_t	**LCD_REG_W**
__IO uint16_t	**LCD_RAM_W**

Detailed Description

Definition at line **58** of file **stm3210c_eval.c**.

Field Documentation

__IO uint16_t TFT_LCD_TypeDef::LCD_RAM_R

Definition at line 61 of file stm3210c_eval.c.

__IO uint16_t TFT_LCD_TypeDef::LCD_RAM_W

Definition at line 63 of file stm3210c_eval.c.

__IO uint16_t TFT_LCD_TypeDef::LCD_REG_R

Definition at line 60 of file stm3210c_eval.c.

__IO uint16_t TFT_LCD_TypeDef::LCD_REG_W

Definition at line 62 of file stm3210c_eval.c.

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

- stm3210c_eval.c

STM3210C_EVAL BSP User Manual

Private Types Definitions

STM3210C_EVAL IO Expander

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Private_Defines

STM3210C_EVAL IO Expander

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Private_Macros

STM3210C_EVAL IO Expander

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Private_Function_Prototypes

STM3210C_EVAL IO Expander

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Exported_Macros

STM3210C_EVAL IO Expander

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

LCD_DrawPropTypeDef Struct Reference

Exported_Types

```
#include <stm3210c_eval_lcd.h>
```

Data Fields

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

Detailed Description

Definition at line **68** of file **stm3210c_eval_lcd.h**.

Field Documentation

uint32_t LCD_DrawPropTypeDef::BackColor

Definition at line 71 of file stm3210c_eval_lcd.h.

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

sFONT* LCD_DrawPropTypeDef::pFont

Definition at line 72 of file stm3210c_eval_lcd.h.

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

uint32_t LCD_DrawPropTypeDef::TextColor

Definition at line 70 of file stm3210c_eval_lcd.h.

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

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

- stm3210c_eval_lcd.h

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP
User Manual by  1.7.6.1

STM3210C_EVAL BSP User Manual

Point Struct Reference

Exported_Constants

```
#include <stm3210c_eval_lcd.h>
```

Data Fields

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

Detailed Description

Definition at line **89** of file **stm3210c_eval_lcd.h**.

Field Documentation

int16_t Point::X

Definition at line **91** of file **stm3210c_eval_lcd.h**.

Referenced by **BSP_LCD_DrawPolygon()**.

int16_t Point::Y

Definition at line **92** of file **stm3210c_eval_lcd.h**.

Referenced by **BSP_LCD_DrawPolygon()**.

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

- **stm3210c_eval_lcd.h**

STM3210C_EVAL BSP User Manual

Private_Types_Definitions

STM3210C_EVAL SD

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Private_Macros

STM3210C_EVAL SD

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Private_Function_Prototypes

STM3210C_EVAL SD

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

SD_CSD Struct Reference

Exported_Types

Card Specific Data: CSD Register. More...

```
#include <stm3210c_eval_sd.h>
```

Data Fields

__IO uint8_t	**CSDStruct**
__IO uint8_t	**SysSpecVersion**
__IO uint8_t	**Reserved1**
__IO uint8_t	**TAAC**
__IO uint8_t	**NSAC**
__IO uint8_t	**MaxBusClkFrec**
__IO uint16_t	**CardComdClasses**
__IO uint8_t	**RdBlockLen**
__IO uint8_t	**PartBlockRead**
__IO uint8_t	**WrBlockMisalign**
__IO uint8_t	**RdBlockMisalign**
__IO uint8_t	**DSRImpl**
__IO uint8_t	**Reserved2**
__IO uint32_t	**DeviceSize**
__IO uint8_t	**MaxRdCurrentVDDMin**
__IO uint8_t	**MaxRdCurrentVDDMax**
__IO uint8_t	**MaxWrCurrentVDDMin**
__IO uint8_t	**MaxWrCurrentVDDMax**
__IO uint8_t	**DeviceSizeMul**
__IO uint8_t	**EraseGrSize**
__IO uint8_t	**EraseGrMul**
__IO uint8_t	**WrProtectGrSize**
__IO uint8_t	**WrProtectGrEnable**
__IO uint8_t	**ManDeflECC**
__IO uint8_t	**WrSpeedFact**
__IO uint8_t	**MaxWrBlockLen**
__IO uint8_t	**WriteBlockPaPartial**
__IO uint8_t	**Reserved3**
__IO uint8_t	**ContentProtectAppli**
__IO uint8_t	**FileFormatGrouop**
__IO uint8_t	**CopyFlag**

__IO uint8_t	**PermWrProtect**
__IO uint8_t	**TempWrProtect**
__IO uint8_t	**FileFormat**
__IO uint8_t	**ECC**
__IO uint8_t	**CSD_CRC**
__IO uint8_t	**Reserved4**

Detailed Description

Card Specific Data: CSD Register.

Definition at line **100** of file **stm3210c_eval_sd.h**.

Field Documentation

__IO uint16_t SD_CSD::CardComdClasses

Definition at line 108 of file stm3210c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::ContentProtectAppli

Definition at line 130 of file stm3210c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::CopyFlag

Definition at line 132 of file stm3210c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::CSD_CRC

Definition at line 137 of file stm3210c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::CSDStruct

Definition at line 102 of file stm3210c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint32_t SD_CSD::DeviceSize

Definition at line 115 of file stm3210c_eval_sd.h.

Referenced by BSP_SD_GetCardInfo(), and SD_GetCSDRegister().

__IO uint8_t SD_CSD::DeviceSizeMul

Definition at line 120 of file stm3210c_eval_sd.h.

Referenced by BSP_SD_GetCardInfo(), and SD_GetCSDRegister().

__IO uint8_t SD_CSD::DSRImpl

Definition at line 113 of file stm3210c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::ECC

Definition at line 136 of file stm3210c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::EraseGrMul

Definition at line 122 of file stm3210c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::EraseGrSize

Definition at line **121** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::FileFormat

Definition at line **135** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::FileFormatGrouop

Definition at line **131** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::ManDeflECC

Definition at line **125** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::MaxBusClkFrec

Definition at line **107** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::MaxRdCurrentVDDMax

Definition at line **117** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::MaxRdCurrentVDDMin

Definition at line **116** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::MaxWrBlockLen

Definition at line **127** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::MaxWrCurrentVDDMax

Definition at line **119** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::MaxWrCurrentVDDMin

Definition at line **118** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::NSAC

Definition at line **106** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::PartBlockRead

Definition at line 110 of file stm3210c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::PermWrProtect

Definition at line 133 of file stm3210c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::RdBlockLen

Definition at line 109 of file stm3210c_eval_sd.h.

Referenced by BSP_SD_GetCardInfo(), and SD_GetCSDRegister().

__IO uint8_t SD_CSD::RdBlockMisalign

Definition at line 112 of file stm3210c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::Reserved1

Definition at line 104 of file stm3210c_eval_sd.h.

Referenced by SD_GetCSDRegister().

__IO uint8_t SD_CSD::Reserved2

Definition at line **114** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::Reserved3

Definition at line **129** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::Reserved4

Definition at line **138** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::SysSpecVersion

Definition at line **103** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::TAAC

Definition at line **105** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::TempWrProtect

Definition at line **134** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::WrBlockMisalign

Definition at line **111** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::WriteBlockPaPartial

Definition at line **128** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::WrProtectGrEnable

Definition at line **124** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::WrProtectGrSize

Definition at line **123** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

__IO uint8_t SD_CSD::WrSpeedFact

Definition at line **126** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

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

- **stm3210c_eval_sd.h**

STM3210C_EVAL BSP User Manual

SD_CID Struct Reference

Exported_Types

Card Identification Data: CID Register. More...

```
#include <stm3210c_eval_sd.h>
```

Data Fields

__IO uint8_t	**ManufacturerID**
__IO uint16_t	**OEM_AppliID**
__IO uint32_t	**ProdName1**
__IO uint8_t	**ProdName2**
__IO uint8_t	**ProdRev**
__IO uint32_t	**ProdSN**
__IO uint8_t	**Reserved1**
__IO uint16_t	**ManufactDate**
__IO uint8_t	**CID_CRC**
__IO uint8_t	**Reserved2**

Detailed Description

Card Identification Data: CID Register.

Definition at line **144** of file **stm3210c_eval_sd.h**.

Field Documentation

__IO uint8_t SD_CID::CID_CRC

Definition at line 154 of file stm3210c_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint16_t SD_CID::ManufactDate

Definition at line 153 of file stm3210c_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint8_t SD_CID::ManufacturerID

Definition at line 146 of file stm3210c_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint16_t SD_CID::OEM_AppliID

Definition at line 147 of file stm3210c_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint32_t SD_CID::ProdName1

Definition at line 148 of file stm3210c_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint8_t SD_CID::ProdName2

Definition at line 149 of file stm3210c_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint8_t SD_CID::ProdRev

Definition at line 150 of file stm3210c_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint32_t SD_CID::ProdSN

Definition at line 151 of file stm3210c_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint8_t SD_CID::Reserved1

Definition at line 152 of file stm3210c_eval_sd.h.

Referenced by SD_GetCIDRegister().

__IO uint8_t SD_CID::Reserved2

Definition at line 155 of file stm3210c_eval_sd.h.

Referenced by SD_GetCIDRegister().

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

- **stm3210c_eval_sd.h**

STM3210C_EVAL BSP User Manual

SD_CardInfo Struct Reference

Exported_Types

SD Card information. More...

```
#include <stm3210c_eval_sd.h>
```

Data Fields

SD_CSD	**Csd**
SD_CID	**Cid**
uint32_t	**CardCapacity**
uint32_t	**CardBlockSize**

Detailed Description

SD Card information.

Definition at line **161** of file **stm3210c_eval_sd.h**.

Field Documentation

uint32_t SD_CardInfo::CardBlockSize

Definition at line **166** of file **stm3210c_eval_sd.h**.

Referenced by **BSP_SD_GetCardInfo()**.

uint32_t SD_CardInfo::CardCapacity

Definition at line **165** of file **stm3210c_eval_sd.h**.

Referenced by **BSP_SD_GetCardInfo()**.

SD_CID SD_CardInfo::Cid

Definition at line **164** of file **stm3210c_eval_sd.h**.

Referenced by **BSP_SD_GetCardInfo()**.

SD_CSD SD_CardInfo::Csd

Definition at line **163** of file **stm3210c_eval_sd.h**.

Referenced by **BSP_SD_GetCardInfo()**.

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

- **stm3210c_eval_sd.h**

STM3210C_EVAL BSP User Manual

Exported_Macro

STM3210C_EVAL SD

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Private_Types_Definitions

STM3210C_EVAL Touch Screen

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Private_Defines

STM3210C_EVAL Touch Screen

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Private_Macros

STM3210C_EVAL Touch Screen

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Private_Function_Prototypes

STM3210C_EVAL Touch Screen

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

TS_StateTypeDef Struct Reference

Exported_Types

```
#include <stm3210c_eval_ts.h>
```

Data Fields

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

Detailed Description

Definition at line **68** of file **stm3210c_eval_ts.h**.

Field Documentation

uint16_t TS_StateTypeDef::TouchDetected

Definition at line 70 of file stm3210c_eval_ts.h.

Referenced by BSP_TS_GetState().

uint16_t TS_StateTypeDef::x

Definition at line 71 of file stm3210c_eval_ts.h.

Referenced by BSP_TS_GetState().

uint16_t TS_StateTypeDef::y

Definition at line 72 of file stm3210c_eval_ts.h.

Referenced by BSP_TS_GetState().

uint16_t TS_StateTypeDef::z

Definition at line 73 of file stm3210c_eval_ts.h.

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

- stm3210c_eval_ts.h

STM3210C_EVAL BSP User Manual

Exported_Macros

STM3210C_EVAL Touch Screen

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Private Types Definitions

STM3210C_EVAL_ACCELEROMETER

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Private Defines

STM3210C_EVAL_ACCELEROMETER

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Private Macros

STM3210C_EVAL_ACCELEROMETER

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Private FunctionPrototypes

STM3210C_EVAL_ACCELEROMETER

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

AUDIO_Private_Types

STM3210C_EVAL_AUDIO

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

AUDIO_Private_Defines

STM3210C_EVAL_AUDIO

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

AUDIO Private_Macros

STM3210C_EVAL_AUDIO

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

AUDIO_Private_Function_Prototypes

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

AUDIO_Exported_Types

STM3210C_EVAL_AUDIO

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Private Types

STM3210C_EVAL_EEPROM

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Private Defines

STM3210C_EVAL_EEPROM

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Private Macros

STM3210C_EVAL_EEPROM

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

EEPROM_DrvTypeDef Struct Reference

Exported Types

```
#include <stm3210c_eval_eeprom.h>
```

Data Fields

uint32_t(*	**Init**)(void)
uint32_t(*	**ReadBuffer**)(uint8_t *, uint16_t, uint32_t *)
uint32_t(*	**WritePage**)(uint8_t *, uint16_t, uint32_t *)

Detailed Description

Definition at line **65** of file **stm3210c_eval_eeprom.h**.

Field Documentation

uint32_t(* EEPROM_DrvTypeDef::Init)(void)

Definition at line 67 of file stm3210c_eval_eeprom.h.

Referenced by BSP_EEPROM_Init().

uint32_t(* EEPROM_DrvTypeDef::ReadBuffer)(uint8_t *, uint16_t, ui

Definition at line 68 of file stm3210c_eval_eeprom.h.

Referenced by BSP_EEPROM_ReadBuffer().

uint32_t(* EEPROM_DrvTypeDef::WritePage)(uint8_t *, uint16_t, uir

Definition at line 69 of file stm3210c_eval_eeprom.h.

Referenced by BSP_EEPROM_WriteBuffer().

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

- stm3210c_eval_eeprom.h

STM3210C_EVAL BSP User Manual

Exported Macros

STM3210C_EVAL_EEPROM

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

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

- b -

- BackColor : **LCD_DrawPropTypeDef**

- c -

- CardBlockSize : **SD_CardInfo**
- CardCapacity : **SD_CardInfo**
- CardComdClasses : **SD_CSD**
- Cid : **SD_CardInfo**
- CID_CRC : **SD_CID**
- ContentProtectAppli : **SD_CSD**
- CopyFlag : **SD_CSD**
- Csd : **SD_CardInfo**
- CSD_CRC : **SD_CSD**
- CSDStruct : **SD_CSD**

- d -

- DeviceSize : **SD_CSD**
- DeviceSizeMul : **SD_CSD**
- DSRImpl : **SD_CSD**

- e -

- ECC : **SD_CSD**
- EraseGrMul : **SD_CSD**
- EraseGrSize : **SD_CSD**

- f -

- FileFormat : **SD_CSD**
- FileFormatGrouop : **SD_CSD**

- i -

- Init : **EEPROM_DrvTypeDef**

- l -

- LCD_RAM_R : **TFT_LCD_TypeDef**
- LCD_RAM_W : **TFT_LCD_TypeDef**
- LCD_REG_R : **TFT_LCD_TypeDef**
- LCD_REG_W : **TFT_LCD_TypeDef**

- m -

- ManDeflECC : **SD_CSD**
- ManufactDate : **SD_CID**
- ManufacturerID : **SD_CID**
- MaxBusClkFrec : **SD_CSD**
- MaxRdCurrentVDDMax : **SD_CSD**
- MaxRdCurrentVDDMin : **SD_CSD**
- MaxWrBlockLen : **SD_CSD**
- MaxWrCurrentVDDMax : **SD_CSD**
- MaxWrCurrentVDDMin : **SD_CSD**

- n -

- NSAC : **SD_CSD**

- o -

- OEM_AppliID : **SD_CID**

- p -

- PartBlockRead : **SD_CSD**
- PermWrProtect : **SD_CSD**
- pFont : **LCD_DrawPropTypeDef**
- ProdName1 : **SD_CID**
- ProdName2 : **SD_CID**
- ProdRev : **SD_CID**
- ProdSN : **SD_CID**

- r -

- RdBlockLen : **SD_CSD**
- RdBlockMisalign : **SD_CSD**
- ReadBuffer : **EEPROM_DrvTypeDef**
- Reserved1 : **SD_CSD** , **SD_CID**
- Reserved2 : **SD_CID** , **SD_CSD**
- Reserved3 : **SD_CSD**
- Reserved4 : **SD_CSD**

- s -

- SysSpecVersion : **SD_CSD**

- t -

- TAAC : **SD_CSD**
- TempWrProtect : **SD_CSD**
- TextColor : **LCD_DrawPropTypeDef**
- TouchDetected : **TS_StateTypeDef**

- w -

- WrBlockMisalign : **SD_CSD**
- WriteBlockPaPartial : **SD_CSD**
- WritePage : **EEPROM_DrvTypeDef**
- WrProtectGrEnable : **SD_CSD**

- WrProtectGrSize : **SD_CSD**
- WrSpeedFact : **SD_CSD**

- x -

- x : **TS_StateTypeDef**
- X : **Point**

- y -

- y : **TS_StateTypeDef**
- Y : **Point**

- z -

- z : **TS_StateTypeDef**

STM3210C_EVAL BSP User Manual

- b -

- BackColor : **LCD_DrawPropTypeDef**

- c -

- CardBlockSize : **SD_CardInfo**
- CardCapacity : **SD_CardInfo**
- CardComdClasses : **SD_CSD**
- Cid : **SD_CardInfo**
- CID_CRC : **SD_CID**
- ContentProtectAppli : **SD_CSD**
- CopyFlag : **SD_CSD**
- Csd : **SD_CardInfo**
- CSD_CRC : **SD_CSD**
- CSDStruct : **SD_CSD**

- d -

- DeviceSize : **SD_CSD**
- DeviceSizeMul : **SD_CSD**
- DSRImpl : **SD_CSD**

- e -

- ECC : **SD_CSD**
- EraseGrMul : **SD_CSD**
- EraseGrSize : **SD_CSD**

- f -

- FileFormat : **SD_CSD**
- FileFormatGrouop : **SD_CSD**

- i -

- Init : **EEPROM_DrvTypeDef**

- l -

- LCD_RAM_R : **TFT_LCD_TypeDef**
- LCD_RAM_W : **TFT_LCD_TypeDef**
- LCD_REG_R : **TFT_LCD_TypeDef**
- LCD_REG_W : **TFT_LCD_TypeDef**

- m -

- ManDeflECC : **SD_CSD**
- ManufactDate : **SD_CID**
- ManufacturerID : **SD_CID**
- MaxBusClkFrec : **SD_CSD**
- MaxRdCurrentVDDMax : **SD_CSD**
- MaxRdCurrentVDDMin : **SD_CSD**
- MaxWrBlockLen : **SD_CSD**
- MaxWrCurrentVDDMax : **SD_CSD**
- MaxWrCurrentVDDMin : **SD_CSD**

- n -

- NSAC : **SD_CSD**

- o -

- OEM_AppliID : **SD_CID**

- p -

- PartBlockRead : **SD_CSD**
- PermWrProtect : **SD_CSD**
- pFont : **LCD_DrawPropTypeDef**
- ProdName1 : **SD_CID**
- ProdName2 : **SD_CID**
- ProdRev : **SD_CID**
- ProdSN : **SD_CID**

- r -

- RdBlockLen : **SD_CSD**
- RdBlockMisalign : **SD_CSD**
- ReadBuffer : **EEPROM_DrvTypeDef**
- Reserved1 : **SD_CSD** , **SD_CID**
- Reserved2 : **SD_CID** , **SD_CSD**
- Reserved3 : **SD_CSD**
- Reserved4 : **SD_CSD**

- s -

- SysSpecVersion : **SD_CSD**

- t -

- TAAC : **SD_CSD**
- TempWrProtect : **SD_CSD**
- TextColor : **LCD_DrawPropTypeDef**
- TouchDetected : **TS_StateTypeDef**

- w -

- WrBlockMisalign : **SD_CSD**
- WriteBlockPaPartial : **SD_CSD**
- WritePage : **EEPROM_DrvTypeDef**
- WrProtectGrEnable : **SD_CSD**
- WrProtectGrSize : **SD_CSD**
- WrSpeedFact : **SD_CSD**

- x -

- x : **TS_StateTypeDef**
- X : **Point**

- y -

- y : **TS_StateTypeDef**
- Y : **Point**

- z -

- z : **TS_StateTypeDef**

STM3210C_EVAL BSP User Manual

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

- _ -

- __STM3210C_EVAL_BSP_VERSION : **stm3210c_eval.c**
- __STM3210C_EVAL_BSP_VERSION_MAIN : **stm3210c_eval.c**
- __STM3210C_EVAL_BSP_VERSION_RC : **stm3210c_eval.c**
- __STM3210C_EVAL_BSP_VERSION_SUB1 : **stm3210c_eval.c**
- __STM3210C_EVAL_BSP_VERSION_SUB2 : **stm3210c_eval.c**

STM3210C_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 : **stm3210c_eval_lcd.c**
- ACCELERO_ERROR : **stm3210c_eval_accelerometer.h**
- ACCELERO_IO_Init() : **stm3210c_eval.c**
- ACCELERO_IO_ITConfig() : **stm3210c_eval.c**
- ACCELERO_IO_Read() : **stm3210c_eval.c**
- ACCELERO_IO_Write() : **stm3210c_eval.c**
- ACCELERO_OK : **stm3210c_eval_accelerometer.h**
- ACCELERO_StatusTypeDef : **stm3210c_eval_accelerometer.h**
- ACCELERO_TIMEOUT : **stm3210c_eval_accelerometer.h**
- AcceleroDrv : **stm3210c_eval_accelerometer.c**
- AFIOCOM1_CLK_DISABLE : **stm3210c_eval.h**
- AFIOCOM1_CLK_ENABLE : **stm3210c_eval.h**
- AFIOCOMx_CLK_DISABLE : **stm3210c_eval.h**
- AFIOCOMx_CLK_ENABLE : **stm3210c_eval.h**
- AFIOCOMx_REMAP : **stm3210c_eval.h**
- AUDIO_ERROR : **stm3210c_eval_audio.h**
- AUDIO_I2C_ADDRESS : **stm3210c_eval.h**
- AUDIO_IO_Init() : **stm3210c_eval.c**
- AUDIO_IO_Read() : **stm3210c_eval.c**

- AUDIO_IO_Write() : **stm3210c_eval.c**
- AUDIO_OK : **stm3210c_eval_audio.h**
- AUDIO_OUT_IRQ_PREPRIO : **stm3210c_eval_audio.h**
- AUDIO_RESET_PIN : **stm3210c_eval.h**
- AUDIO_TIMEOUT : **stm3210c_eval_audio.h**

STM3210C_EVAL BSP User Manual

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

- b -

- bitmap : **stm3210c_eval_lcd.c**
- BSP_ACCELERO_Click_ITClear() : **stm3210c_eval_accelerometer.c**
- BSP_ACCELERO_Click_ITConfig() : **stm3210c_eval_accelerometer.c**
- BSP_ACCELERO_GetXYZ() : **stm3210c_eval_accelerometer.c**
- BSP_ACCELERO_Init() : **stm3210c_eval_accelerometer.c**
- BSP_ACCELERO_ReadID() : **stm3210c_eval_accelerometer.c**
- BSP_ACCELERO_Reset() : **stm3210c_eval_accelerometer.c**
- BSP_AUDIO_OUT_ChangeBuffer() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_Error_CallBack() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_HalfTransfer_CallBack() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_Init() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_Pause() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_Play() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_Resume() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_SetFrequency() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_SetMute() : **stm3210c_eval_audio.c**

- BSP_AUDIO_OUT_SetOutputMode() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_SetVolume() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_Stop() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_TransferComplete_CallBack() : **stm3210c_eval_audio.c**
- BSP_COM_Init() : **stm3210c_eval.c**
- BSP_EEPROM_Init() : **stm3210c_eval_eeprom.c**
- BSP_EEPROM_M24C08 : **stm3210c_eval_eeprom.h**
- BSP_EEPROM_M24C64_32 : **stm3210c_eval_eeprom.h**
- BSP_EEPROM_ReadBuffer() : **stm3210c_eval_eeprom.c**
- BSP_EEPROM_SelectDevice() : **stm3210c_eval_eeprom.c**
- BSP_EEPROM_TIMEOUT_UserCallback() : **stm3210c_eval_eeprom.c**
- BSP_EEPROM_WriteBuffer() : **stm3210c_eval_eeprom.c**
- BSP_GetVersion() : **stm3210c_eval.c**
- BSP_IO_ConfigPin() : **stm3210c_eval_io.c**
- BSP_IO_Init() : **stm3210c_eval_io.c**
- BSP_IO_ITClear() : **stm3210c_eval_io.c**
- BSP_IO_ITGetStatus() : **stm3210c_eval_io.c**
- BSP_IO_ReadPin() : **stm3210c_eval_io.c**
- BSP_IO_TogglePin() : **stm3210c_eval_io.c**
- BSP_IO_WritePin() : **stm3210c_eval_io.c**
- BSP_JOY_GetState() : **stm3210c_eval.c**
- BSP_JOY_Init() : **stm3210c_eval.c**
- BSP_LCD_Clear() : **stm3210c_eval_lcd.c**
- BSP_LCD_ClearStringLine() : **stm3210c_eval_lcd.c**
- BSP_LCD_DisplayChar() : **stm3210c_eval_lcd.c**
- BSP_LCD_DisplayOff() : **stm3210c_eval_lcd.c**
- BSP_LCD_DisplayOn() : **stm3210c_eval_lcd.c**
- BSP_LCD_DisplayStringAt() : **stm3210c_eval_lcd.c**
- BSP_LCD_DisplayStringAtLine() : **stm3210c_eval_lcd.c**
- BSP_LCD_DrawBitmap() : **stm3210c_eval_lcd.c**
- BSP_LCD_DrawCircle() : **stm3210c_eval_lcd.c**
- BSP_LCD_DrawEllipse() : **stm3210c_eval_lcd.c**
- BSP_LCD_DrawHLine() : **stm3210c_eval_lcd.c**
- BSP_LCD_DrawLine() : **stm3210c_eval_lcd.c**
- BSP_LCD_DrawPolygon() : **stm3210c_eval_lcd.c**

- BSP_LCD_DrawRect() : **stm3210c_eval_lcd.c**
- BSP_LCD_DrawRGBImage() : **stm3210c_eval_lcd.c**
- BSP_LCD_DrawVLine() : **stm3210c_eval_lcd.c**
- BSP_LCD_FillCircle() : **stm3210c_eval_lcd.c**
- BSP_LCD_FillEllipse() : **stm3210c_eval_lcd.c**
- BSP_LCD_FillRect() : **stm3210c_eval_lcd.c**
- BSP_LCD_GetBackColor() : **stm3210c_eval_lcd.c**
- BSP_LCD_GetFont() : **stm3210c_eval_lcd.c**
- BSP_LCD_GetTextColor() : **stm3210c_eval_lcd.c**
- BSP_LCD_GetXSize() : **stm3210c_eval_lcd.c**
- BSP_LCD_GetYSize() : **stm3210c_eval_lcd.c**
- BSP_LCD_Init() : **stm3210c_eval_lcd.c**
- BSP_LCD_ReadPixel() : **stm3210c_eval_lcd.c**
- BSP_LCD_SetBackColor() : **stm3210c_eval_lcd.c** , **stm3210c_eval_lcd.h**
- BSP_LCD_SetFont() : **stm3210c_eval_lcd.c**
- BSP_LCD_SetTextColor() : **stm3210c_eval_lcd.c** , **stm3210c_eval_lcd.h**
- BSP_LED_Init() : **stm3210c_eval.c**
- BSP_LED_Off() : **stm3210c_eval.c**
- BSP_LED_On() : **stm3210c_eval.c**
- BSP_LED_Toggle() : **stm3210c_eval.c**
- BSP_PB_GetState() : **stm3210c_eval.c**
- BSP_PB_Init() : **stm3210c_eval.c**
- BSP_SD_Erase() : **stm3210c_eval_sd.c**
- BSP_SD_GetCardInfo() : **stm3210c_eval_sd.c**
- BSP_SD_GetStatus() : **stm3210c_eval_sd.c**
- BSP_SD_Init() : **stm3210c_eval_sd.c**
- BSP_SD_IsDetected() : **stm3210c_eval_sd.c**
- BSP_SD_ReadBlocks() : **stm3210c_eval_sd.c**
- BSP_SD_WriteBlocks() : **stm3210c_eval_sd.c**
- BSP_TS_GetState() : **stm3210c_eval_ts.c**
- BSP_TS_Init() : **stm3210c_eval_ts.c**
- BSP_TS_ITClear() : **stm3210c_eval_ts.c**
- BSP_TS_ITConfig() : **stm3210c_eval_ts.c**
- BSP_TS_ITGetStatus() : **stm3210c_eval_ts.c**
- BUTTON_IRQn : **stm3210c_eval.c**

- BUTTON_KEY : **stm3210c_eval.h**
- BUTTON_MODE_EXTI : **stm3210c_eval.h**
- BUTTON_MODE_GPIO : **stm3210c_eval.h**
- BUTTON_PIN : **stm3210c_eval.c**
- BUTTON_PORT : **stm3210c_eval.c**
- BUTTON_TAMPER : **stm3210c_eval.h**
- Button_TypeDef : **stm3210c_eval.h**
- BUTTON_WAKEUP : **stm3210c_eval.h**
- ButtonMode_TypeDef : **stm3210c_eval.h**
- BUTTONn : **stm3210c_eval.h**
- BUTTONx_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- BUTTONx_GPIO_CLK_ENABLE : **stm3210c_eval.h**

STM3210C_EVAL BSP User Manual

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

- c -

- CENTER_MODE : **stm3210c_eval_lcd.h**
- COM1 : **stm3210c_eval.h**
- COM2 : **stm3210c_eval.h**
- COM_RX_PIN : **stm3210c_eval.c**
- COM_RX_PORT : **stm3210c_eval.c**
- COM_TX_PIN : **stm3210c_eval.c**
- COM_TX_PORT : **stm3210c_eval.c**
- COM_TypeDef : **stm3210c_eval.h**
- COM_USART : **stm3210c_eval.c**
- COMn : **stm3210c_eval.h**
- COMx_CLK_DISABLE : **stm3210c_eval.h**
- COMx_CLK_ENABLE : **stm3210c_eval.h**
- COMx_RX_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- COMx_RX_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- COMx_TX_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- COMx_TX_GPIO_CLK_ENABLE : **stm3210c_eval.h**

User Manual by  1.7.6.1

STM3210C_EVAL BSP User Manual

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

- d -

- DMA_MAX : **stm3210c_eval_audio.h**
- DMA_MAX_SZE : **stm3210c_eval_audio.h**
- DrawProp : **stm3210c_eval_lcd.c**

STM3210C_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_ADDRESS_M24C08_BLOCK0 : **stm3210c_eval_eeprom.h**
- EEPROM_ADDRESS_M24C08_BLOCK1 : **stm3210c_eval_eeprom.h**
- EEPROM_ADDRESS_M24C08_BLOCK2 : **stm3210c_eval_eeprom.h**
- EEPROM_ADDRESS_M24C08_BLOCK3 : **stm3210c_eval_eeprom.h**
- EEPROM_ADDRESS_M24C64_32 : **stm3210c_eval_eeprom.h**
- EEPROM_FAIL : **stm3210c_eval_eeprom.h**
- EEPROM_I2C_Drv : **stm3210c_eval_eeprom.c**
- EEPROM_I2C_Init() : **stm3210c_eval_eeprom.c**
- EEPROM_I2C_IO_Init() : **stm3210c_eval.c** , **stm3210c_eval_eeprom.h**
- EEPROM_I2C_IO_IsDeviceReady() : **stm3210c_eval.c** , **stm3210c_eval_eeprom.h**
- EEPROM_I2C_IO_ReadData() : **stm3210c_eval.c** , **stm3210c_eval_eeprom.h**
- EEPROM_I2C_IO_WriteData() : **stm3210c_eval_eeprom.h** ,

stm3210c_eval.c

- EEPROM_I2C_ReadBuffer() : **stm3210c_eval_eeprom.c**
- EEPROM_I2C_WaitEepromStandbyState() : **stm3210c_eval_eeprom.c**
- EEPROM_I2C_WritePage() : **stm3210c_eval_eeprom.c**
- EEPROM_MAX_TRIALS : **stm3210c_eval_eeprom.h**
- EEPROM_OK : **stm3210c_eval_eeprom.h**
- EEPROM_PAGESIZE_M24C08 : **stm3210c_eval_eeprom.h**
- EEPROM_PAGESIZE_M24C64_32 : **stm3210c_eval_eeprom.h**
- EEPROM_SelectedDevice : **stm3210c_eval_eeprom.c**
- EEPROM_TIMEOUT : **stm3210c_eval_eeprom.h**
- EEPROMAddress : **stm3210c_eval_eeprom.c**
- EEPROMDataRead : **stm3210c_eval_eeprom.c**
- EEPROMDataWrite : **stm3210c_eval_eeprom.c**
- EEPROMPageSize : **stm3210c_eval_eeprom.c**
- EVAL_COM1 : **stm3210c_eval.h**
- EVAL_COM1_CLK_DISABLE : **stm3210c_eval.h**
- EVAL_COM1_CLK_ENABLE : **stm3210c_eval.h**
- EVAL_COM1_IRQn : **stm3210c_eval.h**
- EVAL_COM1_RX_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- EVAL_COM1_RX_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- EVAL_COM1_RX_GPIO_PORT : **stm3210c_eval.h**
- EVAL_COM1_RX_PIN : **stm3210c_eval.h**
- EVAL_COM1_TX_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- EVAL_COM1_TX_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- EVAL_COM1_TX_GPIO_PORT : **stm3210c_eval.h**
- EVAL_COM1_TX_PIN : **stm3210c_eval.h**
- EVAL_I2Cx : **stm3210c_eval.h**
- EVAL_I2Cx_CLK_ENABLE : **stm3210c_eval.h**
- EVAL_I2Cx_ER_IRQHandler : **stm3210c_eval.h**
- EVAL_I2Cx_ER_IRQn : **stm3210c_eval.h**
- EVAL_I2Cx_EV_IRQHandler : **stm3210c_eval.h**
- EVAL_I2Cx_EV_IRQn : **stm3210c_eval.h**
- EVAL_I2Cx_FORCE_RESET : **stm3210c_eval.h**
- EVAL_I2Cx_RELEASE_RESET : **stm3210c_eval.h**
- EVAL_I2Cx_SCL_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- EVAL_I2Cx_SCL_GPIO_PORT : **stm3210c_eval.h**

- EVAL_I2Cx_SCL_PIN : **stm3210c_eval.h**
- EVAL_I2Cx_SDA_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- EVAL_I2Cx_SDA_GPIO_PORT : **stm3210c_eval.h**
- EVAL_I2Cx_SDA_PIN : **stm3210c_eval.h**
- EVAL_I2Cx_TIMEOUT_MAX : **stm3210c_eval.h**
- EVAL_SPIx : **stm3210c_eval.h**
- EVAL_SPIx_CLK_ENABLE : **stm3210c_eval.h**
- EVAL_SPIx_MISO_MOSI_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- EVAL_SPIx_MISO_MOSI_GPIO_PORT : **stm3210c_eval.h**
- EVAL_SPIx_MISO_PIN : **stm3210c_eval.h**
- EVAL_SPIx_MOSI_PIN : **stm3210c_eval.h**
- EVAL_SPIx_SCK_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- EVAL_SPIx_SCK_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- EVAL_SPIx_SCK_GPIO_PORT : **stm3210c_eval.h**
- EVAL_SPIx_SCK_PIN : **stm3210c_eval.h**
- EVAL_SPIx_TIMEOUT_MAX : **stm3210c_eval.h**

STM3210C_EVAL BSP User Manual

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

- h -

- HAL_I2S_ErrorCallback() : **stm3210c_eval_audio.c**
- HAL_I2S_TxCpltCallback() : **stm3210c_eval_audio.c**
- HAL_I2S_TxHalfCpltCallback() : **stm3210c_eval_audio.c**
- hAudioOutI2s : **stm3210c_eval_audio.c**
- heval_I2c : **stm3210c_eval.c**
- heval_Spi : **stm3210c_eval.c**

STM3210C_EVAL BSP User Manual

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

- i -

- I2Cx_Error() : **stm3210c_eval.c**
- I2Cx_Init() : **stm3210c_eval.c**
- I2Cx_IsDeviceReady() : **stm3210c_eval.c**
- I2Cx_ITConfig() : **stm3210c_eval.c**
- I2Cx_MspInit() : **stm3210c_eval.c**
- I2Cx_ReadBuffer() : **stm3210c_eval.c**
- I2Cx_ReadData() : **stm3210c_eval.c**
- I2Cx_ReadMultiple() : **stm3210c_eval.c**
- I2Cx_WriteBuffer() : **stm3210c_eval.c**
- I2Cx_WriteData() : **stm3210c_eval.c**
- I2cxTimeout : **stm3210c_eval.c**
- I2SOUT : **stm3210c_eval_audio.h**
- I2SOUT_CLK_ENABLE : **stm3210c_eval_audio.h**
- I2SOUT_DMAx_CHANNEL : **stm3210c_eval_audio.h**
- I2SOUT_DMAx_CLK_ENABLE : **stm3210c_eval_audio.h**
- I2SOUT_DMAx_IRQ : **stm3210c_eval_audio.h**
- I2SOUT_DMAx_MEM_DATA_SIZE : **stm3210c_eval_audio.h**
- I2SOUT_DMAx_PERIPH_DATA_SIZE : **stm3210c_eval_audio.h**
- I2SOUT_Init() : **stm3210c_eval_audio.c**

- I2SOUT_IRQHandler : **stm3210c_eval_audio.h**
- I2SOUT_MCK_CLK_ENABLE : **stm3210c_eval_audio.h**
- I2SOUT_MCK_GPIO_PORT : **stm3210c_eval_audio.h**
- I2SOUT_MCK_PIN : **stm3210c_eval_audio.h**
- I2SOUT_MspInit() : **stm3210c_eval_audio.c**
- I2SOUT_SCK_PIN : **stm3210c_eval_audio.h**
- I2SOUT_SCK_SD_CLK_ENABLE : **stm3210c_eval_audio.h**
- I2SOUT_SCK_SD_GPIO_PORT : **stm3210c_eval_audio.h**
- I2SOUT_SD_PIN : **stm3210c_eval_audio.h**
- I2SOUT_WS_CLK_ENABLE : **stm3210c_eval_audio.h**
- I2SOUT_WS_GPIO_PORT : **stm3210c_eval_audio.h**
- I2SOUT_WS_PIN : **stm3210c_eval_audio.h**
- INTERNAL_BUFF_SIZE : **stm3210c_eval_audio.h**
- io1_driver : **stm3210c_eval_io.c**
- IO1_I2C_ADDRESS : **stm3210c_eval.h**
- IO1_PIN_0 : **stm3210c_eval_io.h**
- IO1_PIN_1 : **stm3210c_eval_io.h**
- IO1_PIN_2 : **stm3210c_eval_io.h**
- IO1_PIN_3 : **stm3210c_eval_io.h**
- IO1_PIN_4 : **stm3210c_eval_io.h**
- IO1_PIN_5 : **stm3210c_eval_io.h**
- IO1_PIN_6 : **stm3210c_eval_io.h**
- IO1_PIN_7 : **stm3210c_eval_io.h**
- IO1_PIN_ALL : **stm3210c_eval_io.h**
- IO1_PIN_OFFSET : **stm3210c_eval_io.h**
- io2_driver : **stm3210c_eval_io.c**
- IO2_I2C_ADDRESS : **stm3210c_eval.h**
- IO2_PIN_0 : **stm3210c_eval_io.h**
- IO2_PIN_1 : **stm3210c_eval_io.h**
- IO2_PIN_2 : **stm3210c_eval_io.h**
- IO2_PIN_3 : **stm3210c_eval_io.h**
- IO2_PIN_4 : **stm3210c_eval_io.h**
- IO2_PIN_5 : **stm3210c_eval_io.h**
- IO2_PIN_6 : **stm3210c_eval_io.h**
- IO2_PIN_7 : **stm3210c_eval_io.h**
- IO2_PIN_ALL : **stm3210c_eval_io.h**
- IO2_PIN_OFFSET : **stm3210c_eval_io.h**

- IO_ERROR : **stm3210c_eval_io.h**
- IO_OK : **stm3210c_eval_io.h**
- IO_StatusTypeDef : **stm3210c_eval_io.h**
- IO_TIMEOUT : **stm3210c_eval_io.h**
- IOE_Delay() : **stm3210c_eval.c**
- IOE_Init() : **stm3210c_eval.c**
- IOE_IT_EXTI_IRQHANDLER : **stm3210c_eval.h**
- IOE_IT_EXTI_IRQn : **stm3210c_eval.h**
- IOE_IT_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- IOE_IT_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- IOE_IT_GPIO_PORT : **stm3210c_eval.h**
- IOE_IT_PIN : **stm3210c_eval.h**
- IOE_ITConfig() : **stm3210c_eval.c**
- IOE_Read() : **stm3210c_eval.c**
- IOE_ReadMultiple() : **stm3210c_eval.c**
- IOE_Write() : **stm3210c_eval.c**

STM3210C_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 : **stm3210c_eval.h**
- JOY_DOWN : **stm3210c_eval.h**
- JOY_DOWN_PIN : **stm3210c_eval.h**
- JOY_LEFT : **stm3210c_eval.h**
- JOY_LEFT_PIN : **stm3210c_eval.h**
- JOY_MODE_EXTI : **stm3210c_eval.h**
- JOY_MODE_GPIO : **stm3210c_eval.h**
- JOY_NONE : **stm3210c_eval.h**
- JOY_NONE_PIN : **stm3210c_eval.h**
- JOY_RIGHT : **stm3210c_eval.h**
- JOY_RIGHT_PIN : **stm3210c_eval.h**
- JOY_SEL : **stm3210c_eval.h**
- JOY_SEL_PIN : **stm3210c_eval.h**
- JOY_UP : **stm3210c_eval.h**
- JOY_UP_PIN : **stm3210c_eval.h**
- JOYMode_TypeDef : **stm3210c_eval.h**
- JOYState_TypeDef : **stm3210c_eval.h**

Generated on Thu Dec 10 2015 17:13:53 for STM3210C_EVAL BSP
User Manual by  1.7.6.1

STM3210C_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 : **stm3210c_eval.h**
- KEY_BUTTON_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- KEY_BUTTON_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- KEY_BUTTON_GPIO_PORT : **stm3210c_eval.h**
- KEY_BUTTON_PIN : **stm3210c_eval.h**

STM3210C_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 -

- L1S302DL_I2C_ADDRESS : **stm3210c_eval.h**
- LCD_COLOR_BLACK : **stm3210c_eval_lcd.h**
- LCD_COLOR_BLUE : **stm3210c_eval_lcd.h**
- LCD_COLOR_BROWN : **stm3210c_eval_lcd.h**
- LCD_COLOR_CYAN : **stm3210c_eval_lcd.h**
- LCD_COLOR_DARKBLUE : **stm3210c_eval_lcd.h**
- LCD_COLOR_DARKCYAN : **stm3210c_eval_lcd.h**
- LCD_COLOR_DARKGRAY : **stm3210c_eval_lcd.h**
- LCD_COLOR_DARKGREEN : **stm3210c_eval_lcd.h**
- LCD_COLOR_DARKMAGENTA : **stm3210c_eval_lcd.h**
- LCD_COLOR_DARKRED : **stm3210c_eval_lcd.h**
- LCD_COLOR_DARKYELLOW : **stm3210c_eval_lcd.h**
- LCD_COLOR_GRAY : **stm3210c_eval_lcd.h**
- LCD_COLOR_GREEN : **stm3210c_eval_lcd.h**
- LCD_COLOR_LIGHTBLUE : **stm3210c_eval_lcd.h**
- LCD_COLOR_LIGHTCYAN : **stm3210c_eval_lcd.h**
- LCD_COLOR_LIGHTGRAY : **stm3210c_eval_lcd.h**
- LCD_COLOR_LIGHTGREEN : **stm3210c_eval_lcd.h**
- LCD_COLOR_LIGHTMAGENTA : **stm3210c_eval_lcd.h**

- LCD_COLOR_LIGHTRED : **stm3210c_eval_lcd.h**
- LCD_COLOR_LIGHTYELLOW : **stm3210c_eval_lcd.h**
- LCD_COLOR_MAGENTA : **stm3210c_eval_lcd.h**
- LCD_COLOR_ORANGE : **stm3210c_eval_lcd.h**
- LCD_COLOR_RED : **stm3210c_eval_lcd.h**
- LCD_COLOR_WHITE : **stm3210c_eval_lcd.h**
- LCD_COLOR_YELLOW : **stm3210c_eval_lcd.h**
- LCD_CS_HIGH : **stm3210c_eval.h**
- LCD_CS_LOW : **stm3210c_eval.h**
- LCD_DEFAULT_FONT : **stm3210c_eval_lcd.h**
- LCD_Delay() : **stm3210c_eval.c**
- LCD_DrawChar() : **stm3210c_eval_lcd.c**
- LCD_DrawPixel() : **stm3210c_eval_lcd.c**
- lcd_drv : **stm3210c_eval_lcd.c**
- LCD_ERROR : **stm3210c_eval_lcd.h**
- LCD_IO_Init() : **stm3210c_eval.c**
- LCD_IO_ReadData() : **stm3210c_eval.c**
- LCD_IO_WriteMultipleData() : **stm3210c_eval.c**
- LCD_IO_WriteReg() : **stm3210c_eval.c**
- LCD_NCS_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- LCD_NCS_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- LCD_NCS_GPIO_PORT : **stm3210c_eval.h**
- LCD_NCS_PIN : **stm3210c_eval.h**
- LCD_OK : **stm3210c_eval_lcd.h**
- LCD_READ_REG : **stm3210c_eval.c**
- LCD_SetDisplayWindow() : **stm3210c_eval_lcd.c**
- LCD_TIMEOUT : **stm3210c_eval_lcd.h**
- LCD_WRITE_REG : **stm3210c_eval.c**
- LED1 : **stm3210c_eval.h**
- LED1_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- LED1_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- LED1_GPIO_PORT : **stm3210c_eval.h**
- LED1_PIN : **stm3210c_eval.h**
- LED2 : **stm3210c_eval.h**
- LED2_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- LED2_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- LED2_GPIO_PORT : **stm3210c_eval.h**

- LED2_PIN : **stm3210c_eval.h**
- LED3 : **stm3210c_eval.h**
- LED3_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- LED3_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- LED3_GPIO_PORT : **stm3210c_eval.h**
- LED3_PIN : **stm3210c_eval.h**
- LED4 : **stm3210c_eval.h**
- LED4_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- LED4_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- LED4_GPIO_PORT : **stm3210c_eval.h**
- LED4_PIN : **stm3210c_eval.h**
- LED_BLUE : **stm3210c_eval.h**
- LED_GREEN : **stm3210c_eval.h**
- LED_ORANGE : **stm3210c_eval.h**
- LED_PIN : **stm3210c_eval.c**
- LED_PORT : **stm3210c_eval.c**
- LED_RED : **stm3210c_eval.h**
- Led_TypeDef : **stm3210c_eval.h**
- LEDn : **stm3210c_eval.h**
- LEDx_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- LEDx_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- LEFT_MODE : **stm3210c_eval_lcd.h**
- Line_ModeTypdef : **stm3210c_eval_lcd.h**

STM3210C_EVAL BSP User Manual

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

- m -

- MAX_HEIGHT_FONT : **stm3210c_eval_lcd.c**
- MAX_WIDTH_FONT : **stm3210c_eval_lcd.c**
- MEMS_ALL_PINS : **stm3210c_eval.h**
- MEMS_INT1_PIN : **stm3210c_eval.h**
- MEMS_INT2_PIN : **stm3210c_eval.h**
- MII_INT_PIN : **stm3210c_eval.h**
- MSD_ERROR : **stm3210c_eval_sd.h**
- MSD_OK : **stm3210c_eval_sd.h**
- MULTIPLEBYTE_CMD : **stm3210c_eval.h**

STM3210C_EVAL BSP User Manual

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

- o -

- OFFSET_BITMAP : **stm3210c_eval_lcd.c**

STM3210C_EVAL BSP User Manual

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

- p -

- pAudioDrv : **stm3210c_eval_audio.c**
- POLY_X : **stm3210c_eval_lcd.c**
- POLY_Y : **stm3210c_eval_lcd.c**
- pPoint : **stm3210c_eval_lcd.h**

STM3210C_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 -

- READ_STATUS : **stm3210c_eval.c**
- READWRITE_CMD : **stm3210c_eval.h**
- RIGHT_MODE : **stm3210c_eval_lcd.h**

STM3210C_EVAL BSP User Manual

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

- s -

- SD_ADDRESS_ERROR : **stm3210c_eval_sd.h**
- SD_BLOCK_SIZE : **stm3210c_eval_sd.h**
- SD_CMD_CLR_WRITE_PROT : **stm3210c_eval_sd.h**
- SD_CMD_ERASE : **stm3210c_eval_sd.h**
- SD_CMD_ERASE_GRP_END : **stm3210c_eval_sd.h**
- SD_CMD_ERASE_GRP_START : **stm3210c_eval_sd.h**
- SD_CMD_GO_IDLE_STATE : **stm3210c_eval_sd.h**
- SD_CMD_PROG_CSD : **stm3210c_eval_sd.h**
- SD_CMD_READ_MULT_BLOCK : **stm3210c_eval_sd.h**
- SD_CMD_READ_SINGLE_BLOCK : **stm3210c_eval_sd.h**
- SD_CMD_SD_ERASE_GRP_END : **stm3210c_eval_sd.h**
- SD_CMD_SD_ERASE_GRP_START : **stm3210c_eval_sd.h**
- SD_CMD_SEND_CID : **stm3210c_eval_sd.h**
- SD_CMD_SEND_CSD : **stm3210c_eval_sd.h**
- SD_CMD_SEND_OP_COND : **stm3210c_eval_sd.h**
- SD_CMD_SEND_STATUS : **stm3210c_eval_sd.h**
- SD_CMD_SEND_WRITE_PROT : **stm3210c_eval_sd.h**
- SD_CMD_SET_BLOCK_COUNT : **stm3210c_eval_sd.h**
- SD_CMD_SET_BLOCKLEN : **stm3210c_eval_sd.h**

- SD_CMD_SET_WRITE_PROT : **stm3210c_eval_sd.h**
- SD_CMD_STOP_TRANSMISSION : **stm3210c_eval_sd.h**
- SD_CMD_UNTAG_ERASE_GROUP : **stm3210c_eval_sd.h**
- SD_CMD_UNTAG_SECTOR : **stm3210c_eval_sd.h**
- SD_CMD_WRITE_MULT_BLOCK : **stm3210c_eval_sd.h**
- SD_CMD_WRITE_SINGLE_BLOCK : **stm3210c_eval_sd.h**
- SD_COM_CRC_ERROR : **stm3210c_eval_sd.h**
- SD_CS_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- SD_CS_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- SD_CS_GPIO_PORT : **stm3210c_eval.h**
- SD_CS_HIGH : **stm3210c_eval.h**
- SD_CS_LOW : **stm3210c_eval.h**
- SD_CS_PIN : **stm3210c_eval.h**
- SD_DATA_CRC_ERROR : **stm3210c_eval_sd.h**
- SD_DATA_OK : **stm3210c_eval_sd.h**
- SD_DATA_OTHER_ERROR : **stm3210c_eval_sd.h**
- SD_DATA_WRITE_ERROR : **stm3210c_eval_sd.h**
- SD_DETECT_EXTI_IRQn : **stm3210c_eval.h**
- SD_DETECT_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- SD_DETECT_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- SD_DETECT_GPIO_PORT : **stm3210c_eval.h**
- SD_DETECT_PIN : **stm3210c_eval.h**
- SD_DUMMY_BYTE : **stm3210c_eval.c** , **stm3210c_eval_sd.c**
- SD_ERASE_RESET : **stm3210c_eval_sd.h**
- SD_ERASE_SEQUENCE_ERROR : **stm3210c_eval_sd.h**
- SD_GetCIDRegister() : **stm3210c_eval_sd.c**
- SD_GetCSDRegister() : **stm3210c_eval_sd.c**
- SD_GetDataResponse() : **stm3210c_eval_sd.c**
- SD_GoIdleState() : **stm3210c_eval_sd.c**
- SD_ILLEGAL_COMMAND : **stm3210c_eval_sd.h**
- SD_IN_IDLE_STATE : **stm3210c_eval_sd.h**
- SD_Info : **stm3210c_eval_sd.h**
- SD_IO_Init() : **stm3210c_eval.c** , **stm3210c_eval_sd.h**
- SD_IO_ReadByte() : **stm3210c_eval.c** , **stm3210c_eval_sd.h**
- SD_IO_WaitResponse() : **stm3210c_eval.c** , **stm3210c_eval_sd.h**
- SD_IO_WriteByte() : **stm3210c_eval.c** , **stm3210c_eval_sd.h**

- SD_IO_WriteCmd() : **stm3210c_eval_sd.h** , **stm3210c_eval.c**
- SD_IO_WriteDummy() : **stm3210c_eval_sd.h** , **stm3210c_eval.c**
- SD_NO_RESPONSE_EXPECTED : **stm3210c_eval.c** , **stm3210c_eval_sd.c**
- SD_NOT_PRESENT : **stm3210c_eval_sd.h**
- SD_PARAMETER_ERROR : **stm3210c_eval_sd.h**
- SD_PRESENT : **stm3210c_eval_sd.h**
- SD_RESPONSE_FAILURE : **stm3210c_eval_sd.h**
- SD_RESPONSE_NO_ERROR : **stm3210c_eval_sd.h**
- SD_SendCmd() : **stm3210c_eval_sd.c**
- SD_START_DATA_MULTIPLE_BLOCK_READ : **stm3210c_eval_sd.h**
- SD_START_DATA_MULTIPLE_BLOCK_WRITE : **stm3210c_eval_sd.h**
- SD_START_DATA_SINGLE_BLOCK_READ : **stm3210c_eval_sd.h**
- SD_START_DATA_SINGLE_BLOCK_WRITE : **stm3210c_eval_sd.h**
- SD_STOP_DATA_MULTIPLE_BLOCK_WRITE : **stm3210c_eval_sd.h**
- SdStatus : **stm3210c_eval_sd.c**
- SET_INDEX : **stm3210c_eval.c**
- SPIx_Error() : **stm3210c_eval.c**
- SPIx_Init() : **stm3210c_eval.c**
- SPIx_MspInit() : **stm3210c_eval.c**
- SPIx_Read() : **stm3210c_eval.c**
- SPIx_Write() : **stm3210c_eval.c**
- SpixTimeout : **stm3210c_eval.c**
- START_BYTE : **stm3210c_eval.c**

STM3210C_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 : **stm3210c_eval.h**
- TAMPER_BUTTON_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- TAMPER_BUTTON_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- TAMPER_BUTTON_GPIO_PORT : **stm3210c_eval.h**
- TAMPER_BUTTON_PIN : **stm3210c_eval.h**
- TFT_LCD : **stm3210c_eval.c**
- TFT_LCD_BASE : **stm3210c_eval.c**
- ts_driver : **stm3210c_eval_ts.c**
- TS_ERROR : **stm3210c_eval_ts.h**
- TS_I2C_ADDRESS : **stm3210c_eval.h**
- TS_OK : **stm3210c_eval_ts.h**
- ts_orientation : **stm3210c_eval_ts.c**
- TS_StatusTypeDef : **stm3210c_eval_ts.h**
- TS_SWAP_NONE : **stm3210c_eval_ts.h**
- TS_SWAP_X : **stm3210c_eval_ts.h**
- TS_SWAP_XY : **stm3210c_eval_ts.h**
- TS_SWAP_Y : **stm3210c_eval_ts.h**
- TS_TIMEOUT : **stm3210c_eval_ts.h**
- ts_x_boundary : **stm3210c_eval_ts.c**

- ts_y_boundary : **stm3210c_eval_ts.c**
- TSENSOR_IO_Init() : **stm3210c_eval.c**
- TSENSOR_IO_IsDeviceReady() : **stm3210c_eval.c**
- TSENSOR_IO_Read() : **stm3210c_eval.c**
- TSENSOR_IO_Write() : **stm3210c_eval.c**

STM3210C_EVAL BSP User Manual

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

- v -

- VBAT_DIV_PIN : **stm3210c_eval.h**

STM3210C_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 : **stm3210c_eval.h**
- WAKEUP_BUTTON_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- WAKEUP_BUTTON_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- WAKEUP_BUTTON_GPIO_PORT : **stm3210c_eval.h**
- WAKEUP_BUTTON_PIN : **stm3210c_eval.h**

STM3210C_EVAL BSP User Manual

- a -

- ACCELERO_IO_Init() : **stm3210c_eval.c**
- ACCELERO_IO_ITConfig() : **stm3210c_eval.c**
- ACCELERO_IO_Read() : **stm3210c_eval.c**
- ACCELERO_IO_Write() : **stm3210c_eval.c**
- AUDIO_IO_Init() : **stm3210c_eval.c**
- AUDIO_IO_Read() : **stm3210c_eval.c**
- AUDIO_IO_Write() : **stm3210c_eval.c**

- b -

- BSP_ACCELERO_Click_ITClear() : **stm3210c_eval_accelerometer.c**
- BSP_ACCELERO_Click_ITConfig() : **stm3210c_eval_accelerometer.c**
- BSP_ACCELERO_GetXYZ() : **stm3210c_eval_accelerometer.c**
- BSP_ACCELERO_Init() : **stm3210c_eval_accelerometer.c**
- BSP_ACCELERO_ReadID() : **stm3210c_eval_accelerometer.c**
- BSP_ACCELERO_Reset() : **stm3210c_eval_accelerometer.c**
- BSP_AUDIO_OUT_ChangeBuffer() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_Error_CallBack() : **stm3210c_eval_audio.c**

- BSP_AUDIO_OUT_HalfTransfer_CallBack() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_Init() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_Pause() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_Play() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_Resume() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_SetFrequency() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_SetMute() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_SetOutputMode() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_SetVolume() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_Stop() : **stm3210c_eval_audio.c**
- BSP_AUDIO_OUT_TransferComplete_CallBack() : **stm3210c_eval_audio.c**
- BSP_COM_Init() : **stm3210c_eval.c**
- BSP_EEPROM_Init() : **stm3210c_eval_eeprom.c**
- BSP_EEPROM_ReadBuffer() : **stm3210c_eval_eeprom.c**
- BSP_EEPROM_SelectDevice() : **stm3210c_eval_eeprom.c**
- BSP_EEPROM_TIMEOUT_UserCallback() : **stm3210c_eval_eeprom.c**
- BSP_EEPROM_WriteBuffer() : **stm3210c_eval_eeprom.c**
- BSP_GetVersion() : **stm3210c_eval.c**
- BSP_IO_ConfigPin() : **stm3210c_eval_io.c**
- BSP_IO_Init() : **stm3210c_eval_io.c**
- BSP_IO_ITClear() : **stm3210c_eval_io.c**
- BSP_IO_ITGetStatus() : **stm3210c_eval_io.c**
- BSP_IO_ReadPin() : **stm3210c_eval_io.c**
- BSP_IO_TogglePin() : **stm3210c_eval_io.c**
- BSP_IO_WritePin() : **stm3210c_eval_io.c**
- BSP_JOY_GetState() : **stm3210c_eval.c**
- BSP_JOY_Init() : **stm3210c_eval.c**
- BSP_LCD_Clear() : **stm3210c_eval_lcd.c**
- BSP_LCD_ClearStringLine() : **stm3210c_eval_lcd.c**
- BSP_LCD_DisplayChar() : **stm3210c_eval_lcd.c**
- BSP_LCD_DisplayOff() : **stm3210c_eval_lcd.c**
- BSP_LCD_DisplayOn() : **stm3210c_eval_lcd.c**
- BSP_LCD_DisplayStringAt() : **stm3210c_eval_lcd.c**
- BSP_LCD_DisplayStringAtLine() : **stm3210c_eval_lcd.c**

- BSP_LCD_DrawBitmap() : **stm3210c_eval_lcd.c**
- BSP_LCD_DrawCircle() : **stm3210c_eval_lcd.c**
- BSP_LCD_DrawEllipse() : **stm3210c_eval_lcd.c**
- BSP_LCD_DrawHLine() : **stm3210c_eval_lcd.c**
- BSP_LCD_DrawLine() : **stm3210c_eval_lcd.c**
- BSP_LCD_DrawPolygon() : **stm3210c_eval_lcd.c**
- BSP_LCD_DrawRect() : **stm3210c_eval_lcd.c**
- BSP_LCD_DrawRGBImage() : **stm3210c_eval_lcd.c**
- BSP_LCD_DrawVLine() : **stm3210c_eval_lcd.c**
- BSP_LCD_FillCircle() : **stm3210c_eval_lcd.c**
- BSP_LCD_FillEllipse() : **stm3210c_eval_lcd.c**
- BSP_LCD_FillRect() : **stm3210c_eval_lcd.c**
- BSP_LCD_GetBackColor() : **stm3210c_eval_lcd.c**
- BSP_LCD_GetFont() : **stm3210c_eval_lcd.c**
- BSP_LCD_GetTextColor() : **stm3210c_eval_lcd.c**
- BSP_LCD_GetXSize() : **stm3210c_eval_lcd.c**
- BSP_LCD_GetYSize() : **stm3210c_eval_lcd.c**
- BSP_LCD_Init() : **stm3210c_eval_lcd.c**
- BSP_LCD_ReadPixel() : **stm3210c_eval_lcd.c**
- BSP_LCD_SetBackColor() : **stm3210c_eval_lcd.c** , **stm3210c_eval_lcd.h**
- BSP_LCD_SetFont() : **stm3210c_eval_lcd.c**
- BSP_LCD_SetTextColor() : **stm3210c_eval_lcd.h** , **stm3210c_eval_lcd.c**
- BSP_LED_Init() : **stm3210c_eval.c**
- BSP_LED_Off() : **stm3210c_eval.c**
- BSP_LED_On() : **stm3210c_eval.c**
- BSP_LED_Toggle() : **stm3210c_eval.c**
- BSP_PB_GetState() : **stm3210c_eval.c**
- BSP_PB_Init() : **stm3210c_eval.c**
- BSP_SD_Erase() : **stm3210c_eval_sd.c**
- BSP_SD_GetCardInfo() : **stm3210c_eval_sd.c**
- BSP_SD_GetStatus() : **stm3210c_eval_sd.c**
- BSP_SD_Init() : **stm3210c_eval_sd.c**
- BSP_SD_IsDetected() : **stm3210c_eval_sd.c**
- BSP_SD_ReadBlocks() : **stm3210c_eval_sd.c**
- BSP_SD_WriteBlocks() : **stm3210c_eval_sd.c**

- BSP_TS_GetState() : **stm3210c_eval_ts.c**
- BSP_TS_Init() : **stm3210c_eval_ts.c**
- BSP_TS_ITClear() : **stm3210c_eval_ts.c**
- BSP_TS_ITConfig() : **stm3210c_eval_ts.c**
- BSP_TS_ITGetStatus() : **stm3210c_eval_ts.c**

- e -

- EEPROM_I2C_Init() : **stm3210c_eval_eeprom.c**
- EEPROM_I2C_IO_Init() : **stm3210c_eval.c** , **stm3210c_eval_eeprom.h**
- EEPROM_I2C_IO_IsDeviceReady() : **stm3210c_eval.c** , **stm3210c_eval_eeprom.h**
- EEPROM_I2C_IO_ReadData() : **stm3210c_eval.c** , **stm3210c_eval_eeprom.h**
- EEPROM_I2C_IO_WriteData() : **stm3210c_eval_eeprom.h** , **stm3210c_eval.c**
- EEPROM_I2C_ReadBuffer() : **stm3210c_eval_eeprom.c**
- EEPROM_I2C_WaitEepromStandbyState() : **stm3210c_eval_eeprom.c**
- EEPROM_I2C_WritePage() : **stm3210c_eval_eeprom.c**

- h -

- HAL_I2S_ErrorCallback() : **stm3210c_eval_audio.c**
- HAL_I2S_TxCpltCallback() : **stm3210c_eval_audio.c**
- HAL_I2S_TxHalfCpltCallback() : **stm3210c_eval_audio.c**

- i -

- I2Cx_Error() : **stm3210c_eval.c**
- I2Cx_Init() : **stm3210c_eval.c**
- I2Cx_IsDeviceReady() : **stm3210c_eval.c**
- I2Cx_ITConfig() : **stm3210c_eval.c**
- I2Cx_MspInit() : **stm3210c_eval.c**
- I2Cx_ReadBuffer() : **stm3210c_eval.c**
- I2Cx_ReadData() : **stm3210c_eval.c**
- I2Cx_ReadMultiple() : **stm3210c_eval.c**

- I2Cx_WriteBuffer() : **stm3210c_eval.c**
- I2Cx_WriteData() : **stm3210c_eval.c**
- I2SOUT_Init() : **stm3210c_eval_audio.c**
- I2SOUT_MspInit() : **stm3210c_eval_audio.c**
- IOE_Delay() : **stm3210c_eval.c**
- IOE_Init() : **stm3210c_eval.c**
- IOE_ITConfig() : **stm3210c_eval.c**
- IOE_Read() : **stm3210c_eval.c**
- IOE_ReadMultiple() : **stm3210c_eval.c**
- IOE_Write() : **stm3210c_eval.c**

- l -

- LCD_Delay() : **stm3210c_eval.c**
- LCD_DrawChar() : **stm3210c_eval_lcd.c**
- LCD_DrawPixel() : **stm3210c_eval_lcd.c**
- LCD_IO_Init() : **stm3210c_eval.c**
- LCD_IO_ReadData() : **stm3210c_eval.c**
- LCD_IO_WriteMultipleData() : **stm3210c_eval.c**
- LCD_IO_WriteReg() : **stm3210c_eval.c**
- LCD_SetDisplayWindow() : **stm3210c_eval_lcd.c**

- s -

- SD_GetCIDRegister() : **stm3210c_eval_sd.c**
- SD_GetCSDRegister() : **stm3210c_eval_sd.c**
- SD_GetDataResponse() : **stm3210c_eval_sd.c**
- SD_GoIdleState() : **stm3210c_eval_sd.c**
- SD_IO_Init() : **stm3210c_eval.c** , **stm3210c_eval_sd.h**
- SD_IO_ReadByte() : **stm3210c_eval_sd.h** , **stm3210c_eval.c**
- SD_IO_WaitResponse() : **stm3210c_eval.c** , **stm3210c_eval_sd.h**
- SD_IO_WriteByte() : **stm3210c_eval.c** , **stm3210c_eval_sd.h**
- SD_IO_WriteCmd() : **stm3210c_eval.c** , **stm3210c_eval_sd.h**
- SD_IO_WriteDummy() : **stm3210c_eval_sd.h** , **stm3210c_eval.c**
- SD_SendCmd() : **stm3210c_eval_sd.c**
- SPIx_Error() : **stm3210c_eval.c**
- SPIx_Init() : **stm3210c_eval.c**

- SPIx_MspInit() : **stm3210c_eval.c**
- SPIx_Read() : **stm3210c_eval.c**
- SPIx_Write() : **stm3210c_eval.c**

- t -

- TSENSOR_IO_Init() : **stm3210c_eval.c**
- TSENSOR_IO_IsDeviceReady() : **stm3210c_eval.c**
- TSENSOR_IO_Read() : **stm3210c_eval.c**
- TSENSOR_IO_Write() : **stm3210c_eval.c**

STM3210C_EVAL BSP User Manual

- a -

- AcceleroDrv : **stm3210c_eval_accelerometer.c**

- b -

- bitmap : **stm3210c_eval_lcd.c**
- BUTTON_IRQn : **stm3210c_eval.c**
- BUTTON_PIN : **stm3210c_eval.c**
- BUTTON_PORT : **stm3210c_eval.c**

- c -

- COM_RX_PIN : **stm3210c_eval.c**
- COM_RX_PORT : **stm3210c_eval.c**
- COM_TX_PIN : **stm3210c_eval.c**
- COM_TX_PORT : **stm3210c_eval.c**
- COM_USART : **stm3210c_eval.c**

- d -

- DrawProp : **stm3210c_eval_lcd.c**

- e -

- EEPROM_I2C_Drv : **stm3210c_eval_eeprom.c**
- EEPROM_SelectedDevice : **stm3210c_eval_eeprom.c**
- EEPROMAddress : **stm3210c_eval_eeprom.c**
- EEPROMDataRead : **stm3210c_eval_eeprom.c**
- EEPROMDataWrite : **stm3210c_eval_eeprom.c**
- EEPROMPageSize : **stm3210c_eval_eeprom.c**

- h -

- hAudioOutI2s : **stm3210c_eval_audio.c**
- heval_I2c : **stm3210c_eval.c**
- heval_Spi : **stm3210c_eval.c**

- i -

- I2cxTimeout : **stm3210c_eval.c**
- io1_driver : **stm3210c_eval_io.c**
- io2_driver : **stm3210c_eval_io.c**

- l -

- lcd_drv : **stm3210c_eval_lcd.c**
- LED_PIN : **stm3210c_eval.c**
- LED_PORT : **stm3210c_eval.c**

- p -

- pAudioDrv : **stm3210c_eval_audio.c**

- s -

- SdStatus : **stm3210c_eval_sd.c**
- SpixTimeout : **stm3210c_eval.c**

- t -

- ts_driver : **stm3210c_eval_ts.c**

- ts_orientation : **stm3210c_eval_ts.c**
- ts_x_boundary : **stm3210c_eval_ts.c**
- ts_y_boundary : **stm3210c_eval_ts.c**

STM3210C_EVAL BSP User Manual

- pPoint : **stm3210c_eval_lcd.h**

Generated on Thu Dec 10 2015 17:13:53 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

- ACCELERO_StatusTypeDef : **stm3210c_eval_accelerometer.h**
- Button_TypeDef : **stm3210c_eval.h**
- ButtonMode_TypeDef : **stm3210c_eval.h**
- COM_TypeDef : **stm3210c_eval.h**
- IO_StatusTypeDef : **stm3210c_eval_io.h**
- JOYMode_TypeDef : **stm3210c_eval.h**
- JOYState_TypeDef : **stm3210c_eval.h**
- Led_TypeDef : **stm3210c_eval.h**
- Line_ModeTypdef : **stm3210c_eval_lcd.h**
- SD_Info : **stm3210c_eval_sd.h**
- TS_StatusTypeDef : **stm3210c_eval_ts.h**

STM3210C_EVAL BSP User Manual

- a -

- ACCELERO_ERROR : **stm3210c_eval_accelerometer.h**
- ACCELERO_OK : **stm3210c_eval_accelerometer.h**
- ACCELERO_TIMEOUT : **stm3210c_eval_accelerometer.h**

- b -

- BUTTON_KEY : **stm3210c_eval.h**
- BUTTON_MODE_EXTI : **stm3210c_eval.h**
- BUTTON_MODE_GPIO : **stm3210c_eval.h**
- BUTTON_TAMPER : **stm3210c_eval.h**
- BUTTON_WAKEUP : **stm3210c_eval.h**

- c -

- CENTER_MODE : **stm3210c_eval_lcd.h**
- COM1 : **stm3210c_eval.h**
- COM2 : **stm3210c_eval.h**

- i -

- IO_ERROR : **stm3210c_eval_io.h**

- IO_OK : **stm3210c_eval_io.h**
- IO_TIMEOUT : **stm3210c_eval_io.h**

- j -

- JOY_DOWN : **stm3210c_eval.h**
- JOY_LEFT : **stm3210c_eval.h**
- JOY_MODE_EXTI : **stm3210c_eval.h**
- JOY_MODE_GPIO : **stm3210c_eval.h**
- JOY_NONE : **stm3210c_eval.h**
- JOY_RIGHT : **stm3210c_eval.h**
- JOY_SEL : **stm3210c_eval.h**
- JOY_UP : **stm3210c_eval.h**

- l -

- LED1 : **stm3210c_eval.h**
- LED2 : **stm3210c_eval.h**
- LED3 : **stm3210c_eval.h**
- LED4 : **stm3210c_eval.h**
- LED_BLUE : **stm3210c_eval.h**
- LED_GREEN : **stm3210c_eval.h**
- LED_ORANGE : **stm3210c_eval.h**
- LED_RED : **stm3210c_eval.h**
- LEFT_MODE : **stm3210c_eval_lcd.h**

- r -

- RIGHT_MODE : **stm3210c_eval_lcd.h**

- s -

- SD_ADDRESS_ERROR : **stm3210c_eval_sd.h**
- SD_COM_CRC_ERROR : **stm3210c_eval_sd.h**
- SD_DATA_CRC_ERROR : **stm3210c_eval_sd.h**
- SD_DATA_OK : **stm3210c_eval_sd.h**
- SD_DATA_OTHER_ERROR : **stm3210c_eval_sd.h**
- SD_DATA_WRITE_ERROR : **stm3210c_eval_sd.h**

- SD_ERASE_RESET : **stm3210c_eval_sd.h**
- SD_ERASE_SEQUENCE_ERROR : **stm3210c_eval_sd.h**
- SD_ILLEGAL_COMMAND : **stm3210c_eval_sd.h**
- SD_IN_IDLE_STATE : **stm3210c_eval_sd.h**
- SD_PARAMETER_ERROR : **stm3210c_eval_sd.h**
- SD_RESPONSE_FAILURE : **stm3210c_eval_sd.h**
- SD_RESPONSE_NO_ERROR : **stm3210c_eval_sd.h**

- t -

- TS_ERROR : **stm3210c_eval_ts.h**
- TS_OK : **stm3210c_eval_ts.h**
- TS_TIMEOUT : **stm3210c_eval_ts.h**

STM3210C_EVAL BSP User Manual

- _ -

- __STM3210C_EVAL_BSP_VERSION : **stm3210c_eval.c**
- __STM3210C_EVAL_BSP_VERSION_MAIN : **stm3210c_eval.c**
- __STM3210C_EVAL_BSP_VERSION_RC : **stm3210c_eval.c**
- __STM3210C_EVAL_BSP_VERSION_SUB1 : **stm3210c_eval.c**
- __STM3210C_EVAL_BSP_VERSION_SUB2 : **stm3210c_eval.c**

Generated on Thu Dec 10 2015 17:13:53 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

- a -

STM3210C_EVAL BSP User Manual

- b -

- BSP_EEPROM_M24C08 : **stm3210c_eval_eeprom.h**
- BSP_EEPROM_M24C64_32 : **stm3210c_eval_eeprom.h**
- BUTTONn : **stm3210c_eval.h**
- BUTTONx_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- BUTTONx_GPIO_CLK_ENABLE : **stm3210c_eval.h**

STM3210C_EVAL BSP User Manual

- c -

- COMn : **stm3210c_eval.h**
- COMx_CLK_DISABLE : **stm3210c_eval.h**
- COMx_CLK_ENABLE : **stm3210c_eval.h**
- COMx_RX_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- COMx_RX_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- COMx_TX_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- COMx_TX_GPIO_CLK_ENABLE : **stm3210c_eval.h**

STM3210C_EVAL BSP User Manual

- d -

- DMA_MAX : **stm3210c_eval_audio.h**
- DMA_MAX_SZE : **stm3210c_eval_audio.h**

STM3210C_EVAL BSP User Manual

- e -

- EEPROM_ADDRESS_M24C08_BLOCK0 : **stm3210c_eval_eeprom.h**
- EEPROM_ADDRESS_M24C08_BLOCK1 : **stm3210c_eval_eeprom.h**
- EEPROM_ADDRESS_M24C08_BLOCK2 : **stm3210c_eval_eeprom.h**
- EEPROM_ADDRESS_M24C08_BLOCK3 : **stm3210c_eval_eeprom.h**
- EEPROM_ADDRESS_M24C64_32 : **stm3210c_eval_eeprom.h**
- EEPROM_FAIL : **stm3210c_eval_eeprom.h**
- EEPROM_MAX_TRIALS : **stm3210c_eval_eeprom.h**
- EEPROM_OK : **stm3210c_eval_eeprom.h**
- EEPROM_PAGESIZE_M24C08 : **stm3210c_eval_eeprom.h**
- EEPROM_PAGESIZE_M24C64_32 : **stm3210c_eval_eeprom.h**
- EEPROM_TIMEOUT : **stm3210c_eval_eeprom.h**
- EVAL_COM1 : **stm3210c_eval.h**
- EVAL_COM1_CLK_DISABLE : **stm3210c_eval.h**
- EVAL_COM1_CLK_ENABLE : **stm3210c_eval.h**
- EVAL_COM1_IRQn : **stm3210c_eval.h**
- EVAL_COM1_RX_GPIO_CLK_DISABLE : **stm3210c_eval.h**

- EVAL_COM1_RX_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- EVAL_COM1_RX_GPIO_PORT : **stm3210c_eval.h**
- EVAL_COM1_RX_PIN : **stm3210c_eval.h**
- EVAL_COM1_TX_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- EVAL_COM1_TX_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- EVAL_COM1_TX_GPIO_PORT : **stm3210c_eval.h**
- EVAL_COM1_TX_PIN : **stm3210c_eval.h**
- EVAL_I2Cx : **stm3210c_eval.h**
- EVAL_I2Cx_CLK_ENABLE : **stm3210c_eval.h**
- EVAL_I2Cx_ER_IRQHandler : **stm3210c_eval.h**
- EVAL_I2Cx_ER_IRQn : **stm3210c_eval.h**
- EVAL_I2Cx_EV_IRQHandler : **stm3210c_eval.h**
- EVAL_I2Cx_EV_IRQn : **stm3210c_eval.h**
- EVAL_I2Cx_FORCE_RESET : **stm3210c_eval.h**
- EVAL_I2Cx_RELEASE_RESET : **stm3210c_eval.h**
- EVAL_I2Cx_SCL_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- EVAL_I2Cx_SCL_GPIO_PORT : **stm3210c_eval.h**
- EVAL_I2Cx_SCL_PIN : **stm3210c_eval.h**
- EVAL_I2Cx_SDA_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- EVAL_I2Cx_SDA_GPIO_PORT : **stm3210c_eval.h**
- EVAL_I2Cx_SDA_PIN : **stm3210c_eval.h**
- EVAL_I2Cx_TIMEOUT_MAX : **stm3210c_eval.h**
- EVAL_SPIx : **stm3210c_eval.h**
- EVAL_SPIx_CLK_ENABLE : **stm3210c_eval.h**
- EVAL_SPIx_MISO_MOSI_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- EVAL_SPIx_MISO_MOSI_GPIO_PORT : **stm3210c_eval.h**
- EVAL_SPIx_MISO_PIN : **stm3210c_eval.h**
- EVAL_SPIx_MOSI_PIN : **stm3210c_eval.h**
- EVAL_SPIx_SCK_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- EVAL_SPIx_SCK_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- EVAL_SPIx_SCK_GPIO_PORT : **stm3210c_eval.h**
- EVAL_SPIx_SCK_PIN : **stm3210c_eval.h**
- EVAL_SPIx_TIMEOUT_MAX : **stm3210c_eval.h**

Generated on Thu Dec 10 2015 17:13:53 for STM3210C_EVAL BSP
User Manual by  1.7.6.1

STM3210C_EVAL BSP User Manual

- i -

- I2SOUT : **stm3210c_eval_audio.h**
- I2SOUT_CLK_ENABLE : **stm3210c_eval_audio.h**
- I2SOUT_DMAx_CHANNEL : **stm3210c_eval_audio.h**
- I2SOUT_DMAx_CLK_ENABLE : **stm3210c_eval_audio.h**
- I2SOUT_DMAx_IRQ : **stm3210c_eval_audio.h**
- I2SOUT_DMAx_MEM_DATA_SIZE : **stm3210c_eval_audio.h**
- I2SOUT_DMAx_PERIPH_DATA_SIZE : **stm3210c_eval_audio.h**
- I2SOUT_IRQHandler : **stm3210c_eval_audio.h**
- I2SOUT_MCK_CLK_ENABLE : **stm3210c_eval_audio.h**
- I2SOUT_MCK_GPIO_PORT : **stm3210c_eval_audio.h**
- I2SOUT_MCK_PIN : **stm3210c_eval_audio.h**
- I2SOUT_SCK_PIN : **stm3210c_eval_audio.h**
- I2SOUT_SCK_SD_CLK_ENABLE : **stm3210c_eval_audio.h**
- I2SOUT_SCK_SD_GPIO_PORT : **stm3210c_eval_audio.h**
- I2SOUT_SD_PIN : **stm3210c_eval_audio.h**
- I2SOUT_WS_CLK_ENABLE : **stm3210c_eval_audio.h**
- I2SOUT_WS_GPIO_PORT : **stm3210c_eval_audio.h**
- I2SOUT_WS_PIN : **stm3210c_eval_audio.h**
- INTERNAL_BUFF_SIZE : **stm3210c_eval_audio.h**
- IO1_I2C_ADDRESS : **stm3210c_eval.h**

- IO1_PIN_0 : **stm3210c_eval_io.h**
- IO1_PIN_1 : **stm3210c_eval_io.h**
- IO1_PIN_2 : **stm3210c_eval_io.h**
- IO1_PIN_3 : **stm3210c_eval_io.h**
- IO1_PIN_4 : **stm3210c_eval_io.h**
- IO1_PIN_5 : **stm3210c_eval_io.h**
- IO1_PIN_6 : **stm3210c_eval_io.h**
- IO1_PIN_7 : **stm3210c_eval_io.h**
- IO1_PIN_ALL : **stm3210c_eval_io.h**
- IO1_PIN_OFFSET : **stm3210c_eval_io.h**
- IO2_I2C_ADDRESS : **stm3210c_eval.h**
- IO2_PIN_0 : **stm3210c_eval_io.h**
- IO2_PIN_1 : **stm3210c_eval_io.h**
- IO2_PIN_2 : **stm3210c_eval_io.h**
- IO2_PIN_3 : **stm3210c_eval_io.h**
- IO2_PIN_4 : **stm3210c_eval_io.h**
- IO2_PIN_5 : **stm3210c_eval_io.h**
- IO2_PIN_6 : **stm3210c_eval_io.h**
- IO2_PIN_7 : **stm3210c_eval_io.h**
- IO2_PIN_ALL : **stm3210c_eval_io.h**
- IO2_PIN_OFFSET : **stm3210c_eval_io.h**
- IOE_IT_EXTI_IRQHANDLER : **stm3210c_eval.h**
- IOE_IT_EXTI_IRQn : **stm3210c_eval.h**
- IOE_IT_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- IOE_IT_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- IOE_IT_GPIO_PORT : **stm3210c_eval.h**
- IOE_IT_PIN : **stm3210c_eval.h**

STM3210C_EVAL BSP User Manual

- j -

- JOY_ALL_PINS : **stm3210c_eval.h**
- JOY_DOWN_PIN : **stm3210c_eval.h**
- JOY_LEFT_PIN : **stm3210c_eval.h**
- JOY_NONE_PIN : **stm3210c_eval.h**
- JOY_RIGHT_PIN : **stm3210c_eval.h**
- JOY_SEL_PIN : **stm3210c_eval.h**
- JOY_UP_PIN : **stm3210c_eval.h**

STM3210C_EVAL BSP User Manual

- k -

- KEY_BUTTON_EXTI_IRQn : **stm3210c_eval.h**
- KEY_BUTTON_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- KEY_BUTTON_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- KEY_BUTTON_GPIO_PORT : **stm3210c_eval.h**
- KEY_BUTTON_PIN : **stm3210c_eval.h**

STM3210C_EVAL BSP User Manual

- l -

- L1S302DL_I2C_ADDRESS : **stm3210c_eval.h**
- LCD_COLOR_BLACK : **stm3210c_eval_lcd.h**
- LCD_COLOR_BLUE : **stm3210c_eval_lcd.h**
- LCD_COLOR_BROWN : **stm3210c_eval_lcd.h**
- LCD_COLOR_CYAN : **stm3210c_eval_lcd.h**
- LCD_COLOR_DARKBLUE : **stm3210c_eval_lcd.h**
- LCD_COLOR_DARKCYAN : **stm3210c_eval_lcd.h**
- LCD_COLOR_DARKGRAY : **stm3210c_eval_lcd.h**
- LCD_COLOR_DARKGREEN : **stm3210c_eval_lcd.h**
- LCD_COLOR_DARKMAGENTA : **stm3210c_eval_lcd.h**
- LCD_COLOR_DARKRED : **stm3210c_eval_lcd.h**
- LCD_COLOR_DARKYELLOW : **stm3210c_eval_lcd.h**
- LCD_COLOR_GRAY : **stm3210c_eval_lcd.h**
- LCD_COLOR_GREEN : **stm3210c_eval_lcd.h**
- LCD_COLOR_LIGHTBLUE : **stm3210c_eval_lcd.h**
- LCD_COLOR_LIGHTCYAN : **stm3210c_eval_lcd.h**
- LCD_COLOR_LIGHTGRAY : **stm3210c_eval_lcd.h**
- LCD_COLOR_LIGHTGREEN : **stm3210c_eval_lcd.h**
- LCD_COLOR_LIGHTMAGENTA : **stm3210c_eval_lcd.h**
- LCD_COLOR_LIGHTRED : **stm3210c_eval_lcd.h**

- LCD_COLOR_LIGHTYELLOW : **stm3210c_eval_lcd.h**
- LCD_COLOR_MAGENTA : **stm3210c_eval_lcd.h**
- LCD_COLOR_ORANGE : **stm3210c_eval_lcd.h**
- LCD_COLOR_RED : **stm3210c_eval_lcd.h**
- LCD_COLOR_WHITE : **stm3210c_eval_lcd.h**
- LCD_COLOR_YELLOW : **stm3210c_eval_lcd.h**
- LCD_CS_HIGH : **stm3210c_eval.h**
- LCD_CS_LOW : **stm3210c_eval.h**
- LCD_DEFAULT_FONT : **stm3210c_eval_lcd.h**
- LCD_ERROR : **stm3210c_eval_lcd.h**
- LCD_NCS_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- LCD_NCS_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- LCD_NCS_GPIO_PORT : **stm3210c_eval.h**
- LCD_NCS_PIN : **stm3210c_eval.h**
- LCD_OK : **stm3210c_eval_lcd.h**
- LCD_READ_REG : **stm3210c_eval.c**
- LCD_TIMEOUT : **stm3210c_eval_lcd.h**
- LCD_WRITE_REG : **stm3210c_eval.c**
- LED1_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- LED1_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- LED1_GPIO_PORT : **stm3210c_eval.h**
- LED1_PIN : **stm3210c_eval.h**
- LED2_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- LED2_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- LED2_GPIO_PORT : **stm3210c_eval.h**
- LED2_PIN : **stm3210c_eval.h**
- LED3_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- LED3_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- LED3_GPIO_PORT : **stm3210c_eval.h**
- LED3_PIN : **stm3210c_eval.h**
- LED4_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- LED4_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- LED4_GPIO_PORT : **stm3210c_eval.h**
- LED4_PIN : **stm3210c_eval.h**
- LEDn : **stm3210c_eval.h**
- LEDx_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- LEDx_GPIO_CLK_ENABLE : **stm3210c_eval.h**

Generated on Thu Dec 10 2015 17:13:53 for STM3210C_EVAL BSP
User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

- m -

- MAX_HEIGHT_FONT : **stm3210c_eval_lcd.c**
- MAX_WIDTH_FONT : **stm3210c_eval_lcd.c**
- MEMS_ALL_PINS : **stm3210c_eval.h**
- MEMS_INT1_PIN : **stm3210c_eval.h**
- MEMS_INT2_PIN : **stm3210c_eval.h**
- MII_INT_PIN : **stm3210c_eval.h**
- MSD_ERROR : **stm3210c_eval_sd.h**
- MSD_OK : **stm3210c_eval_sd.h**
- MULTIPLEBYTE_CMD : **stm3210c_eval.h**

STM3210C_EVAL BSP User Manual

- o -

- OFFSET_BITMAP : **stm3210c_eval_lcd.c**

STM3210C_EVAL BSP User Manual

- p -

- POLY_X : **stm3210c_eval_lcd.c**
- POLY_Y : **stm3210c_eval_lcd.c**

STM3210C_EVAL BSP User Manual

- r -

- READ_STATUS : **stm3210c_eval.c**
- READWRITE_CMD : **stm3210c_eval.h**

STM3210C_EVAL BSP User Manual

- s -

- SD_BLOCK_SIZE : **stm3210c_eval_sd.h**
- SD_CMD_CLR_WRITE_PROT : **stm3210c_eval_sd.h**
- SD_CMD_ERASE : **stm3210c_eval_sd.h**
- SD_CMD_ERASE_GRP_END : **stm3210c_eval_sd.h**
- SD_CMD_ERASE_GRP_START : **stm3210c_eval_sd.h**
- SD_CMD_GO_IDLE_STATE : **stm3210c_eval_sd.h**
- SD_CMD_PROG_CSD : **stm3210c_eval_sd.h**
- SD_CMD_READ_MULT_BLOCK : **stm3210c_eval_sd.h**
- SD_CMD_READ_SINGLE_BLOCK : **stm3210c_eval_sd.h**
- SD_CMD_SD_ERASE_GRP_END : **stm3210c_eval_sd.h**
- SD_CMD_SD_ERASE_GRP_START : **stm3210c_eval_sd.h**
- SD_CMD_SEND_CID : **stm3210c_eval_sd.h**
- SD_CMD_SEND_CSD : **stm3210c_eval_sd.h**
- SD_CMD_SEND_OP_COND : **stm3210c_eval_sd.h**
- SD_CMD_SEND_STATUS : **stm3210c_eval_sd.h**
- SD_CMD_SEND_WRITE_PROT : **stm3210c_eval_sd.h**
- SD_CMD_SET_BLOCK_COUNT : **stm3210c_eval_sd.h**
- SD_CMD_SET_BLOCKLEN : **stm3210c_eval_sd.h**
- SD_CMD_SET_WRITE_PROT : **stm3210c_eval_sd.h**
- SD_CMD_STOP_TRANSMISSION : **stm3210c_eval_sd.h**

- SD_CMD_UNTAG_ERASE_GROUP : **stm3210c_eval_sd.h**
- SD_CMD_UNTAG_SECTOR : **stm3210c_eval_sd.h**
- SD_CMD_WRITE_MULT_BLOCK : **stm3210c_eval_sd.h**
- SD_CMD_WRITE_SINGLE_BLOCK : **stm3210c_eval_sd.h**
- SD_CS_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- SD_CS_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- SD_CS_GPIO_PORT : **stm3210c_eval.h**
- SD_CS_HIGH : **stm3210c_eval.h**
- SD_CS_LOW : **stm3210c_eval.h**
- SD_CS_PIN : **stm3210c_eval.h**
- SD_DETECT_EXTI_IRQn : **stm3210c_eval.h**
- SD_DETECT_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- SD_DETECT_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- SD_DETECT_GPIO_PORT : **stm3210c_eval.h**
- SD_DETECT_PIN : **stm3210c_eval.h**
- SD_DUMMY_BYTE : **stm3210c_eval.c** , **stm3210c_eval_sd.c**
- SD_NO_RESPONSE_EXPECTED : **stm3210c_eval_sd.c** , **stm3210c_eval.c**
- SD_NOT_PRESENT : **stm3210c_eval_sd.h**
- SD_PRESENT : **stm3210c_eval_sd.h**
- SD_START_DATA_MULTIPLE_BLOCK_READ : **stm3210c_eval_sd.h**
- SD_START_DATA_MULTIPLE_BLOCK_WRITE : **stm3210c_eval_sd.h**
- SD_START_DATA_SINGLE_BLOCK_READ : **stm3210c_eval_sd.h**
- SD_START_DATA_SINGLE_BLOCK_WRITE : **stm3210c_eval_sd.h**
- SD_STOP_DATA_MULTIPLE_BLOCK_WRITE : **stm3210c_eval_sd.h**
- SET_INDEX : **stm3210c_eval.c**
- START_BYTE : **stm3210c_eval.c**

STM3210C_EVAL BSP User Manual

- t -

- TAMPER_BUTTON_EXTI_IRQn : **stm3210c_eval.h**
- TAMPER_BUTTON_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- TAMPER_BUTTON_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- TAMPER_BUTTON_GPIO_PORT : **stm3210c_eval.h**
- TAMPER_BUTTON_PIN : **stm3210c_eval.h**
- TFT_LCD : **stm3210c_eval.c**
- TFT_LCD_BASE : **stm3210c_eval.c**
- TS_I2C_ADDRESS : **stm3210c_eval.h**
- TS_SWAP_NONE : **stm3210c_eval_ts.h**
- TS_SWAP_X : **stm3210c_eval_ts.h**
- TS_SWAP_XY : **stm3210c_eval_ts.h**
- TS_SWAP_Y : **stm3210c_eval_ts.h**

STM3210C_EVAL BSP User Manual

- v -

- VBAT_DIV_PIN : **stm3210c_eval.h**

STM3210C_EVAL BSP User Manual

- w -

- WAKEUP_BUTTON_EXTI_IRQn : **stm3210c_eval.h**
- WAKEUP_BUTTON_GPIO_CLK_DISABLE : **stm3210c_eval.h**
- WAKEUP_BUTTON_GPIO_CLK_ENABLE : **stm3210c_eval.h**
- WAKEUP_BUTTON_GPIO_PORT : **stm3210c_eval.h**
- WAKEUP_BUTTON_PIN : **stm3210c_eval.h**

STM3210C_EVAL BSP User Manual

stm3210c_eval.c File Reference

This file provides a set of firmware functions to manage Leds, push-button and COM ports for STM3210C_EVAL. More...

```
#include "stm3210c_eval.h"
```

Go to the source code of this file.

Data Structures

struct	**TFT_LCD_TypeDef**

Defines

#define	**START_BYTE** 0x70
#define	**SET_INDEX** 0x00
#define	**READ_STATUS** 0x01
#define	**LCD_WRITE_REG** 0x02
#define	**LCD_READ_REG** 0x03
#define	**SD_DUMMY_BYTE** 0xFF
#define	**SD_NO_RESPONSE_EXPECTED** 0x80
#define	**__STM3210C_EVAL_BSP_VERSION_MAIN** (0x06) STM3210C EVAL BSP Driver version number V6.0.1.
#define	**__STM3210C_EVAL_BSP_VERSION_SUB1** (0x00)
#define	**__STM3210C_EVAL_BSP_VERSION_SUB2** (0x01)
#define	**__STM3210C_EVAL_BSP_VERSION_RC** (0x00)
#define	**__STM3210C_EVAL_BSP_VERSION**
#define	**TFT_LCD_BASE** ((uint32_t)(0x60000000 | 0x0C000000))
#define	**TFT_LCD** ((**TFT_LCD_TypeDef** *) **TFT_LCD_BASE**)

Functions

static void	**I2Cx_Init** (void) Eval I2Cx Bus initialization.
static void	**I2Cx_ITConfig** (void) Configures I2C Interrupt.
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_ReadBuffer** (uint16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuffer, uint16_t Length) Reads multiple data on the BUS.
static void	**I2Cx_WriteData** (uint16_t Addr, uint8_t Reg, uint8_t Value) Write a value in a register of the device through BUS.
static HAL_StatusTypeDef	**I2Cx_WriteBuffer** (uint16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuffer, uint16_t Length) Write a value in a register of the device through BUS.
static uint8_t	**I2Cx_ReadData** (uint16_t Addr, uint8_t Reg) Read a value in a register of the device through BUS.
static HAL_StatusTypeDef	**I2Cx_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.
static void	**I2Cx_Error** (uint8_t Addr) Manages error callback by re-initializing I2C.

static void	**I2Cx_MspInit** (I2C_HandleTypeDef *hi2c) Eval I2Cx MSP Initialization.
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	**EEPROM_I2C_IO_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_I2C_IO_WriteData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Write data to I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_I2C_IO_ReadData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Read data from I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_I2C_IO_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.
void	**TSENSOR_IO_Init** (void) Initializes peripherals used by the I2C Temperature Sensor driver.

void	**TSENSOR_IO_Write** (uint16_t DevAddress, uint8_t *pBuffer, uint8_t WriteAddr, uint16_t Length) Writes one byte to the TSENSOR.
void	**TSENSOR_IO_Read** (uint16_t DevAddress, uint8_t *pBuffer, uint8_t ReadAddr, uint16_t Length) Reads one byte from the TSENSOR.
uint16_t	**TSENSOR_IO_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if Temperature Sensor is ready for communication.
void	**AUDIO_IO_Init** (void) Initializes Audio low level.
void	**AUDIO_IO_Write** (uint8_t Addr, uint8_t Reg, uint8_t Value) Writes a single data.
uint8_t	**AUDIO_IO_Read** (uint8_t Addr, uint8_t Reg) Reads a single data.
void	**ACCELERO_IO_Init** (void) Configures ACCELEROMETER SPI interface.
void	**ACCELERO_IO_ITConfig** (void) Configures ACCELERO INT2 config.
void	**ACCELERO_IO_Write** (uint8_t *pBuffer, uint8_t WriteAddr, uint16_t NumByteToWrite) Writes one byte to the ACCELEROMETER.
void	**ACCELERO_IO_Read** (uint8_t *pBuffer, uint8_t ReadAddr, uint16_t NumByteToRead) Reads a block of data from the ACCELEROMETER.

static void	**SPIx_Init** (void) Initializes SPI HAL.
static void	**SPIx_Write** (uint8_t Value) SPI Write a byte to device.
static uint32_t	**SPIx_Read** (void) SPI Read 4 bytes from device.
static void	**SPIx_Error** (void) SPI error treatment function.
static void	**SPIx_MspInit** (SPI_HandleTypeDef *hspi) Initializes SPI MSP.
void	**LCD_IO_Init** (void) Configures the LCD_SPI interface.
void	**LCD_IO_WriteMultipleData** (uint8_t *pData, uint32_t Size) Write register value.
void	**LCD_IO_WriteReg** (uint8_t Reg) register address.
uint16_t	**LCD_IO_ReadData** (uint16_t Reg) Read register value.
void	**LCD_Delay** (uint32_t Delay) Wait for loop in ms.
void	**SD_IO_Init** (void) Initializes the SD Card and put it into StandBy State (Ready for data transfer).
HAL_StatusTypeDef	**SD_IO_WriteCmd** (uint8_t Cmd, uint32_t Arg, uint8_t Crc, uint8_t Response) Send 5 bytes command to the SD card and get response.
HAL_StatusTypeDef	**SD_IO_WaitResponse** (uint8_t Response) Wait response from the SD card.
void	**SD_IO_WriteDummy** (void)

	Send dummy byte with CS High.
void	**SD_IO_WriteByte** (uint8_t Data) Write a byte on the SD.
uint8_t	**SD_IO_ReadByte** (void) Read a byte from the SD.
uint32_t	**BSP_GetVersion** (void) This method returns the STM3210C EVAL BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LED GPIO.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** Button_Mode) Configures push button GPIO and EXTI Line.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
uint8_t	**BSP_JOY_Init** (**JOYMode_TypeDef** Joy_Mode) Configures joystick GPIO and EXTI modes.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the current joystick status.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) Configures COM port.

Variables

GPIO_TypeDef *	**LED_PORT** [**LEDn**] LED variables.
const uint16_t	**LED_PIN** [**LEDn**]
GPIO_TypeDef *	**BUTTON_PORT** [**BUTTONn**] BUTTON variables.
const uint16_t	**BUTTON_PIN** [**BUTTONn**]
const uint16_t	**BUTTON_IRQn** [**BUTTONn**]
USART_TypeDef *	**COM_USART** [**COMn**] = {**EVAL_COM1**} COM variables.
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**}
uint32_t	**SpixTimeout** = **EVAL_SPIx_TIMEOUT_MAX** BUS variables.
static SPI_HandleTypeDef	**heval_Spi**
uint32_t	**I2cxTimeout** = **EVAL_I2Cx_TIMEOUT_MAX**
I2C_HandleTypeDef	**heval_I2c**

Detailed Description

This file provides a set of firmware functions to manage Leds, push-button and COM ports for STM3210C_EVAL.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

Definition in file **stm3210c_eval.c**.

STM3210C_EVAL BSP User Manual

Defines | Enumerations | Functions

stm3210c_eval.h File Reference

This file contains definitions for STM3210C_EVAL's LEDs, push-buttons and COM ports hardware resources. More...

```
#include "stm32f1xx_hal.h" #include "stm3210c_eval_io.h"
```

Go to the source code of this file.

Defines

#define **LEDn** 4

#define **LED1_PIN** GPIO_PIN_7 /* PD.07*/

#define **LED1_GPIO_PORT** GPIOD

#define **LED1_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOD_CLK_E

#define **LED1_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOD_CLK_D

#define **LED2_PIN** GPIO_PIN_13 /* PD.13*/

#define **LED2_GPIO_PORT** GPIOD

#define **LED2_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOD_CLK_E

#define **LED2_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOD_CLK_D

#define **LED3_PIN** GPIO_PIN_3 /* PD.03*/

#define **LED3_GPIO_PORT** GPIOD

#define **LED3_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOD_CLK_E

#define **LED3_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOD_CLK_D

#define **LED4_PIN** GPIO_PIN_4 /* PD.04*/

#define **LED4_GPIO_PORT** GPIOD

#define **LED4_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOD_CLK_E

#define **LED4_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOD_CLK_D

#define **LEDx_GPIO_CLK_ENABLE**(__LED__)

#define **LEDx_GPIO_CLK_DISABLE**(__LED__)

#define **BUTTONn** 3

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

Tamper push-button.

#define **TAMPER_BUTTON_GPIO_PORT** GPIOC

#define **TAMPER_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_G

#define **TAMPER_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_C

#define **TAMPER_BUTTON_EXTI_IRQn** EXTI15_10_IRQn

#define **KEY_BUTTON_PIN** GPIO_PIN_9 /* PB.09*/

Key push-button.

#define **KEY_BUTTON_GPIO_PORT** GPIOB

#define **KEY_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB

#define **KEY_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_GPIO

#define **KEY_BUTTON_EXTI_IRQn** EXTI9_5_IRQn

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

#define **WAKEUP_BUTTON_GPIO_PORT** GPIOA

#define **WAKEUP_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_G

#define **WAKEUP_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_(

#define **WAKEUP_BUTTON_EXTI_IRQn** EXTI0_IRQn

#define **BUTTONx_GPIO_CLK_ENABLE**(__BUTTON__)

#define **BUTTONx_GPIO_CLK_DISABLE**(__BUTTON__)

#define **JOY_SEL_PIN** (**IO2_PIN_7**) /* IO_Expander_2 */
IO Pins definition.

#define **JOY_DOWN_PIN** (**IO2_PIN_6**) /* IO_Expander_2 */

#define **JOY_LEFT_PIN** (**IO2_PIN_5**) /* IO_Expander_2 */

#define **JOY_RIGHT_PIN** (**IO2_PIN_4**) /* IO_Expander_2 */

#define **JOY_UP_PIN** (**IO2_PIN_3**) /* IO_Expander_2 */

#define **JOY_NONE_PIN** **JOY_ALL_PINS**

#define **JOY_ALL_PINS** (**JOY_SEL_PIN** | **JOY_DOWN_PIN** | **JOY_L** | **JOY_UP_PIN**)

#define **MEMS_INT1_PIN** (**IO1_PIN_3**) /* IO_Expander_1 */ /* Input */

#define **MEMS_INT2_PIN** (**IO1_PIN_2**) /* IO_Expander_1 */ /* Input */

#define **MEMS_ALL_PINS** (**MEMS_INT1_PIN** | **MEMS_INT2_PIN**)

#define **AUDIO_RESET_PIN** (**IO2_PIN_2**) /* IO_Expander_2 */ /* Out

#define **MII_INT_PIN** (**IO2_PIN_0**) /* IO_Expander_2 */ /* Output */

#define **VBAT_DIV_PIN** (**IO1_PIN_0**) /* IO_Expander_1 */ /* Output */

#define **COMn** 1

#define **EVAL_COM1** USART2
Definition for COM port1, connected to USART2.

#define **EVAL_COM1_CLK_ENABLE**() __HAL_RCC_USART2_CLK_

#define **EVAL_COM1_CLK_DISABLE**() __HAL_RCC_USART2_CLK

#define **AFIOCOM1_CLK_ENABLE**() __HAL_RCC_AFIO_CLK_ENA

#define **AFIOCOM1_CLK_DISABLE**() __HAL_RCC_AFIO_CLK_DIS

#define **EVAL_COM1_TX_PIN** GPIO_PIN_5 /* PD.05*/

#define **EVAL_COM1_TX_GPIO_PORT** GPIOD

#define **EVAL_COM1_TX_GPIO_CLK_ENABLE**() __HAL_RCC_GPI

#define **EVAL_COM1_TX_GPIO_CLK_DISABLE**() __HAL_RCC_GP

#define **EVAL_COM1_RX_PIN** GPIO_PIN_6 /* PD.06*/

#define **EVAL_COM1_RX_GPIO_PORT** GPIOD

#define **EVAL_COM1_RX_GPIO_CLK_ENABLE**() __HAL_RCC_GPI

#define **EVAL_COM1_RX_GPIO_CLK_DISABLE**() __HAL_RCC_GP

#define **EVAL_COM1_IRQn** USART2_IRQn

#define **COMx_CLK_ENABLE**(__INDEX__) do { if((__INDEX__) == **C**
EVAL_COM1_CLK_ENABLE();} while(0)

#define **COMx_CLK_DISABLE**(__INDEX__) (((__INDEX__) == **COM**
EVAL_COM1_CLK_DISABLE() : 0)

#define **AFIOCOMx_CLK_ENABLE**(__INDEX__) do { if((__INDEX__
AFIOCOM1_CLK_ENABLE();} while(0)

#define **AFIOCOMx_CLK_DISABLE**(__INDEX__) (((__INDEX__) ==
AFIOCOM1_CLK_DISABLE() : 0)

#define **AFIOCOMx_REMAP**(__INDEX__) (((__INDEX__) == **COM1**)
(AFIO_MAPR_USART2_REMAP)) : 0)

#define **COMx_TX_GPIO_CLK_ENABLE**(__INDEX__) do { if((__IND
EVAL_COM1_TX_GPIO_CLK_ENABLE();} while(0)

#define **COMx_TX_GPIO_CLK_DISABLE**(__INDEX__) (((__INDEX_
EVAL_COM1_TX_GPIO_CLK_DISABLE() : 0)

#define **COMx_RX_GPIO_CLK_ENABLE**(__INDEX__) do { if((__IND
EVAL_COM1_RX_GPIO_CLK_ENABLE();} while(0)

#define **COMx_RX_GPIO_CLK_DISABLE**(__INDEX__) (((__INDEX_
EVAL_COM1_RX_GPIO_CLK_DISABLE() : 0)

#define **IOE_IT_PIN** GPIO_PIN_14
IO Expander Interrupt line on EXTI.

#define **IOE_IT_GPIO_PORT** GPIOB

#define **IOE_IT_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_CLK_I

#define **IOE_IT_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_CLK_

#define **IOE_IT_EXTI_IRQn** EXTI15_10_IRQn

#define **IOE_IT_EXTI_IRQHANDLER** EXTI15_10_IRQHandler

#define **IO1_I2C_ADDRESS** 0x82

#define **IO2_I2C_ADDRESS** 0x88

#define **TS_I2C_ADDRESS** 0x82

#define **L1S302DL_I2C_ADDRESS** 0x38

```
#define READWRITE_CMD ((uint8_t)0x80)
#define MULTIPLEBYTE_CMD ((uint8_t)0x40)
#define EVAL_I2Cx_SCL_PIN GPIO_PIN_6 /* PB.06*/
#define EVAL_I2Cx_SCL_GPIO_PORT GPIOB
#define EVAL_I2Cx_SDA_PIN GPIO_PIN_7 /* PB.07*/
#define EVAL_I2Cx_SDA_GPIO_PORT GPIOB
#define EVAL_I2Cx I2C1
#define EVAL_I2Cx_CLK_ENABLE() __HAL_RCC_I2C1_CLK_ENAI
#define EVAL_I2Cx_SDA_GPIO_CLK_ENABLE() __HAL_RCC_GPI
#define EVAL_I2Cx_SCL_GPIO_CLK_ENABLE() __HAL_RCC_GPI
#define EVAL_I2Cx_FORCE_RESET() __HAL_RCC_I2C1_FORCE_
#define EVAL_I2Cx_RELEASE_RESET() __HAL_RCC_I2C1_RELE/
#define EVAL_I2Cx_EV_IRQn I2C1_EV_IRQn
#define EVAL_I2Cx_EV_IRQHandler I2C1_EV_IRQHandler
#define EVAL_I2Cx_ER_IRQn I2C1_ER_IRQn
#define EVAL_I2Cx_ER_IRQHandler I2C1_ER_IRQHandler
#define EVAL_I2Cx_TIMEOUT_MAX 3000
#define EVAL_SPIx SPI3
#define EVAL_SPIx_CLK_ENABLE() __HAL_RCC_SPI3_CLK_ENA
#define EVAL_SPIx_SCK_GPIO_PORT GPIOC /* PC.10*/
#define EVAL_SPIx_SCK_PIN GPIO_PIN_10
#define EVAL_SPIx_SCK_GPIO_CLK_ENABLE() __HAL_RCC_GPI
#define EVAL_SPIx_SCK_GPIO_CLK_DISABLE() __HAL_RCC_GP
#define EVAL_SPIx_MISO_MOSI_GPIO_PORT GPIOC
#define EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE() __HAL_R
#define EVAL_SPIx_MISO_MOSI_GPIO_CLK_DISABLE() __HAL_F
#define EVAL_SPIx_MISO_PIN GPIO_PIN_11 /* PC.11*/
#define EVAL_SPIx_MOSI_PIN GPIO_PIN_12 /* PC.12*/
#define EVAL_SPIx_TIMEOUT_MAX 1000
#define LCD_CS_LOW() HAL_GPIO_WritePin(LCD_NCS_GPIO_PO
GPIO_PIN_RESET)
#define LCD_CS_HIGH() HAL_GPIO_WritePin(LCD_NCS_GPIO_PC
GPIO_PIN_SET)
#define LCD_NCS_PIN GPIO_PIN_2 /* PB.02*/
```

	LCD Control Interface pins.
#define	**LCD_NCS_GPIO_PORT** GPIOB
#define	**LCD_NCS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_CL
#define	**LCD_NCS_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_C
#define	**SD_CS_LOW**() HAL_GPIO_WritePin(**SD_CS_GPIO_PORT**, GPIO_PIN_RESET)
#define	**SD_CS_HIGH**() HAL_GPIO_WritePin(**SD_CS_GPIO_PORT**, GPIO_PIN_SET)
#define	**SD_CS_PIN** GPIO_PIN_4 /* PA.04*/ SD Control Interface pins.
#define	**SD_CS_GPIO_PORT** GPIOA
#define	**SD_CS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOA_CLK_
#define	**SD_CS_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOA_CLK_
#define	**SD_DETECT_PIN** GPIO_PIN_0 SD Detect Interface pins.
#define	**SD_DETECT_GPIO_PORT** GPIOE
#define	**SD_DETECT_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_
#define	**SD_DETECT_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_
#define	**SD_DETECT_EXTI_IRQn** EXTI0_IRQn
#define	**AUDIO_I2C_ADDRESS** 0x94 AUDIO I2C Interface pins.

Enumerations

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

enum	**Button_TypeDef** { **BUTTON_WAKEUP** = 0, **BUTTON_TAMPER** = 1, **BUTTON_KEY** = 2 } BUTTON Types Definition. More...
enum	**ButtonMode_TypeDef** { **BUTTON_MODE_GPIO** = 0, **BUTTON_MODE_EXTI** = 1 }

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

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

Functions

uint32_t	**BSP_GetVersion** (void) This method returns the STM3210C EVAL BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LED GPIO.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** Button_Mode) Configures push button GPIO and EXTI Line.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) Configures COM port.
uint8_t	**BSP_JOY_Init** (**JOYMode_TypeDef** Joy_Mode) Configures joystick GPIO and EXTI modes.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the current joystick status.

Detailed Description

This file contains definitions for STM3210C_EVAL's LEDs, push-buttons and COM ports hardware resources.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

Definition in file **stm3210c_eval.h**.

STM3210C_EVAL BSP User Manual

stm3210c_eval_accelerometer.c File Reference

This file provides a set of functions needed to manage the ACCELEROMETER MEMS accelerometer available on STM3210C_EVAL board. More...

```
#include "stm3210c_eval_accelerometer.h"
```

Go to the source code of this file.

Functions

uint8_t	**BSP_ACCELERO_Init** (void) Set ACCELEROMETER Initialization.
uint8_t	**BSP_ACCELERO_ReadID** (void) Read ID of Accelerometer component.
void	**BSP_ACCELERO_Reset** (void) Reboot memory content of ACCELEROMETER.
void	**BSP_ACCELERO_Click_ITConfig** (void) Config Accelerometer click IT.
void	**BSP_ACCELERO_Click_ITClear** (void) Clear Accelerometer click IT.
void	**BSP_ACCELERO_GetXYZ** (int16_t *pDataXYZ) Get XYZ acceleration.

Variables

static ACCELERO_DrvTypeDef * **AcceleroDrv**

Detailed Description

This file provides a set of functions needed to manage the ACCELEROMETER MEMS accelerometer available on STM3210C_EVAL board.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

Definition in file **stm3210c_eval_accelerometer.c**.

STM3210C_EVAL BSP User Manual

Enumerations | Functions

stm3210c_eval_accelerometer.h File Reference

This file contains all the functions prototypes for the **stm3210c_eval_accelerometer.c** firmware driver. More...

```
#include "stm3210c_eval.h" #include
"../Components/lis302dl/lis302dl.h"
```

Go to the source code of this file.

Enumerations

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

Functions

uint8_t	**BSP_ACCELERO_Init** (void) Set ACCELEROMETER Initialization.
uint8_t	**BSP_ACCELERO_ReadID** (void) Read ID of Accelerometer component.
void	**BSP_ACCELERO_Reset** (void) Reboot memory content of ACCELEROMETER.
void	**BSP_ACCELERO_Click_ITConfig** (void) Config Accelerometer click IT.
void	**BSP_ACCELERO_Click_ITClear** (void) Clear Accelerometer click IT.
void	**BSP_ACCELERO_GetXYZ** (int16_t *pDataXYZ) Get XYZ acceleration.

Detailed Description

This file contains all the functions prototypes for the **stm3210c_eval_accelerometer.c** firmware driver.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

Definition in file **stm3210c_eval_accelerometer.h**.

STM3210C_EVAL BSP User Manual

stm3210c_eval_audio.c File Reference

This file provides the Audio driver for the STM3210C-Eval board.
More...

```
#include "stm3210c_eval_audio.h"
```

Go to the source code of this file.

Functions

static void	**I2SOUT_MspInit** (void) AUDIO OUT I2S MSP Init.
static void	**I2SOUT_Init** (uint32_t AudioFreq) Initializes the Audio Codec audio interface (I2S)
uint8_t	**BSP_AUDIO_OUT_Init** (uint16_t OutputDevice, uint8_t Volume, uint32_t AudioFreq) Configure the audio peripherals.
uint8_t	**BSP_AUDIO_OUT_Play** (uint16_t *pBuffer, uint32_t Size) Starts playing audio stream from a data buffer for a determined size.
void	**BSP_AUDIO_OUT_ChangeBuffer** (uint16_t *pData, uint16_t Size) Sends n-Bytes on the I2S interface.
uint8_t	**BSP_AUDIO_OUT_Pause** (void) This function Pauses the audio file stream.
uint8_t	**BSP_AUDIO_OUT_Resume** (void) This function Resumes the audio file stream.
uint8_t	**BSP_AUDIO_OUT_Stop** (uint32_t Option) Stops audio playing and Power down the Audio Codec.
uint8_t	**BSP_AUDIO_OUT_SetVolume** (uint8_t Volume) Controls the current audio volume level.
uint8_t	**BSP_AUDIO_OUT_SetMute** (uint32_t Cmd) Enables or disables the MUTE mode by software.
uint8_t	**BSP_AUDIO_OUT_SetOutputMode** (uint8_t Output) Switch dynamically (while audio file is played) the output target (speaker or headphone).
void	**BSP_AUDIO_OUT_SetFrequency** (uint32_t AudioFreq) Update the audio frequency.
void	**HAL_I2S_TxCpltCallback** (I2S_HandleTypeDef *hi2s) Tx Transfer completed callbacks.

void	**HAL_I2S_TxHalfCpltCallback** (I2S_HandleTypeDef *hi2s) Tx Transfer Half completed callbacks.
void	**HAL_I2S_ErrorCallback** (I2S_HandleTypeDef *hi2s) I2S error callbacks.
__weak void	**BSP_AUDIO_OUT_TransferComplete_CallBack** (void) Manages the DMA full Transfer complete event.
__weak void	**BSP_AUDIO_OUT_HalfTransfer_CallBack** (void) Manages the DMA Half Transfer complete event.
__weak void	**BSP_AUDIO_OUT_Error_CallBack** (void) Manages the DMA FIFO error event.

Variables

static AUDIO_DrvTypeDef *	**pAudioDrv**
I2S_HandleTypeDef	**hAudioOutI2s**

Detailed Description

This file provides the Audio driver for the STM3210C-Eval board.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

```
  ==========================================
==================================
                     ##### How to use this dri
ver #####
  ==========================================
==================================
   [..]
  (#) This driver supports STM32F107xC devices
 on STM3210C-Eval Kit:
       (++) to play an audio file (all functio
ns names start by BSP_AUDIO_OUT_xxx)

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

```
 (8000, 16000, 22500, 32000 ...)
              this parameter is relative to th
e audio file/stream type.
               )
           This function configures all the ha
rdware required for the audio application
           (codec, I2C, I2S, GPIOs, DMA and in
terrupt if needed). This function returns 0
           if configuration is OK.
           If the returned value is different
from 0 or the function is stuck then the
           communication with the codec (try t
o un-plug the power or reset device in this ca
se).
              (+++) OUTPUT_DEVICE_SPEAKER: onl
y speaker will be set as output for the
                    audio stream.
              (+++) OUTPUT_DEVICE_HEADPHONE: o
nly headphones will be set as output for
                    the audio stream.
              (+++) OUTPUT_DEVICE_AUTO: Select
ion of output device is made through external
                    switch (implemented into t
he audio jack on the evaluation board).
                    When the Headphone is conn
ected it is used as output.
                    When the headphone is disc
onnected from the audio jack, the output is
                    automatically switched to
Speaker.
              (+++) OUTPUT_DEVICE_BOTH: both S
peaker and Headphone are used as outputs
                    for the audio stream at th
e same time.
       (++) Call the function BSP_AUDIO_OUT_Pl
ay(
               pBuffer: pointer to the audio
```

```
data file address
                Size: size of the buffer to be
 sent in Bytes
               )
              to start playing (for the first
time) from the audio file/stream.
       (++) Call the function BSP_AUDIO_OUT_Pa
use() to pause playing
       (++) Call the function BSP_AUDIO_OUT_Re
sume() to resume playing.
            Note. After calling BSP_AUDIO_OUT_
Pause() function for pause,
            only BSP_AUDIO_OUT_Resume() should
 be called for resume
            (it is not allowed to call BSP_AUD
IO_OUT_Play() in this case).
            Note. This function should be call
ed only when the audio file is played
            or paused (not stopped).
       (++) For each mode, you may need to imp
lement the relative callback functions
            into your code.
            The Callback functions are named B
SP_AUDIO_OUT_XXXCallBack() and only
            their prototypes are declared in t
he stm3210c_eval_audio.h file.
            (refer to the example for more det
ails on the callbacks implementations)
       (++) To Stop playing, to modify the vol
ume level, the frequency or to mute,
            use the functions BSP_AUDIO_OUT_St
op(), BSP_AUDIO_OUT_SetVolume(),
            AUDIO_OUT_SetFrequency() BSP_AUDIO
_OUT_SetOutputMode and BSP_AUDIO_OUT_SetMute()
.
       (++) The driver API and the callback fu
nctions are at the end of the
```

```
            stm3210c_eval_audio.h file.

       (++) This driver provide the High Audio
 Layer: consists of the function API
            exported in the stm3210c_eval_audi
o.h file (BSP_AUDIO_OUT_Init(),
            BSP_AUDIO_OUT_Play() ...)
       (++) This driver provide also the Media
 Access Layer (MAL): which consists
            of functions allowing to access th
e media containing/providing the
            audio file/stream. These functions
 are also included as local functions into
            the stm3210c_eval_audio.c file (I2
SOUT_Init()...)

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

   (#) Communication with the audio codec (thr
ough I2C) may be corrupted if it
       is interrupted by some user interrupt r
outines (in this case, interrupts
       could be disabled just before the start
```

```
 of communication then re-enabled
       when it is over). Note that this commun
ication is only done at the
       configuration phase (BSP_AUDIO_OUT_Init
() or BSP_AUDIO_OUT_Stop())
       and when Volume control modification is
 performed (BSP_AUDIO_OUT_SetVolume()
       or BSP_AUDIO_OUT_SetMute()or BSP_AUDIO_
OUT_SetOutputMode()).
       When the audio data is played, no commu
nication is required with the audio codec.

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

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

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

Attention:

Definition in file **stm3210c_eval_audio.c**.

STM3210C_EVAL BSP User Manual

stm3210c_eval_audio.h File Reference

This file contains the common defines and functions prototypes for **stm3210c_eval_audio.c** driver. More...

```
#include "../Components/cs43l22/cs43l22.h" #include
"stm3210c_eval.h"
```

Go to the source code of this file.

Defines

```
#define I2SOUT SPI2
#define I2SOUT_CLK_ENABLE() __HAL_RCC_SPI2_CLK_ENABLE
#define I2SOUT_SCK_SD_CLK_ENABLE() __HAL_RCC_GPIOB_C
#define I2SOUT_MCK_CLK_ENABLE() __HAL_RCC_GPIOC_CLK_
#define I2SOUT_WS_CLK_ENABLE() __HAL_RCC_GPIOB_CLK_E
#define I2SOUT_WS_PIN GPIO_PIN_12 /* PB.12*/
#define I2SOUT_SCK_PIN GPIO_PIN_13 /* PB.13*/
#define I2SOUT_SD_PIN GPIO_PIN_15 /* PB.15*/
#define I2SOUT_MCK_PIN GPIO_PIN_6 /* PC.06*/
#define I2SOUT_SCK_SD_GPIO_PORT GPIOB
#define I2SOUT_WS_GPIO_PORT GPIOB
#define I2SOUT_MCK_GPIO_PORT GPIOC
#define I2SOUT_DMAx_CLK_ENABLE() __HAL_RCC_DMA1_CLK_
#define I2SOUT_DMAx_CHANNEL DMA1_Channel5
#define I2SOUT_DMAx_IRQ DMA1_Channel5_IRQn
#define I2SOUT_DMAx_PERIPH_DATA_SIZE DMA_PDATAALIGN_
#define I2SOUT_DMAx_MEM_DATA_SIZE DMA_MDATAALIGN_HA
#define DMA_MAX_SZE 0xFFFF
#define I2SOUT_IRQHandler DMA1_Channel5_IRQHandler
#define AUDIO_OUT_IRQ_PREPRIO 5 /* Select the preemption prio
#define AUDIO_OK 0
#define AUDIO_ERROR 1
#define AUDIO_TIMEOUT 2
#define INTERNAL_BUFF_SIZE 128*DEFAULT_AUDIO_IN_FREQ/1
#define DMA_MAX(_X_) (((_X_) <= DMA_MAX_SZE)? (_X_):DMA_
```

Functions

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

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

Detailed Description

This file contains the common defines and functions prototypes for **stm3210c_eval_audio.c** driver.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

Definition in file **stm3210c_eval_audio.h**.

STM3210C_EVAL BSP User Manual

stm3210c_eval_eeprom.c File Reference

This file provides a set of functions needed to manage a M24C64 I2C EEPROM memory. More...

```
#include "stm3210c_eval_eeprom.h"
```

Go to the source code of this file.

Functions

static uint32_t	**EEPROM_I2C_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
static uint32_t	**EEPROM_I2C_ReadBuffer** (uint8_t *pBuffer, uint16_t ReadAddr, uint32_t *NumByteToRead) Reads a block of data from the I2C EEPROM.
static uint32_t	**EEPROM_I2C_WritePage** (uint8_t *pBuffer, uint16_t WriteAddr, uint32_t *NumByteToWrite) Writes more than one byte to the EEPROM with a single WRITE cycle.
static uint32_t	**EEPROM_I2C_WaitEepromStandbyState** (void) Wait for EEPROM I2C Standby state.
uint32_t	**BSP_EEPROM_Init** (void) Initializes peripherals used by the EEPROM device selected.
void	**BSP_EEPROM_SelectDevice** (uint8_t DeviceID) Select the EEPROM device to communicate.
uint32_t	**BSP_EEPROM_ReadBuffer** (uint8_t *pBuffer, uint16_t ReadAddr, uint32_t *NumByteToRead) Reads a block of data from the EEPROM device selected.
uint32_t	**BSP_EEPROM_WriteBuffer** (uint8_t *pBuffer, uint16_t WriteAddr, uint32_t NumByteToWrite) Writes buffer of data to the EEPROM device selected.
__weak void	**BSP_EEPROM_TIMEOUT_UserCallback** (void) Basic management of the timeout situation.

Variables

__IO uint16_t	**EEPROMAddress** = 0
__IO uint16_t	**EEPROMPageSize** = 0
__IO uint16_t	**EEPROMDataRead** = 0
__IO uint8_t	**EEPROMDataWrite** = 0
static **EEPROM_DrvTypeDef** *	**EEPROM_SelectedDevice** = 0
EEPROM_DrvTypeDef	**EEPROM_I2C_Drv**

Detailed Description

This file provides a set of functions needed to manage a M24C64 I2C EEPROM memory.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015
==========
Notes:

- This driver is intended for STM32F1xx families devices only.
- The I2C EEPROM memory (M24CXX) is available directly on STM3210C EVAL board.
  ==========

It implements a high level communication layer for read and write from/to this memory. The needed STM32F10x hardware resources (I2C and GPIO) are defined in **stm3210c_eval.h** file, and the initialization is performed in **EEPROM_I2C_IO_Init()** functions declared in **stm3210c_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **EEPROM_I2C_IO_Init()** functions.

> **Note:**
> In this driver, basic read and write functions (**BSP_EEPROM_ReadBuffer()** and **BSP_EEPROM_WriteBuffer()**) use Polling mode to perform the data transfer to/from EEPROM memories. +---------- ---------- ----------+ | Pin assignment for M24CXX EEPROM | +----------+----------+----------+ | STM32F1xx I2C Pins | EEPROM | Pin | +---------- -------+----------+----------+ | EEPROM_I2C_SDA_PIN/ SDA |

SDA | 5 | | EEPROM_I2C_SCL_PIN/ SCL | SCL | 6 | | . | VDD | 7 (3.3V) | | . | GND | 8 (0 V) | +----------+----------+----------+----------+----------+----------+----------+----------+

Attention:

Definition in file **stm3210c_eval_eeprom.c**.

STM3210C_EVAL BSP User Manual

stm3210c_eval_eeprom.h File Reference

This file contains all the functions prototypes for the **stm3210c_eval_eeprom.c** firmware driver. More...

```
#include "stm3210c_eval.h"
```

Go to the source code of this file.

Data Structures

struct	**EEPROM_DrvTypeDef**

Defines

#define	**EEPROM_ADDRESS_M24C64_32** 0xA0 /* Support the devices: M24C32 and M24C64 */
#define	**EEPROM_ADDRESS_M24C08_BLOCK0** 0xA0
#define	**EEPROM_ADDRESS_M24C08_BLOCK1** 0xA2
#define	**EEPROM_ADDRESS_M24C08_BLOCK2** 0xA4
#define	**EEPROM_ADDRESS_M24C08_BLOCK3** 0xA6
#define	**EEPROM_PAGESIZE_M24C64_32** 32 /* Support the devices: M24C32 and M24C64 */
#define	**EEPROM_PAGESIZE_M24C08** 16 /* Support the device: M24C08. */
#define	**EEPROM_OK** 0
#define	**EEPROM_FAIL** 1
#define	**EEPROM_TIMEOUT** 2
#define	**BSP_EEPROM_M24C64_32** 1 /* RF I2C EEPROM M24C32 and M24C64 */
#define	**BSP_EEPROM_M24C08** 2 /* RF I2C EEPROM M24C08 */
#define	**EEPROM_MAX_TRIALS** 300

Functions

uint32_t	**BSP_EEPROM_Init** (void) Initializes peripherals used by the EEPROM device selected.
void	**BSP_EEPROM_SelectDevice** (uint8_t DeviceID) Select the EEPROM device to communicate.
uint32_t	**BSP_EEPROM_ReadBuffer** (uint8_t *pBuffer, uint16_t ReadAddr, uint32_t *NumByteToRead) Reads a block of data from the EEPROM device selected.
uint32_t	**BSP_EEPROM_WriteBuffer** (uint8_t *pBuffer, uint16_t WriteAddr, uint32_t NumByteToWrite) Writes buffer of data to the EEPROM device selected.
__weak void	**BSP_EEPROM_TIMEOUT_UserCallback** (void) Basic management of the timeout situation.
void	**EEPROM_I2C_IO_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_I2C_IO_WriteData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Write data to I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_I2C_IO_ReadData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Read data from I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_I2C_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 **stm3210c_eval_eeprom.c** firmware driver.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

Definition in file **stm3210c_eval_eeprom.h**.

STM3210C_EVAL BSP User Manual

stm3210c_eval_io.c File Reference

This file provides a set of functions needed to manage the IO pins on STM3210C-EVAL evaluation board. More...

```
#include "stm3210c_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..).
uint32_t	**BSP_IO_ITGetStatus** (uint32_t IO_Pin) Gets the selected pins IT status.
void	**BSP_IO_ITClear** (uint32_t IO_Pin) Clears the selected IO IT pending bit.
uint8_t	**BSP_IO_ConfigPin** (uint32_t IO_Pin, IO_ModeTypedef IO_Mode) Configures the IO pin(s) according to IO mode structure value.
void	**BSP_IO_WritePin** (uint32_t IO_Pin, uint8_t PinState) Sets the selected pins state.
uint32_t	**BSP_IO_ReadPin** (uint32_t IO_Pin) Gets the selected pins current state.
void	**BSP_IO_TogglePin** (uint32_t IO_Pin) Toggles the selected pins state.

Variables

static IO_DrvTypeDef *	**io1_driver**
static IO_DrvTypeDef *	**io2_driver**

Detailed Description

This file provides a set of functions needed to manage the IO pins on STM3210C-EVAL evaluation board.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

Definition in file **stm3210c_eval_io.c**.

STM3210C_EVAL BSP User Manual

stm3210c_eval_io.h File Reference

This file contains the common defines and functions prototypes for the **stm3210c_eval_io.c** driver. More...

```
#include "stm3210c_eval.h" #include
"../Components/stmpe811/stmpe811.h"
```

Go to the source code of this file.

Defines

#define	**IO1_PIN_OFFSET** 0
#define	**IO2_PIN_OFFSET** 8
#define	**IO1_PIN_0** (uint32_t)(0x00000001 << IO1_PIN_OFFSET)
#define	**IO1_PIN_1** (uint32_t)(0x00000002 << IO1_PIN_OFFSET)
#define	**IO1_PIN_2** (uint32_t)(0x00000004 << IO1_PIN_OFFSET)
#define	**IO1_PIN_3** (uint32_t)(0x00000008 << IO1_PIN_OFFSET)
#define	**IO1_PIN_4** (uint32_t)(0x00000010 << IO1_PIN_OFFSET)
#define	**IO1_PIN_5** (uint32_t)(0x00000020 << IO1_PIN_OFFSET)
#define	**IO1_PIN_6** (uint32_t)(0x00000040 << IO1_PIN_OFFSET)
#define	**IO1_PIN_7** (uint32_t)(0x00000080 << IO1_PIN_OFFSET)
#define	**IO1_PIN_ALL** (uint32_t)(0x000000FF << IO1_PIN_OFFSET)
#define	**IO2_PIN_0** (uint32_t)(0x00000001 << IO2_PIN_OFFSET)
#define	**IO2_PIN_1** (uint32_t)(0x00000002 << IO2_PIN_OFFSET)
#define	**IO2_PIN_2** (uint32_t)(0x00000004 << IO2_PIN_OFFSET)
#define	**IO2_PIN_3** (uint32_t)(0x00000008 << IO2_PIN_OFFSET)
#define	**IO2_PIN_4** (uint32_t)(0x00000010 << IO2_PIN_OFFSET)
#define	**IO2_PIN_5** (uint32_t)(0x00000020 << IO2_PIN_OFFSET)
#define	**IO2_PIN_6** (uint32_t)(0x00000040 << IO2_PIN_OFFSET)
#define	**IO2_PIN_7** (uint32_t)(0x00000080 << IO2_PIN_OFFSET)
#define	**IO2_PIN_ALL** (uint32_t)(0x000000FF << IO2_PIN_OFFSET)

Enumerations

enum	**IO_StatusTypeDef** { **IO_OK** = 0x00, **IO_ERROR** = 0x01, **IO_TIMEOUT** = 0x02 }

Functions

uint8_t	**BSP_IO_Init** (void) Initializes and configures the IO functionalities and configures all necessary hardware resources (GPIOs, clocks..).
void	**BSP_IO_ITClear** (uint32_t IO_Pin) Clears the selected IO IT pending bit.
uint32_t	**BSP_IO_ITGetStatus** (uint32_t IO_Pin) Gets the selected pins IT status.
uint8_t	**BSP_IO_ConfigPin** (uint32_t IO_Pin, IO_ModeTypedef IO_Mode) Configures the IO pin(s) according to IO mode structure value.
void	**BSP_IO_WritePin** (uint32_t IO_Pin, uint8_t PinState) Sets the selected pins state.
uint32_t	**BSP_IO_ReadPin** (uint32_t IO_Pin) Gets the selected pins current state.
void	**BSP_IO_TogglePin** (uint32_t IO_Pin) Toggles the selected pins state.

Detailed Description

This file contains the common defines and functions prototypes for the **stm3210c_eval_io.c** driver.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

Definition in file **stm3210c_eval_io.h**.

STM3210C_EVAL BSP User Manual

stm3210c_eval_lcd.c File Reference

This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM3210C-EVAL evaluation board. More...

```
#include "stm3210c_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 MAX_HEIGHT_FONT   17
#define MAX_WIDTH_FONT   24
#define OFFSET_BITMAP   54
#define ABS(X)   ((X) > 0 ? (X) : -(X))
```

Functions

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

void	**BSP_LCD_DisplayChar** (uint16_t Xpos, uint16_t Ypos, uint8_t Ascii) Displays one character.
void	**BSP_LCD_DisplayStringAt** (uint16_t Xpos, uint16_t Ypos, uint8_t *pText, **Line_ModeTypdef** Mode) Displays characters on the LCD.
void	**BSP_LCD_DisplayStringAtLine** (uint16_t Line, uint8_t *pText) Displays a character on the LCD.
uint16_t	**BSP_LCD_ReadPixel** (uint16_t Xpos, uint16_t Ypos) Reads an LCD pixel.
void	**BSP_LCD_DrawHLine** (uint16_t Xpos, uint16_t Ypos, uint16_t Length) Draws an horizontal line.
void	**BSP_LCD_DrawVLine** (uint16_t Xpos, uint16_t Ypos, uint16_t Length) Draws a vertical line.
void	**BSP_LCD_DrawLine** (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2) Draws an uni-line (between two points).
void	**BSP_LCD_DrawRect** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Draws a rectangle.
void	**BSP_LCD_DrawCircle** (uint16_t Xpos, uint16_t Ypos, uint16_t Radius) Draws a circle.
void	**BSP_LCD_DrawPolygon** (**pPoint** Points, uint16_t PointCount) Draws an poly-line (between many points).
void	**BSP_LCD_DrawEllipse** (int Xpos, int Ypos, int XRadius, int YRadius) Draws an ellipse on LCD.
void	**BSP_LCD_DrawBitmap** (uint16_t Xpos, uint16_t Ypos, uint8_t *pbmp)

	Draws a bitmap picture (16 bpp).
void	**BSP_LCD_DrawRGBImage** (uint16_t Xpos, uint16_t Ypos, uint16_t Xsize, uint16_t Ysize, uint8_t *pdata) Draws RGB Image (16 bpp).
void	**BSP_LCD_FillRect** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Draws a full rectangle.
void	**BSP_LCD_FillCircle** (uint16_t Xpos, uint16_t Ypos, uint16_t Radius) Draws a full circle.
void	**BSP_LCD_FillEllipse** (int Xpos, int Ypos, int XRadius, int YRadius) Draws a full ellipse.
void	**BSP_LCD_DisplayOn** (void) Enables the display.
void	**BSP_LCD_DisplayOff** (void) Disables the display.

Variables

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

Detailed Description

This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM3210C-EVAL evaluation board.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

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

   (#) This driver supports the AM-240320L8TNQ
W00H (LCD_ILI9320) and AM240320D5TOQW01H
       (LCD_ILI9325) LCD mounted on MB785 daug
hter board

   (#) The ILI9320 and ILI9325 components driv
er MUST be included with this driver.

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

   (#) Display on LCD
```

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

Attention:

Definition in file **stm3210c_eval_lcd.c**.

STM3210C_EVAL BSP User Manual

Data Structures | Defines | Typedefs | Enumerations | Functions

stm3210c_eval_lcd.h File Reference

This file contains the common defines and functions prototypes for the **stm3210c_eval_lcd.c** driver. More...

```
#include "stm3210c_eval.h" #include
"../Components/ili9325/ili9325.h"
#include "../Components/ili9320/ili9320.h"
#include "../../../Utilities/Fonts/fonts.h"
```

Go to the source code of this file.

Data Structures

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

Defines

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

LCD default font.

Typedefs

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

Enumerations

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

Functions

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

void	**BSP_LCD_DrawHLine** (uint16_t Xpos, uint16_t Ypos, uint16_t Length) Draws an horizontal line.
void	**BSP_LCD_DrawVLine** (uint16_t Xpos, uint16_t Ypos, uint16_t Length) Draws a vertical line.
void	**BSP_LCD_DrawLine** (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2) Draws an uni-line (between two points).
void	**BSP_LCD_DrawRect** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Draws a rectangle.
void	**BSP_LCD_DrawCircle** (uint16_t Xpos, uint16_t Ypos, uint16_t Radius) Draws a circle.
void	**BSP_LCD_DrawPolygon** (**pPoint** Points, uint16_t PointCount) Draws an poly-line (between many points).
void	**BSP_LCD_DrawEllipse** (int Xpos, int Ypos, int XRadius, int YRadius) Draws an ellipse on LCD.
void	**BSP_LCD_DrawBitmap** (uint16_t Xpos, uint16_t Ypos, uint8_t *pbmp) Draws a bitmap picture (16 bpp).
void	**BSP_LCD_DrawRGBImage** (uint16_t Xpos, uint16_t Ypos, uint16_t Xsize, uint16_t Ysize, uint8_t *pdata) Draws RGB Image (16 bpp).
void	**BSP_LCD_FillRect** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Draws a full rectangle.
void	**BSP_LCD_FillCircle** (uint16_t Xpos, uint16_t Ypos, uint16_t Radius) Draws a full circle.
	BSP_LCD_FillEllipse (int Xpos, int Ypos, int XRadius, int

void	YRadius) Draws a full ellipse.
void	**BSP_LCD_DisplayOff** (void) Disables the display.
void	**BSP_LCD_DisplayOn** (void) Enables the display.

Detailed Description

This file contains the common defines and functions prototypes for the **stm3210c_eval_lcd.c** driver.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

Definition in file **stm3210c_eval_lcd.h**.

STM3210C_EVAL BSP User Manual

stm3210c_eval_sd.c File Reference

This file provides a set of functions needed to manage the SPI SD Card memory mounted on STM3210C-EVAL board. It implements a high level communication layer for read and write from/to this memory. The needed STM32F10x hardware resources (SPI and GPIO) are defined in **stm3210c_eval.h** file, and the initialization is performed in **SD_IO_Init()** function declared in **stm3210c_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **SD_IO_Init()** function. More...

```
#include "stm3210c_eval_sd.h"
```

Go to the source code of this file.

Defines

#define	**SD_DUMMY_BYTE** 0xFF
#define	**SD_NO_RESPONSE_EXPECTED** 0x80

Functions

static uint8_t	**SD_GetCIDRegister** (**SD_CID** *Cid) Read the CID card register.
static uint8_t	**SD_GetCSDRegister** (**SD_CSD** *Csd) Read the CSD card register.
static **SD_Info**	**SD_GetDataResponse** (void) Get SD card data response.
static uint8_t	**SD_GoIdleState** (void) Put SD in Idle state.
static uint8_t	**SD_SendCmd** (uint8_t Cmd, uint32_t Arg, uint8_t Crc, uint8_t Response) Send 5 bytes command to the SD card and get response.
uint8_t	**BSP_SD_Init** (void) Initializes the SD/SD communication.
uint8_t	**BSP_SD_IsDetected** (void) Detects if SD card is correctly plugged in the memory slot or not.
uint8_t	**BSP_SD_GetCardInfo** (**SD_CardInfo** *pCardInfo) Returns information about specific card.
uint8_t	**BSP_SD_ReadBlocks** (uint32_t *p32Data, uint64_t ReadAddr, uint16_t BlockSize, uint32_t NumberOfBlocks) Reads block(s) from a specified address in an SD card, in polling mode.
uint8_t	**BSP_SD_WriteBlocks** (uint32_t *p32Data, uint64_t WriteAddr, uint16_t BlockSize, uint32_t NumberOfBlocks) Writes block(s) to a specified address in an SD card, in polling mode.
uint8_t	**BSP_SD_GetStatus** (void) Returns the SD status.

uint8_t	**BSP_SD_Erase** (uint32_t StartAddr, uint32_t EndAddr) Erases the specified memory area of the given SD card.

Variables

__IO uint8_t **SdStatus** = **SD_PRESENT**

Detailed Description

This file provides a set of functions needed to manage the SPI SD Card memory mounted on STM3210C-EVAL board. It implements a high level communication layer for read and write from/to this memory. The needed STM32F10x hardware resources (SPI and GPIO) are defined in **stm3210c_eval.h** file, and the initialization is performed in **SD_IO_Init()** function declared in **stm3210c_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **SD_IO_Init()** function.

Author:

MCD Application Team

Version:

V6.0.1

Date:

18-December-2015 +---+ | Pin assignment | +-----------------------+--------------+-------------+ | STM32F10x SPI Pins | SD | Pin | +-----------------------+--------------+-------------+ | SD_SPI_CS_PIN | ChipSelect | 2 | | SD_SPI_MOSI_PIN / MOSI | DataIn | 3 | | | GND | 9 (0 V) | | | VDD | 4 (3.3 V)| | SD_SPI_SCK_PIN / SCLK | Clock | 5 | | | GND | 6 (0 V) | | | SD_SPI_MISO_PIN / MISO | DataOut | 7 | +-----------------------+--------------+-------------+

Attention:

Definition in file **stm3210c_eval_sd.c**.

STM3210C_EVAL BSP User Manual

Data Structures | Defines | Enumerations | Functions

stm3210c_eval_sd.h File Reference

This file contains the common defines and functions prototypes for the **stm3210c_eval_sd.c** driver. More...

```
#include "stm3210c_eval.h"
```

Go to the source code of this file.

Data Structures

struct	**SD_CSD** Card Specific Data: CSD Register. More...
struct	**SD_CID** Card Identification Data: CID Register. More...
struct	**SD_CardInfo** SD Card information. More...

Defines

#define	**MSD_OK** 0x00 SD status structure definition.
#define	**MSD_ERROR** 0x01
#define	**SD_BLOCK_SIZE** 0x200 Block Size.
#define	**SD_START_DATA_SINGLE_BLOCK_READ** 0xFE /* Data token start byte, Start Single Block Read */ Start Data tokens: Tokens (necessary because at nop/idle (and CS active) only 0xff is on the data/command line)
#define	**SD_START_DATA_MULTIPLE_BLOCK_READ** 0xFE /* Data token start byte, Start Multiple Block Read */
#define	**SD_START_DATA_SINGLE_BLOCK_WRITE** 0xFE /* Data token start byte, Start Single Block Write */
#define	**SD_START_DATA_MULTIPLE_BLOCK_WRITE** 0xFD /* Data token start byte, Start Multiple Block Write */
#define	**SD_STOP_DATA_MULTIPLE_BLOCK_WRITE** 0xFD /* Data toke stop byte, Stop Multiple Block Write */
#define	**SD_PRESENT** ((uint8_t)0x01) SD detection on its memory slot.
#define	**SD_NOT_PRESENT** ((uint8_t)0x00)
#define	**SD_CMD_GO_IDLE_STATE** 0 /* CMD0 = 0x40 */ Commands: CMDxx = CMD-number | 0x40.
#define	**SD_CMD_SEND_OP_COND** 1 /* CMD1 = 0x41 */
#define	**SD_CMD_SEND_CSD** 9 /* CMD9 = 0x49 */
#define	**SD_CMD_SEND_CID** 10 /* CMD10 = 0x4A */
#define	**SD_CMD_STOP_TRANSMISSION** 12 /* CMD12 = 0x4C */
#define	**SD_CMD_SEND_STATUS** 13 /* CMD13 = 0x4D */
#define	**SD_CMD_SET_BLOCKLEN** 16 /* CMD16 = 0x50 */
#define	**SD_CMD_READ_SINGLE_BLOCK** 17 /* CMD17 = 0x51 */
#define	**SD_CMD_READ_MULT_BLOCK** 18 /* CMD18 = 0x52 */

```
#define SD_CMD_SET_BLOCK_COUNT  23 /* CMD23 = 0x57 */
#define SD_CMD_WRITE_SINGLE_BLOCK  24 /* CMD24 = 0x58 */
#define SD_CMD_WRITE_MULT_BLOCK  25 /* CMD25 = 0x59 */
#define SD_CMD_PROG_CSD  27 /* CMD27 = 0x5B */
#define SD_CMD_SET_WRITE_PROT  28 /* CMD28 = 0x5C */
#define SD_CMD_CLR_WRITE_PROT  29 /* CMD29 = 0x5D */
#define SD_CMD_SEND_WRITE_PROT  30 /* CMD30 = 0x5E */
#define SD_CMD_SD_ERASE_GRP_START  32 /* CMD32 = 0x60 */
#define SD_CMD_SD_ERASE_GRP_END  33 /* CMD33 = 0x61 */
#define SD_CMD_UNTAG_SECTOR  34 /* CMD34 = 0x62 */
#define SD_CMD_ERASE_GRP_START  35 /* CMD35 = 0x63 */
#define SD_CMD_ERASE_GRP_END  36 /* CMD36 = 0x64 */
#define SD_CMD_UNTAG_ERASE_GROUP  37 /* CMD37 = 0x65 */
#define SD_CMD_ERASE  38 /* CMD38 = 0x66 */
```

Enumerations

enum	**SD_Info** {
	SD_RESPONSE_NO_ERROR = (0x00),
	SD_IN_IDLE_STATE = (0x01), **SD_ERASE_RESET** = (0x02),
	SD_ILLEGAL_COMMAND = (0x04),
	SD_COM_CRC_ERROR = (0x08),
	SD_ERASE_SEQUENCE_ERROR = (0x10),
	SD_ADDRESS_ERROR = (0x20),
	SD_PARAMETER_ERROR = (0x40),
	SD_RESPONSE_FAILURE = (0xFF), **SD_DATA_OK** = (0x05), **SD_DATA_CRC_ERROR** = (0x0B),
	SD_DATA_WRITE_ERROR = (0x0D),
	SD_DATA_OTHER_ERROR = (0xFF)
	}

Functions

uint8_t	**BSP_SD_Init** (void) Initializes the SD/SD communication.
uint8_t	**BSP_SD_IsDetected** (void) Detects if SD card is correctly plugged in the memory slot or not.
uint8_t	**BSP_SD_ReadBlocks** (uint32_t *p32Data, uint64_t ReadAddr, uint16_t BlockSize, uint32_t NumberOfBlocks) Reads block(s) from a specified address in an SD card, in polling mode.
uint8_t	**BSP_SD_WriteBlocks** (uint32_t *p32Data, uint64_t WriteAddr, uint16_t BlockSize, uint32_t NumberOfBlocks) Writes block(s) to a specified address in an SD card, in polling mode.
uint8_t	**BSP_SD_Erase** (uint32_t StartAddr, uint32_t EndAddr) Erases the specified memory area of the given SD card.
uint8_t	**BSP_SD_GetStatus** (void) Returns the SD status.
uint8_t	**BSP_SD_GetCardInfo** (**SD_CardInfo** *pCardInfo) Returns information about specific card.
void	**SD_IO_Init** (void) Initializes the SD Card and put it into StandBy State (Ready for data transfer).
void	**SD_IO_WriteByte** (uint8_t Data) Write a byte on the SD.
uint8_t	**SD_IO_ReadByte** (void) Read a byte from the SD.

HAL_StatusTypeDef	**SD_IO_WriteCmd** (uint8_t Cmd, uint32_t Arg, uint8_t Crc, uint8_t Response) Send 5 bytes command to the SD card and get response.
HAL_StatusTypeDef	**SD_IO_WaitResponse** (uint8_t Response) Wait response from the SD card.
void	**SD_IO_WriteDummy** (void) Send dummy byte with CS High.

Detailed Description

This file contains the common defines and functions prototypes for the **stm3210c_eval_sd.c** driver.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

Definition in file **stm3210c_eval_sd.h**.

STM3210C_EVAL BSP User Manual

stm3210c_eval_ts.c File Reference

This file provides a set of functions needed to manage the touch screen on STM3210C_EVAL evaluation board. More...

```
#include "stm3210c_eval_ts.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_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**

Detailed Description

This file provides a set of functions needed to manage the touch screen on STM3210C_EVAL evaluation board.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

Definition in file **stm3210c_eval_ts.c**.

STM3210C_EVAL BSP User Manual

Data Structures | Defines | Enumerations | Functions

stm3210c_eval_ts.h File Reference

This file contains the common defines and functions prototypes for the **stm3210c_eval_ts.c** driver. More...

```
#include "stm3210c_eval.h" #include
"../Components/stmpe811/stmpe811.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

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_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 **stm3210c_eval_ts.c** driver.

Author:
MCD Application Team

Version:
V6.0.1

Date:
18-December-2015

Attention:

Definition in file **stm3210c_eval_ts.h**.

STM3210C_EVAL BSP User Manual

Modules

Here is a list of all modules:

- STM3210C_EVAL_AUDIO
  - AUDIO_Private_Types
  - AUDIO_Private_Defines
  - AUDIO Private_Macros
  - AUDIO_Private_Variables
  - AUDIO_Private_Function_Prototypes
  - AUDIO_OUT_Exported_Functions
  - AUDIO_Exported_Types
  - AUDIO_OUT_Exported_Constants
  - AUDIO_Exported_Macros
- STM3210C_EVAL_EEPROM
  - Private Types
  - Private Defines
  - Private Macros
  - Private Variables
  - Private Function Prototypes
  - Exported Functions
  - Exported Types
  - Exported Constants
  - Exported Macros

STM3210C_EVAL BSP User Manual

Data Structures

Here are the data structures with brief descriptions:

EEPROM_DrvTypeDef	
LCD_DrawPropTypeDef	
Point	
SD_CardInfo	SD Card information
SD_CID	Card Identification Data: CID Register
SD_CSD	Card Specific Data: CSD Register
TFT_LCD_TypeDef	
TS_StateTypeDef	

STM3210C_EVAL BSP User Manual

File List

Here is a list of all files with brief descriptions:

stm3210c_eval.c [code]	This file provides a set of fi functions to manage Leds, button and COM ports for STM3210C_EVAL
stm3210c_eval.h [code]	This file contains definitions STM3210C_EVAL's LEDs, buttons and COM ports har resources
stm3210c_eval_accelerometer.c [code]	This file provides a set of functions needed to manag ACCELEROMETER MEMS accelerometer available on STM3210C_EVAL board
stm3210c_eval_accelerometer.h [code]	This file contains all the fun prototypes for the **stm3210c_eval_acceleror** firmware driver
stm3210c_eval_audio.c [code]	This file provides the Audio for the STM3210C-Eval boa
stm3210c_eval_audio.h [code]	This file contains the comm defines and functions proto for **stm3210c_eval_audio.**

	driver
stm3210c_eval_eeprom.c [code]	This file provides a set of functions needed to manag M24C64 I2C EEPROM me
stm3210c_eval_eeprom.h [code]	This file contains all the fun prototypes for the **stm3210c_eval_eeprom.c** firmware driver
stm3210c_eval_io.c [code]	This file provides a set of functions needed to manag IO pins on STM3210C-EVA evaluation board
stm3210c_eval_io.h [code]	This file contains the comm defines and functions proto for the **stm3210c_eval_io.** driver
stm3210c_eval_lcd.c [code]	This file includes the driver Liquid Crystal Display (LCD module mounted on STM32 EVAL evaluation board
stm3210c_eval_lcd.h [code]	This file contains the comm defines and functions proto for the **stm3210c_eval_lcd** driver
stm3210c_eval_sd.c [code]	This file provides a set of functions needed to manag SPI SD Card memory mou STM3210C-EVAL board. It implements a high level communication layer for rea write from/to this memory. T needed STM32F10x hardw resources (SPI and GPIO) defined in **stm3210c_eval.** and the initialization is perfo

	in **SD_IO_Init()** function de in **stm3210c_eval.c** file. Yo easily tailor this driver to an development board, by just adapting the defines for har resources and **SD_IO_Init(** function
stm3210c_eval_sd.h [code]	This file contains the comm defines and functions proto for the **stm3210c_eval_sd.** driver
stm3210c_eval_ts.c [code]	This file provides a set of functions needed to manag touch screen on STM3210C_EVAL evaluatio board
stm3210c_eval_ts.h [code]	This file contains the comm defines and functions proto for the **stm3210c_eval_ts.** driver

STM3210C_EVAL BSP User Manual

Directories

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

- Drivers
  - BSP
    - STM3210C_EVAL

Generated on Thu Dec 10 2015 17:13:53 for STM3210C_EVAL BSP User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Data Structure Index

E | L | P | S | T

E | L | P | S | T

STM3210C_EVAL BSP User Manual

Private Types Definitions

STM3210C-EVAL Common

Data Structures

struct	**TFT_LCD_TypeDef**

STM3210C_EVAL BSP User Manual

stm3210c_eval.c

Go to the documentation of this file.

```
00001 /**
00002   ********************************************************************************
00003   * @file    stm3210c_eval.c
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file provides a set of firmware functions to manage Leds,
00008   *          push-button and COM ports for STM3210C_EVAL
00009   ********************************************************************************
00010   * @attention
00011   *
00012   * <h2><center>&copy; COPYRIGHT(c) 2015 STMicroelectronics</center></h2>
00013   *
00014   * Redistribution and use in source and binary forms, with or without modification,
00015   * are permitted provided that the following conditions are met:
```

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

```
00035   *
00036   ******************************************************************************
00037   */
00038
00039 /* Includes ------------------------------------------------------------------*/
00040 #include "stm3210c_eval.h"
00041
00042 /** @addtogroup BSP
00043   * @{
00044   */
00045
00046 /** @defgroup STM3210C_EVAL STM3210C-EVAL
00047   * @{
00048   */
00049
00050 /** @defgroup STM3210C_EVAL_COMMON STM3210C-EVAL Common
00051   * @{
00052   */
00053
00054 /** @defgroup STM3210C_EVAL_Private_TypesDefinitions Private Types Definitions
00055   * @{
00056   */
00057
00058 typedef struct
00059 {
00060   __IO uint16_t LCD_REG_R; /* Read Register */
00061   __IO uint16_t LCD_RAM_R; /* Read RAM */
00062   __IO uint16_t LCD_REG_W; /* Write Register */
00063   __IO uint16_t LCD_RAM_W; /* Write RAM */
00064 } TFT_LCD_TypeDef;
00065
```

```
00066 /**
00067   * @}
00068   */
00069
00070 /** @defgroup STM3210C_EVAL_Private_Defines
Private Defines
00071   * @{
00072   */
00073
00074 /* LINK LCD */
00075 #define START_BYTE          0x70
00076 #define SET_INDEX           0x00
00077 #define READ_STATUS         0x01
00078 #define LCD_WRITE_REG       0x02
00079 #define LCD_READ_REG        0x03
00080
00081 /* LINK SD Card */
00082 #define SD_DUMMY_BYTE       0xFF
00083 #define SD_NO_RESPONSE_EXPECTED  0x80
00084
00085 /**
00086  * @brief STM3210C EVAL BSP Driver version n
umber V6.0.1
00087   */
00088 #define __STM3210C_EVAL_BSP_VERSION_MAIN
   (0x06) /*!< [31:24] main version */
00089 #define __STM3210C_EVAL_BSP_VERSION_SUB1
   (0x00) /*!< [23:16] sub1 version */
00090 #define __STM3210C_EVAL_BSP_VERSION_SUB2
   (0x01) /*!< [15:8]  sub2 version */
00091 #define __STM3210C_EVAL_BSP_VERSION_RC
   (0x00) /*!< [7:0]  release candidate */
00092 #define __STM3210C_EVAL_BSP_VERSION
   ((__STM3210C_EVAL_BSP_VERSION_MAIN << 24)\
00093
   |(__STM3210C_EVAL_BSP_VERSION_SUB1 << 16)\
00094
```

```
   |(__STM3210C_EVAL_BSP_VERSION_SUB2 << 8 )\
00095
   |(__STM3210C_EVAL_BSP_VERSION_RC))
00096
00097
00098 /* Note: LCD /CS is CE4 - Bank 4 of NOR/SRAM
 Bank 1~4 */
00099 #define TFT_LCD_BASE           ((uint32_t)(0
x60000000 | 0x0C000000))
00100 #define TFT_LCD                ((TFT_LCD_Typ
eDef *) TFT_LCD_BASE)
00101
00102 /**
00103   * @}
00104   */
00105
00106
00107 /** @defgroup STM3210C_EVAL_Private_Variable
s Private Variables
00108   * @{
00109   */
00110 /**
00111  * @brief LED variables
00112  */
00113 GPIO_TypeDef* LED_PORT[LEDn] = {LED1_GPIO_PO
RT,
00114                                 LED2_GPIO_PO
RT,
00115                                 LED3_GPIO_PO
RT,
00116                                 LED4_GPIO_PO
RT};
00117
00118 const uint16_t LED_PIN[LEDn] = {LED1_PIN,
00119                                 LED2_PIN,
00120                                 LED3_PIN,
00121                                 LED4_PIN};
```

```
00122
00123 /**
00124  * @brief BUTTON variables
00125  */
00126 GPIO_TypeDef* BUTTON_PORT[BUTTONn] = {WAKEUP
_BUTTON_GPIO_PORT,
00127                                       TAMPER
_BUTTON_GPIO_PORT,
00128                                       KEY_BU
TTON_GPIO_PORT};
00129
00130 const uint16_t BUTTON_PIN[BUTTONn] = {WAKEUP
_BUTTON_PIN,
00131                                       TAMPER
_BUTTON_PIN,
00132                                       KEY_BU
TTON_PIN};
00133

00134 const uint16_t BUTTON_IRQn[BUTTONn] = {WAKEU
P_BUTTON_EXTI_IRQn,
00135                                        TAMPE
R_BUTTON_EXTI_IRQn,
00136                                        KEY_B
UTTON_EXTI_IRQn};
00137
00138
00139 /**
00140  * @brief COM variables
00141  */
00142 USART_TypeDef* COM_USART[COMn]   = {EVAL_COM1
};
00143
00144 GPIO_TypeDef* COM_TX_PORT[COMn]   = {EVAL_CO
M1_TX_GPIO_PORT};
00145
00146 GPIO_TypeDef* COM_RX_PORT[COMn]   = {EVAL_CO
```

```
M1_RX_GPIO_PORT};
00147
00148 const uint16_t COM_TX_PIN[COMn]   = {EVAL_CO
M1_TX_PIN};
00149
00150 const uint16_t COM_RX_PIN[COMn]   = {EVAL_CO
M1_RX_PIN};
00151
00152 /**
00153  * @brief BUS variables
00154  */
00155 #ifdef HAL_SPI_MODULE_ENABLED
00156 uint32_t SpixTimeout = EVAL_SPIx_TIMEOUT_MAX
;         /*<! Value of Timeout when SPI communicat
ion fails */
00157 static SPI_HandleTypeDef heval_Spi;
00158 #endif /* HAL_SPI_MODULE_ENABLED */
00159
00160 #ifdef HAL_I2C_MODULE_ENABLED
00161 uint32_t I2cxTimeout = EVAL_I2Cx_TIMEOUT_MAX
;   /*<! Value of Timeout when I2C communication f
ails */
00162 I2C_HandleTypeDef heval_I2c;
00163 #endif /* HAL_I2C_MODULE_ENABLED */
00164
00165 /**
00166   * @}
00167   */
00168
00169 /* I2Cx bus function */
00170 #ifdef HAL_I2C_MODULE_ENABLED
00171 /* Link function for I2C EEPROM peripheral */

00172 static void                I2Cx_Init(void);
00173 static void                I2Cx_ITConfig(void
);
00174 static HAL_StatusTypeDef  I2Cx_ReadMultiple(
```

```
uint8_t Addr, uint16_t Reg, uint16_t MemAddress, u
int8_t *Buffer, uint16_t Length);
00175 static HAL_StatusTypeDef  I2Cx_ReadBuffer(ui
nt16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_
t *pBuffer, uint16_t Length);
00176 static void               I2Cx_WriteData(uin
t16_t Addr, uint8_t Reg, uint8_t Value);
00177 static HAL_StatusTypeDef  I2Cx_WriteBuffer(u
int16_t Addr, uint8_t Reg, uint16_t RegSize, uint8
_t *pBuffer, uint16_t Length);
00178 static uint8_t            I2Cx_ReadData(uint
16_t Addr, uint8_t Reg);
00179 static HAL_StatusTypeDef  I2Cx_IsDeviceReady
(uint16_t DevAddress, uint32_t Trials);
00180 static void               I2Cx_Error(uint8_t
 Addr);
00181 static void               I2Cx_MspInit(I2C_H
andleTypeDef *hi2c);
00182
00183 /* Link function for IO Expander over I2C */
00184 void                      IOE_Init(void);
00185 void                      IOE_ITConfig(void)
;
00186 void                      IOE_Delay(uint32_t
 Delay);
00187 void                      IOE_Write(uint8_t
Addr, uint8_t Reg, uint8_t Value);
00188 uint8_t                   IOE_Read(uint8_t A
ddr, uint8_t Reg);
00189 uint16_t                  IOE_ReadMultiple(u
int8_t Addr, uint8_t Reg, uint8_t *Buffer, uint16_
t Length);
00190
00191 /* Link function for EEPROM peripheral over
I2C */
00192 void                      EEPROM_I2C_IO_Init(
void);
```

```
00193 HAL_StatusTypeDef         EEPROM_I2C_IO_Writ
eData(uint16_t DevAddress, uint16_t MemAddress, ui
nt8_t* pBuffer, uint32_t BufferSize);
00194 HAL_StatusTypeDef         EEPROM_I2C_IO_Read
Data(uint16_t DevAddress, uint16_t MemAddress, uin
t8_t* pBuffer, uint32_t BufferSize);
00195 HAL_StatusTypeDef         EEPROM_I2C_IO_IsDe
viceReady(uint16_t DevAddress, uint32_t Trials);
00196
00197 /* Link functions for Temperature Sensor per
ipheral */
00198 void                      TSENSOR_IO_Init(vo
id);
00199 void                      TSENSOR_IO_Write(u
int16_t DevAddress, uint8_t* pBuffer, uint8_t Writ
eAddr, uint16_t Length);
00200 void                      TSENSOR_IO_Read(ui
nt16_t DevAddress, uint8_t* pBuffer, uint8_t ReadA
ddr, uint16_t Length);
00201 uint16_t                  TSENSOR_IO_IsDevic
eReady(uint16_t DevAddress, uint32_t Trials);
00202
00203 /* Link function for Audio peripheral */
00204 void                      AUDIO_IO_Init(void
);
00205 void                      AUDIO_IO_Write(uin
t8_t Addr, uint8_t Reg, uint8_t Value);
00206 uint8_t                   AUDIO_IO_Read(uint
8_t Addr, uint8_t Reg);
00207
00208 /* Link function for Accelero peripheral */
00209 void                      ACCELERO_IO_Init(v
oid);
00210 void                      ACCELERO_IO_ITConf
ig(void);
00211 void                      ACCELERO_IO_Write(
uint8_t* pBuffer, uint8_t WriteAddr, uint16_t NumB
```

```
yteToWrite);
00212 void                    ACCELERO_IO_Read(u
int8_t* pBuffer, uint8_t ReadAddr, uint16_t NumByt
eToRead);
00213
00214 #endif /* HAL_I2C_MODULE_ENABLED */
00215
00216 /* SPIx bus function */
00217 #ifdef HAL_SPI_MODULE_ENABLED
00218 static void                SPIx_Init(void);
00219 static void                SPIx_Write(uint8_t
 Value);
00220 static uint32_t            SPIx_Read(void);
00221 static void                SPIx_Error (void);
00222 static void                SPIx_MspInit(SPI_H
andleTypeDef *hspi);
00223
00224 /* Link function for LCD peripheral over SPI
 */
00225 void                       LCD_IO_Init(void);
00226 void                       LCD_IO_WriteMultip
leData(uint8_t *pData, uint32_t Size);
00227 void                       LCD_IO_WriteReg(ui
nt8_t Reg);
00228 uint16_t                   LCD_IO_ReadData(ui
nt16_t RegValue);
00229 void                       LCD_Delay (uint32_
t delay);
00230
00231 /* Link functions for SD Card peripheral ove
r SPI */
00232 void                       SD_IO_Init(void);
00233 HAL_StatusTypeDef          SD_IO_WriteCmd(uin
t8_t Cmd, uint32_t Arg, uint8_t Crc, uint8_t Respo
nse);
00234 HAL_StatusTypeDef          SD_IO_WaitResponse
(uint8_t Response);
```

```
00235 void                            SD_IO_WriteDummy(v
oid);
00236 void                            SD_IO_WriteByte(ui
nt8_t Data);
00237 uint8_t                         SD_IO_ReadByte(void
);
00238
00239 #endif /* HAL_SPI_MODULE_ENABLED */
00240
00241
00242 /** @defgroup STM3210C_EVAL_Exported_Functio
ns Exported Functions
00243   * @{
00244   */
00245
00246 /**
00247   * @brief  This method returns the STM3210C
 EVAL BSP Driver revision
00248   * @retval version : 0xXYZR (8bits for each
 decimal, R for RC)
00249   */
00250 uint32_t BSP_GetVersion(void)
00251 {
00252   return __STM3210C_EVAL_BSP_VERSION;
00253 }
00254
00255 /**
00256   * @brief  Configures LED GPIO.
00257   * @param  Led: Specifies the Led to be con
figured.
00258   *   This parameter can be one of following
 parameters:
00259   *     @arg LED1
00260   *     @arg LED2
00261   *     @arg LED3
00262   *     @arg LED4
00263   * @retval None
```

```
00264   */
00265 void BSP_LED_Init(Led_TypeDef Led)
00266 {
00267   GPIO_InitTypeDef  gpioinitstruct = {0};
00268
00269   /* Enable the GPIO_LED clock */
00270   LEDx_GPIO_CLK_ENABLE(Led);
00271
00272   /* Configure the GPIO_LED pin */
00273   gpioinitstruct.Pin    = LED_PIN[Led];
00274   gpioinitstruct.Mode   = GPIO_MODE_OUTPUT_P
P;
00275   gpioinitstruct.Pull   = GPIO_NOPULL;
00276   gpioinitstruct.Speed  = GPIO_SPEED_FREQ_HI
GH;
00277
00278   HAL_GPIO_Init(LED_PORT[Led], &gpioinitstru
ct);
00279
00280   HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[L
ed], GPIO_PIN_RESET);
00281 }
00282
00283 /**
00284   * @brief  Turns selected LED On.
00285   * @param  Led: Specifies the Led to be set
 on.
00286   *   This parameter can be one of following
 parameters:
00287   *     @arg LED1
00288   *     @arg LED2
00289   *     @arg LED3
00290   *     @arg LED4
00291   * @retval None
00292   */
00293 void BSP_LED_On(Led_TypeDef Led)
00294 {
```

```
00295   HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[L
ed], GPIO_PIN_SET);
00296 }
00297
00298 /**
00299   * @brief  Turns selected LED Off.
00300   * @param  Led: Specifies the Led to be set
 off.
00301   *   This parameter can be one of following
 parameters:
00302   *     @arg LED1
00303   *     @arg LED2
00304   *     @arg LED3
00305   *     @arg LED4
00306   * @retval None
00307   */
00308 void BSP_LED_Off(Led_TypeDef Led)
00309 {
00310   HAL_GPIO_WritePin(LED_PORT[Led], LED_PIN[L
ed], GPIO_PIN_RESET);
00311 }
00312
00313 /**
00314   * @brief  Toggles the selected LED.
00315   * @param  Led: Specifies the Led to be tog
gled.
00316   *   This parameter can be one of following
 parameters:
00317   *     @arg LED1
00318   *     @arg LED2
00319   *     @arg LED3
00320   *     @arg LED4
00321   * @retval None
00322   */
00323 void BSP_LED_Toggle(Led_TypeDef Led)
00324 {
00325   HAL_GPIO_TogglePin(LED_PORT[Led], LED_PIN[
```

```
Led]);
00326 }
00327
00328 /**
00329   * @brief  Configures push button GPIO and
EXTI Line.
00330   * @param  Button: Button to be configured.
00331   *   This parameter can be one of the follo
wing values:
00332   *     @arg BUTTON_WAKEUP: Wakeup Push Butt
on
00333   *     @arg BUTTON_TAMPER: Tamper Push Butt
on
00334   *     @arg BUTTON_KEY: Key Push Button
00335   * @param  Button_Mode: Button mode request
ed.
00336   *   This parameter can be one of the follo
wing values:
00337   *     @arg BUTTON_MODE_GPIO: Button will b
e used as simple IO
00338   *     @arg BUTTON_MODE_EXTI: Button will b
e connected to EXTI line
00339   *                                with interrup
t generation capability
00340   * @retval None
00341   */
00342 void BSP_PB_Init(Button_TypeDef Button, Butt
onMode_TypeDef Button_Mode)
00343 {
00344   GPIO_InitTypeDef gpioinitstruct = {0};
00345
00346   /* Enable the corresponding Push Button cl
ock */
00347   BUTTONx_GPIO_CLK_ENABLE(Button);
00348
00349   /* Configure Push Button pin as input */
00350   gpioinitstruct.Pin    = BUTTON_PIN[Button]
```

```
;
00351   gpioinitstruct.Pull   = GPIO_NOPULL;
00352   gpioinitstruct.Speed  = GPIO_SPEED_FREQ_HI
GH;
00353
00354   if (Button_Mode == BUTTON_MODE_GPIO)
00355   {
00356     /* Configure Button pin as input */
00357     gpioinitstruct.Mode = GPIO_MODE_INPUT;
00358     HAL_GPIO_Init(BUTTON_PORT[Button], &gpio
initstruct);
00359   }
00360   else if (Button_Mode == BUTTON_MODE_EXTI)
00361   {
00362     if(Button != BUTTON_WAKEUP)
00363     {
00364       /* Configure Joystick Button pin as in
put with External interrupt, falling edge */
00365       gpioinitstruct.Mode = GPIO_MODE_IT_FAL
LING;
00366     }
00367     else
00368     {
00369       /* Configure Key Push Button pin as in
put with External interrupt, rising edge */
00370       gpioinitstruct.Mode = GPIO_MODE_IT_RIS
ING;
00371     }
00372     HAL_GPIO_Init(BUTTON_PORT[Button], &gpio
initstruct);
00373
00374     /* Enable and set Button EXTI Interrupt
to the lowest priority */
00375     HAL_NVIC_SetPriority((IRQn_Type)(BUTTON_
IRQn[Button]), 0x0F, 0);
00376     HAL_NVIC_EnableIRQ((IRQn_Type)(BUTTON_IR
Qn[Button]));
```

```
00377   }
00378 }
00379
00380 /**
00381   * @brief  Returns the selected button stat
e.
00382   * @param  Button: Button to be checked.
00383   *   This parameter can be one of the follo
wing values:
00384   *     @arg BUTTON_TAMPER: Key/Tamper Push
Button
00385   * @retval Button state
00386   */
00387 uint32_t BSP_PB_GetState(Button_TypeDef Butt
on)
00388 {
00389   return HAL_GPIO_ReadPin(BUTTON_PORT[Button
], BUTTON_PIN[Button]);
00390 }
00391
00392 #ifdef HAL_I2C_MODULE_ENABLED
00393 /**
00394   * @brief  Configures joystick GPIO and EXT
I modes.
00395   * @param  Joy_Mode: Button mode.
00396   *          This parameter can be one of th
e following values:
00397   *            @arg  JOY_MODE_GPIO: Joystick
 pins will be used as simple IOs
00398   *            @arg  JOY_MODE_EXTI: Joystick
 pins will be connected to EXTI line
00399   *                                 with int
errupt generation capability
00400   * @retval IO_OK: if all initializations ar
e OK. Other value if error.
00401   */
00402 uint8_t BSP_JOY_Init(JOYMode_TypeDef Joy_Mod
```

```
e)
00403 {
00404   uint8_t ret = 0;
00405
00406   /* Initialize the IO functionalities */
00407   ret = BSP_IO_Init();
00408
00409   /* Configure joystick pins in IT mode */
00410   if((ret == IO_OK) && (Joy_Mode == JOY_MODE
_EXTI))
00411   {
00412     /* Configure joystick pins in IT mode */
00413     BSP_IO_ConfigPin(JOY_ALL_PINS, IO_MODE_I
T_FALLING_EDGE);
00414   }
00415
00416   return ret;
00417 }
00418
00419 /**
00420   * @brief  Returns the current joystick sta
tus.
00421   * @retval Code of the joystick key pressed
00422   *          This code can be one of the fol
lowing values:
00423   *            @arg  JOY_NONE
00424   *            @arg  JOY_SEL
00425   *            @arg  JOY_DOWN
00426   *            @arg  JOY_LEFT
00427   *            @arg  JOY_RIGHT
00428   *            @arg  JOY_UP
00429   */
00430 JOYState_TypeDef BSP_JOY_GetState(void)
00431 {
00432   uint32_t tmp = 0;
00433
00434   /* Read the status joystick pins */
```

```
00435   tmp = BSP_IO_ReadPin(JOY_ALL_PINS);
00436
00437   /* Check the pressed keys */
00438   if((tmp & JOY_NONE_PIN) == JOY_NONE)
00439   {
00440     return(JOYState_TypeDef) JOY_NONE;
00441   }
00442   else if(!(tmp & JOY_SEL_PIN))
00443   {
00444     return(JOYState_TypeDef) JOY_SEL;
00445   }
00446   else if(!(tmp & JOY_DOWN_PIN))
00447   {
00448     return(JOYState_TypeDef) JOY_DOWN;
00449   }
00450   else if(!(tmp & JOY_LEFT_PIN))
00451   {
00452     return(JOYState_TypeDef) JOY_LEFT;
00453   }
00454   else if(!(tmp & JOY_RIGHT_PIN))
00455   {
00456     return(JOYState_TypeDef) JOY_RIGHT;
00457   }
00458   else if(!(tmp & JOY_UP_PIN))
00459   {
00460     return(JOYState_TypeDef) JOY_UP;
00461   }
00462   else
00463   {
00464     return(JOYState_TypeDef) JOY_NONE;
00465   }
00466 }
00467 #endif /*HAL_I2C_MODULE_ENABLED*/
00468
00469 #ifdef HAL_UART_MODULE_ENABLED
00470 /**
00471   * @brief  Configures COM port.
```

```
00472   * @param  COM: Specifies the COM port to b
e configured.
00473   *   This parameter can be one of following
 parameters:
00474   *     @arg COM1
00475   * @param  huart: pointer to a UART_HandleT
ypeDef structure that
00476   *   contains the configuration information
 for the specified UART peripheral.
00477   * @retval None
00478   */
00479 void BSP_COM_Init(COM_TypeDef COM, UART_Hand
leTypeDef* huart)
00480 {
00481   GPIO_InitTypeDef gpioinitstruct = {0};
00482
00483   /* Enable GPIO clock */
00484   COMx_TX_GPIO_CLK_ENABLE(COM);
00485   COMx_RX_GPIO_CLK_ENABLE(COM);
00486
00487   /* Enable USART clock */
00488   COMx_CLK_ENABLE(COM);
00489
00490   /* Remap AFIO if needed */
00491   AFIOCOMx_CLK_ENABLE(COM);
00492   AFIOCOMx_REMAP(COM);
00493
00494   /* Configure USART Tx as alternate functio
n push-pull */
00495   gpioinitstruct.Pin        = COM_TX_PIN[COM
];
00496   gpioinitstruct.Mode       = GPIO_MODE_AF_P
P;
00497   gpioinitstruct.Speed      = GPIO_SPEED_FRE
Q_HIGH;
00498   gpioinitstruct.Pull       = GPIO_PULLUP;
00499   HAL_GPIO_Init(COM_TX_PORT[COM], &gpioinits
```

```
truct);
00500
00501   /* Configure USART Rx as alternate functio
n push-pull */
00502   gpioinitstruct.Mode       = GPIO_MODE_INPU
T;
00503   gpioinitstruct.Pin        = COM_RX_PIN[COM
];
00504   HAL_GPIO_Init(COM_RX_PORT[COM], &gpioinits
truct);
00505
00506   /* USART configuration */
00507   huart->Instance = COM_USART[COM];
00508   HAL_UART_Init(huart);
00509 }
00510 #endif /* HAL_UART_MODULE_ENABLED */
00511
00512 /**
00513   * @}
00514   */
00515
00516 /** @defgroup STM3210C_EVAL_BusOperations_Fu
nctions Bus Operations Functions
00517   * @{
00518   */
00519
00520 /*******************************************
***********************************
00521                                     BUS OPERATIONS
00522 *******************************************
***********************************/
00523
00524 #ifdef HAL_I2C_MODULE_ENABLED
00525 /******************************** I2C Routine
s********************************/
00526
00527 /**
```

```
00528   * @brief Eval I2Cx MSP Initialization
00529   * @param hi2c: I2C handle
00530   * @retval None
00531   */
00532 static void I2Cx_MspInit(I2C_HandleTypeDef *
hi2c)
00533 {
00534   GPIO_InitTypeDef  gpioinitstruct = {0};
00535
00536   if (hi2c->Instance == EVAL_I2Cx)
00537   {
00538     /*## Configure the GPIOs ##############
##################################*/
00539
00540     /* Enable GPIO clock */
00541     EVAL_I2Cx_SDA_GPIO_CLK_ENABLE();
00542     EVAL_I2Cx_SCL_GPIO_CLK_ENABLE();
00543
00544     /* Configure I2C Tx as alternate functio
n  */
00545     gpioinitstruct.Pin       = EVAL_I2Cx_SCL
_PIN;
00546     gpioinitstruct.Mode      = GPIO_MODE_AF_
OD;
00547     gpioinitstruct.Pull      = GPIO_NOPULL;
00548     gpioinitstruct.Speed     = GPIO_SPEED_FR
EQ_HIGH;
00549     HAL_GPIO_Init(EVAL_I2Cx_SCL_GPIO_PORT, &
gpioinitstruct);
00550
00551     /* Configure I2C Rx as alternate functio
n  */
00552     gpioinitstruct.Pin = EVAL_I2Cx_SDA_PIN;
00553     HAL_GPIO_Init(EVAL_I2Cx_SDA_GPIO_PORT, &
gpioinitstruct);
00554
00555     /*## Configure the Eval I2Cx peripheral
```

```
########################################*/
00556     /* Enable Eval_I2Cx clock */
00557     EVAL_I2Cx_CLK_ENABLE();
00558
00559     /* Add delay related to RCC workaround */

00560     while (READ_BIT(RCC->APB1ENR, RCC_APB1EN
R_I2C1EN) != RCC_APB1ENR_I2C1EN) {};
00561
00562     /* Force the I2C Periheral Clock Reset */

00563     EVAL_I2Cx_FORCE_RESET();
00564
00565     /* Release the I2C Periheral Clock Reset
 */
00566     EVAL_I2Cx_RELEASE_RESET();
00567
00568     /* Enable and set Eval I2Cx Interrupt to
 the highest priority */
00569     HAL_NVIC_SetPriority(EVAL_I2Cx_EV_IRQn,
5, 0);
00570     HAL_NVIC_EnableIRQ(EVAL_I2Cx_EV_IRQn);
00571
00572     /* Enable and set Eval I2Cx Interrupt to
 the highest priority */
00573     HAL_NVIC_SetPriority(EVAL_I2Cx_ER_IRQn,
5, 0);
00574     HAL_NVIC_EnableIRQ(EVAL_I2Cx_ER_IRQn);
00575   }
00576 }
00577
00578 /**
00579   * @brief Eval I2Cx Bus initialization
00580   * @retval None
00581   */
00582 static void I2Cx_Init(void)
00583 {
```

```
00584   if(HAL_I2C_GetState(&heval_I2c) == HAL_I2C
_STATE_RESET)
00585   {
00586     heval_I2c.Instance              = EVAL_I
2Cx;
00587     heval_I2c.Init.ClockSpeed       = BSP_I2
C_SPEED;
00588     heval_I2c.Init.DutyCycle        = I2C_DU
TYCYCLE_2;
00589     heval_I2c.Init.OwnAddress1      = 0;
00590     heval_I2c.Init.AddressingMode   = I2C_AD
DRESSINGMODE_7BIT;
00591     heval_I2c.Init.DualAddressMode  = I2C_DU
ALADDRESS_DISABLE;
00592     heval_I2c.Init.OwnAddress2      = 0;
00593     heval_I2c.Init.GeneralCallMode  = I2C_GE
NERALCALL_DISABLE;
00594     heval_I2c.Init.NoStretchMode    = I2C_NO
STRETCH_DISABLE;
00595
00596     /* Init the I2C */
00597     I2Cx_MspInit(&heval_I2c);
00598     HAL_I2C_Init(&heval_I2c);
00599   }
00600 }
00601
00602 /**
00603   * @brief  Configures I2C Interrupt.
00604   * @retval None
00605   */
00606 static void I2Cx_ITConfig(void)
00607 {
00608   static uint8_t I2C_IT_Enabled = 0;
00609   GPIO_InitTypeDef  gpioinitstruct = {0};
00610
00611   if(I2C_IT_Enabled == 0)
00612   {
```

```
00613     I2C_IT_Enabled = 1;
00614
00615     /* Enable the GPIO EXTI clock */
00616     IOE_IT_GPIO_CLK_ENABLE();
00617
00618     gpioinitstruct.Pin   = IOE_IT_PIN;
00619     gpioinitstruct.Pull  = GPIO_NOPULL;
00620     gpioinitstruct.Speed = GPIO_SPEED_FREQ_H
IGH;
00621     gpioinitstruct.Mode  = GPIO_MODE_IT_FALL
ING;
00622     HAL_GPIO_Init(IOE_IT_GPIO_PORT, &gpioini
tstruct);
00623
00624     /* Set priority and Enable GPIO EXTI Int
errupt */
00625     HAL_NVIC_SetPriority((IRQn_Type)(IOE_IT_
EXTI_IRQn), 5, 0);
00626     HAL_NVIC_EnableIRQ((IRQn_Type)(IOE_IT_EX
TI_IRQn));
00627   }
00628 }
00629
00630 /**
00631   * @brief  Reads multiple data.
00632   * @param  Addr: I2C address
00633   * @param  Reg: Reg address
00634   * @param  MemAddress: Internal memory addr
ess
00635   * @param  Buffer: Pointer to data buffer
00636   * @param  Length: Length of the data
00637   * @retval Number of read data
00638   */
00639 static HAL_StatusTypeDef I2Cx_ReadMultiple(u
int8_t Addr, uint16_t Reg, uint16_t MemAddress, ui
nt8_t *Buffer, uint16_t Length)
00640 {
```

```
00641   HAL_StatusTypeDef status = HAL_OK;
00642
00643   status = HAL_I2C_Mem_Read(&heval_I2c, Addr
, (uint16_t)Reg, MemAddress, Buffer, Length, I2cxT
imeout);
00644
00645   /* Check the communication status */
00646   if(status != HAL_OK)
00647   {
00648     /* I2C error occured */
00649     I2Cx_Error(Addr);
00650   }
00651   return status;
00652 }
00653
00654 /**
00655   * @brief  Write a value in a register of t
he device through BUS.
00656   * @param  Addr: Device address on BUS Bus.

00657   * @param  Reg: The target register address
 to write
00658   * @param  Value: The target register value
 to be written
00659   * @retval  None
00660   */
00661 static void I2Cx_WriteData(uint16_t Addr, ui
nt8_t Reg, uint8_t Value)
00662 {
00663   HAL_StatusTypeDef status = HAL_OK;
00664
00665   status = HAL_I2C_Mem_Write(&heval_I2c, Add
r, (uint16_t)Reg, I2C_MEMADD_SIZE_8BIT, &Value, 1,
I2cxTimeout);
00666
00667   /* Check the communication status */
00668   if(status != HAL_OK)
```

```
00669   {
00670     /* Execute user timeout callback */
00671     I2Cx_Error(Addr);
00672   }
00673 }
00674
00675 /**
00676   * @brief  Write a value in a register of t
he device through BUS.
00677   * @param  Addr: Device address on BUS Bus.

00678   * @param  Reg: The target register address
 to write
00679   * @param  RegSize: The target register siz
e (can be 8BIT or 16BIT)
00680   * @param  pBuffer: The target register val
ue to be written
00681   * @param  Length: buffer size to be written

00682   * @retval None
00683   */
00684 static HAL_StatusTypeDef I2Cx_WriteBuffer(ui
nt16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_
t *pBuffer, uint16_t Length)
00685 {
00686   HAL_StatusTypeDef status = HAL_OK;
00687
00688   status = HAL_I2C_Mem_Write(&heval_I2c, Add
r, (uint16_t)Reg, RegSize, pBuffer, Length, I2cxTi
meout);
00689
00690 /* Check the communication status */
00691   if(status != HAL_OK)
00692   {
00693     /* Re-Initiaize the BUS */
00694     I2Cx_Error(Addr);
00695   }
```

```
00696   return status;
00697 }
00698
00699 /**
00700   * @brief  Read a value in a register of th
e device through BUS.
00701   * @param  Addr: Device address on BUS Bus.

00702   * @param  Reg: The target register address
 to write
00703   * @retval Data read at register @
00704   */
00705 static uint8_t I2Cx_ReadData(uint16_t Addr,
uint8_t Reg)
00706 {
00707   HAL_StatusTypeDef status = HAL_OK;
00708   uint8_t value = 0;
00709
00710   status = HAL_I2C_Mem_Read(&heval_I2c, Addr
, Reg, I2C_MEMADD_SIZE_8BIT, &value, 1, I2cxTimeout
);
00711
00712   /* Check the communication status */
00713   if(status != HAL_OK)
00714   {
00715     /* Execute user timeout callback */
00716     I2Cx_Error(Addr);
00717
00718   }
00719   return value;
00720 }
00721
00722 /**
00723   * @brief  Reads multiple data on the BUS.
00724   * @param  Addr: I2C Address
00725   * @param  Reg: Reg Address
00726   * @param  RegSize : The target register si
```

```
ze (can be 8BIT or 16BIT)
00727   * @param  pBuffer: pointer to read data bu
ffer
00728   * @param  Length: length of the data
00729   * @retval 0 if no problems to read multipl
e data
00730   */
00731 static HAL_StatusTypeDef I2Cx_ReadBuffer(uin
t16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t
 *pBuffer, uint16_t Length)
00732 {
00733   HAL_StatusTypeDef status = HAL_OK;
00734
00735   status = HAL_I2C_Mem_Read(&heval_I2c, Addr
, (uint16_t)Reg, RegSize, pBuffer, Length, I2cxTim
eout);
00736
00737   /* Check the communication status */
00738   if(status != HAL_OK)
00739   {
00740     /* Re-Initiaize the BUS */
00741     I2Cx_Error(Addr);
00742   }
00743   return status;
00744 }
00745
00746 /**
00747 * @brief  Checks if target device is ready f
or communication.
00748 * @note   This function is used with Memory
devices
00749 * @param  DevAddress: Target device address
00750 * @param  Trials: Number of trials
00751 * @retval HAL status
00752 */
00753 static HAL_StatusTypeDef I2Cx_IsDeviceReady(
uint16_t DevAddress, uint32_t Trials)
```

```
00754 {
00755   return (HAL_I2C_IsDeviceReady(&heval_I2c,
DevAddress, Trials, I2cxTimeout));
00756 }
00757
00758 /**
00759   * @brief  Manages error callback by re-ini
tializing I2C.
00760   * @param  Addr: I2C Address
00761   * @retval None
00762   */
00763 static void I2Cx_Error(uint8_t Addr)
00764 {
00765   /* De-initialize the IOE comunication BUS
*/
00766   HAL_I2C_DeInit(&heval_I2c);
00767
00768   /* Re-Initiaize the IOE comunication BUS */

00769   I2Cx_Init();
00770 }
00771
00772 #endif /* HAL_I2C_MODULE_ENABLED */
00773
00774 /****************************** SPI Routine
s******************************/
00775 #ifdef HAL_SPI_MODULE_ENABLED
00776 /**
00777   * @brief  Initializes SPI MSP.
00778   * @retval None
00779   */
00780 static void SPIx_MspInit(SPI_HandleTypeDef *
hspi)
00781 {
00782   GPIO_InitTypeDef  gpioinitstruct = {0};
00783
00784   /*** Configure the GPIOs ***/
```

```
00785   /* Enable GPIO clock */
00786   EVAL_SPIx_SCK_GPIO_CLK_ENABLE();
00787   EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE();
00788   __HAL_RCC_AFIO_CLK_ENABLE();
00789   __HAL_AFIO_REMAP_SPI3_ENABLE();
00790
00791   /* configure SPI SCK */
00792   gpioinitstruct.Pin        = EVAL_SPIx_SCK_
PIN;
00793   gpioinitstruct.Mode       = GPIO_MODE_AF_P
P;
00794   gpioinitstruct.Pull       = GPIO_NOPULL;
00795   gpioinitstruct.Speed      = GPIO_SPEED_FRE
Q_HIGH;
00796   HAL_GPIO_Init(EVAL_SPIx_SCK_GPIO_PORT, &gp
ioinitstruct);
00797
00798   /* configure SPI MISO and MOSI */
00799   gpioinitstruct.Pin        = (EVAL_SPIx_MIS
O_PIN | EVAL_SPIx_MOSI_PIN);
00800   gpioinitstruct.Mode       = GPIO_MODE_AF_P
P;
00801   gpioinitstruct.Pull       = GPIO_NOPULL;
00802   gpioinitstruct.Speed      = GPIO_SPEED_FRE
Q_HIGH;
00803   HAL_GPIO_Init(EVAL_SPIx_MISO_MOSI_GPIO_PORT
, &gpioinitstruct);
00804
00805   /*** Configure the SPI peripheral ***/
00806   /* Enable SPI clock */
00807   EVAL_SPIx_CLK_ENABLE();
00808 }
00809
00810 /**
00811   * @brief  Initializes SPI HAL.
00812   * @retval None
00813   */
```

```
00814 static void SPIx_Init(void)
00815 {
00816   /* DeInitializes the SPI peripheral */
00817   heval_Spi.Instance = EVAL_SPIx;
00818   HAL_SPI_DeInit(&heval_Spi);
00819
00820   /* SPI Config */
00821   /* SPI baudrate is set to 9 MHz (PCLK2/SPI
_BaudRatePrescaler = 72/8 = 9 MHz) */
00822   heval_Spi.Init.BaudRatePrescaler  = SPI_BA
UDRATEPRESCALER_8;
00823   heval_Spi.Init.Direction          = SPI_DI
RECTION_2LINES;
00824   heval_Spi.Init.CLKPhase           = SPI_PH
ASE_2EDGE;
00825   heval_Spi.Init.CLKPolarity        = SPI_PO
LARITY_HIGH;
00826   heval_Spi.Init.CRCCalculation     = SPI_CR
CCALCULATION_DISABLE;
00827   heval_Spi.Init.CRCPolynomial      = 7;
00828   heval_Spi.Init.DataSize           = SPI_DA
TASIZE_8BIT;
00829   heval_Spi.Init.FirstBit           = SPI_FI
RSTBIT_MSB;
00830   heval_Spi.Init.NSS                = SPI_NS
S_SOFT;
00831   heval_Spi.Init.TIMode             = SPI_TI
MODE_DISABLE;
00832   heval_Spi.Init.Mode               = SPI_MO
DE_MASTER;
00833
00834   SPIx_MspInit(&heval_Spi);
00835   if (HAL_SPI_Init(&heval_Spi) != HAL_OK)
00836   {
00837     /* Should not occur */
00838     while(1) {};
00839   }
```

```
00840 }
00841
00842 /**
00843   * @brief SPI Read 4 bytes from device
00844   * @retval Read data
00845 */
00846 static uint32_t SPIx_Read(void)
00847 {
00848   HAL_StatusTypeDef status = HAL_OK;
00849   uint32_t          readvalue = 0;
00850   uint32_t          writevalue = 0xFFFFFFFF;
00851
00852   status = HAL_SPI_TransmitReceive(&heval_Spi
, (uint8_t*) &writevalue, (uint8_t*) &readvalue, 1
, SpixTimeout);
00853
00854   /* Check the communication status */
00855   if(status != HAL_OK)
00856   {
00857     /* Execute user timeout callback */
00858     SPIx_Error();
00859   }
00860
00861   return readvalue;
00862 }
00863
00864 /**
00865   * @brief SPI Write a byte to device
00866   * @param Value: value to be written
00867   * @retval None
00868   */
00869 static void SPIx_Write(uint8_t Value)
00870 {
00871   HAL_StatusTypeDef status = HAL_OK;
00872
00873   status = HAL_SPI_Transmit(&heval_Spi, (uin
t8_t*) &Value, 1, SpixTimeout);
```

```
00874
00875   /* Check the communication status */
00876   if(status != HAL_OK)
00877   {
00878     /* Execute user timeout callback */
00879     SPIx_Error();
00880   }
00881 }
00882
00883 /**
00884   * @brief SPI error treatment function
00885   * @retval None
00886   */
00887 static void SPIx_Error (void)
00888 {
00889   /* De-initialize the SPI communication BUS
 */
00890   HAL_SPI_DeInit(&heval_Spi);
00891
00892   /* Re- Initiaize the SPI communication BUS
 */
00893   SPIx_Init();
00894 }
00895 #endif /* HAL_SPI_MODULE_ENABLED */
00896
00897 /**
00898   * @}
00899   */
00900
00901 /** @defgroup STM3210C_EVAL_LinkOperations_F
unctions Link Operations Functions
00902   * @{
00903   */
00904
00905 /*************************************************
*********************************
00906                                     LINK OPERATIONS
```

```
00907 ************************************************
**********************************/
00908
00909 #ifdef HAL_I2C_MODULE_ENABLED
00910 /****************************** LINK IOE ****
**********************************/
00911
00912 /**
00913   * @brief  Initializes IOE low level.
00914   * @retval None
00915   */
00916 void IOE_Init(void)
00917 {
00918   I2Cx_Init();
00919 }
00920
00921 /**
00922   * @brief  Configures IOE low level Interru
pt.
00923   * @retval None
00924   */
00925 void IOE_ITConfig(void)
00926 {
00927   I2Cx_ITConfig();
00928 }
00929
00930 /**
00931   * @brief  IOE writes single data.
00932   * @param  Addr: I2C address
00933   * @param  Reg: Reg address
00934   * @param  Value: Data to be written
00935   * @retval None
00936   */
00937 void IOE_Write(uint8_t Addr, uint8_t Reg, ui
nt8_t Value)
00938 {
00939   I2Cx_WriteData(Addr, Reg, Value);
```

```
00940 }
00941
00942 /**
00943   * @brief  IOE reads single data.
00944   * @param  Addr: I2C address
00945   * @param  Reg: Reg address
00946   * @retval Read data
00947   */
00948 uint8_t IOE_Read(uint8_t Addr, uint8_t Reg)
00949 {
00950   return I2Cx_ReadData(Addr, Reg);
00951 }
00952
00953 /**
00954   * @brief  IOE reads multiple data.
00955   * @param  Addr: I2C address
00956   * @param  Reg: Reg address
00957   * @param  Buffer: Pointer to data buffer
00958   * @param  Length: Length of the data
00959   * @retval Number of read data
00960   */
00961 uint16_t IOE_ReadMultiple(uint8_t Addr, uint
8_t Reg, uint8_t *Buffer, uint16_t Length)
00962 {
00963  return I2Cx_ReadMultiple(Addr, Reg, I2C_MEM
ADD_SIZE_8BIT, Buffer, Length);
00964 }
00965
00966 /**
00967   * @brief  IOE delay.
00968   * @param  Delay: Delay in ms
00969   * @retval None
00970   */
00971 void IOE_Delay(uint32_t Delay)
00972 {
00973   HAL_Delay(Delay);
00974 }
```

```
00975
00976 #endif /* HAL_I2C_MODULE_ENABLED */
00977
00978 #ifdef HAL_SPI_MODULE_ENABLED
00979 /****************************** LINK LCD
*******************************/
00980
00981 /**
00982   * @brief  Configures the LCD_SPI interface.

00983   * @retval None
00984   */
00985 void LCD_IO_Init(void)
00986 {
00987   GPIO_InitTypeDef gpioinitstruct;
00988
00989   /* Configure the LCD Control pins --------
---------------------------------*/
00990   LCD_NCS_GPIO_CLK_ENABLE();
00991
00992   /* Configure NCS in Output Push-Pull mode
*/
00993   gpioinitstruct.Pin     = LCD_NCS_PIN;
00994   gpioinitstruct.Mode    = GPIO_MODE_OUTPUT_
PP;
00995   gpioinitstruct.Pull    = GPIO_NOPULL;
00996   gpioinitstruct.Speed   = GPIO_SPEED_FREQ_H
IGH;
00997   HAL_GPIO_Init(LCD_NCS_GPIO_PORT, &gpioinit
struct);
00998
00999   /* Set or Reset the control line */
01000   LCD_CS_LOW();
01001   LCD_CS_HIGH();
01002
01003   SPIx_Init();
01004 }
```

```
01005
01006 /**
01007   * @brief  Write register value.
01008   * @param  pData Pointer on the register va
lue
01009   * @param  Size Size of byte to transmit to
 the register
01010   * @retval None
01011   */
01012 void LCD_IO_WriteMultipleData(uint8_t *pData
, uint32_t Size)
01013 {
01014   uint32_t counter = 0;
01015
01016   /* Reset LCD control line(/CS) and Send da
ta */
01017   LCD_CS_LOW();
01018
01019   /* Send Start Byte */
01020   SPIx_Write(START_BYTE | LCD_WRITE_REG);
01021
01022   for (counter = Size; counter != 0; counter
--)
01023   {
01024     while(((heval_Spi.Instance->SR) & SPI_FL
AG_TXE) != SPI_FLAG_TXE)
01025     {
01026     }
01027     /* Need to invert bytes for LCD*/
01028     *((__IO uint8_t*)&heval_Spi.Instance->DR
) = *(pData+1);
01029
01030     while(((heval_Spi.Instance->SR) & SPI_FL
AG_TXE) != SPI_FLAG_TXE)
01031     {
01032     }
01033     *((__IO uint8_t*)&heval_Spi.Instance->DR
```

```
) = *pData;
01034     counter--;
01035     pData += 2;
01036   }
01037
01038   /* Wait until the bus is ready before rele
asing Chip select */
01039   while(((heval_Spi.Instance->SR) & SPI_FLAG
_BSY) != RESET)
01040   {
01041   }
01042
01043   /* Reset LCD control line(/CS) and Send da
ta */
01044   LCD_CS_HIGH();
01045 }
01046
01047 /**
01048   * @brief  register address.
01049   * @param  Reg
01050   * @retval None
01051   */
01052 void LCD_IO_WriteReg(uint8_t Reg)
01053 {
01054   /* Reset LCD control line(/CS) and Send co
mmand */
01055   LCD_CS_LOW();
01056
01057   /* Send Start Byte */
01058   SPIx_Write(START_BYTE | SET_INDEX);
01059
01060   /* Write 16-bit Reg Index (High Byte is 0)
 */
01061   SPIx_Write(0x00);
01062   SPIx_Write(Reg);
01063
01064   /* Deselect : Chip Select high */
```

```
01065  LCD_CS_HIGH();
01066 }
01067
01068 /**
01069   * @brief  Read register value.
01070   * @param  Reg
01071   * @retval None
01072   */
01073 uint16_t LCD_IO_ReadData(uint16_t Reg)
01074 {
01075   uint32_t readvalue = 0;
01076
01077   /* Send Reg value to Read */
01078   LCD_IO_WriteReg(Reg);
01079
01080   /* Reset LCD control line(/CS) and Send co
mmand */
01081   LCD_CS_LOW();
01082
01083   /* Send Start Byte */
01084   SPIx_Write(START_BYTE | LCD_READ_REG);
01085   /* Read Upper Byte */
01086   SPIx_Write(0xFF);
01087   readvalue = SPIx_Read();
01088   readvalue = readvalue << 8;
01089   readvalue |= SPIx_Read();
01090
01091   HAL_Delay(10);
01092
01093   /* Deselect : Chip Select high */
01094   LCD_CS_HIGH();
01095   return readvalue;
01096 }
01097
01098 /**
01099   * @brief  Wait for loop in ms.
01100   * @param  Delay in ms.
```

```
01101   * @retval None
01102   */
01103 void LCD_Delay (uint32_t Delay)
01104 {
01105   HAL_Delay(Delay);
01106 }
01107
01108 /********************************** LINK SD Ca
rd ************************************/
01109
01110 /**
01111   * @brief  Initializes the SD Card and put
it into StandBy State (Ready for
01112   *         data transfer).
01113   * @retval None
01114   */
01115 void SD_IO_Init(void)
01116 {
01117   GPIO_InitTypeDef  gpioinitstruct;
01118   uint8_t counter;
01119
01120   /* SD_CS_GPIO and SD_DETECT_GPIO Periph cl
ock enable */
01121   SD_CS_GPIO_CLK_ENABLE();
01122   SD_DETECT_GPIO_CLK_ENABLE();
01123
01124   /* Configure SD_CS_PIN pin: SD Card CS pin
 */
01125   gpioinitstruct.Pin    = SD_CS_PIN;
01126   gpioinitstruct.Mode   = GPIO_MODE_OUTPUT_P
P;
01127   gpioinitstruct.Pull   = GPIO_PULLUP;
01128   gpioinitstruct.Speed  = GPIO_SPEED_FREQ_HI
GH;
01129   HAL_GPIO_Init(SD_CS_GPIO_PORT, &gpioinitst
ruct);
01130
```

```
01131   /* Configure SD_DETECT_PIN pin: SD Card de
tect pin */
01132   gpioinitstruct.Pin    = SD_DETECT_PIN;
01133   gpioinitstruct.Mode   = GPIO_MODE_IT_RISIN
G_FALLING;
01134   gpioinitstruct.Pull   = GPIO_PULLUP;
01135   HAL_GPIO_Init(SD_DETECT_GPIO_PORT, &gpioin
itstruct);
01136
01137   /* Enable and set SD EXTI Interrupt to the
 lowest priority */
01138   HAL_NVIC_SetPriority(SD_DETECT_EXTI_IRQn,
0x0F, 0);
01139   HAL_NVIC_EnableIRQ(SD_DETECT_EXTI_IRQn);
01140
01141   /*------------Put SD in SPI mode----------
----*/
01142   /* SD SPI Config */
01143   SPIx_Init();
01144
01145   /* SD chip select high */
01146   SD_CS_HIGH();
01147
01148   /* Send dummy byte 0xFF, 10 times with CS
high */
01149   /* Rise CS and MOSI for 80 clocks cycles */

01150   for (counter = 0; counter <= 9; counter++)
01151   {
01152     /* Send dummy byte 0xFF */
01153     SD_IO_WriteByte(SD_DUMMY_BYTE);
01154   }
01155 }
01156
01157 /**
01158   * @brief  Write a byte on the SD.
01159   * @param  Data: byte to send.
```

```
01160   * @retval None
01161   */
01162 void SD_IO_WriteByte(uint8_t Data)
01163 {
01164   /* Send the byte */
01165   SPIx_Write(Data);
01166 }
01167
01168 /**
01169   * @brief  Read a byte from the SD.
01170   * @retval The received byte.
01171   */
01172 uint8_t SD_IO_ReadByte(void)
01173 {
01174   uint8_t data = 0;
01175
01176   /* Get the received data */
01177   data = SPIx_Read();
01178
01179   /* Return the shifted data */
01180   return data;
01181 }
01182
01183 /**
01184   * @brief  Send 5 bytes command to the SD c
ard and get response
01185   * @param  Cmd: The user expected command t
o send to SD card.
01186   * @param  Arg: The command argument.
01187   * @param  Crc: The CRC.
01188   * @param  Response: Expected response from
 the SD card
01189   * @retval  HAL_StatusTypeDef HAL Status
01190   */
01191 HAL_StatusTypeDef SD_IO_WriteCmd(uint8_t Cmd
, uint32_t Arg, uint8_t Crc, uint8_t Response)
01192 {
```

```
01193   uint32_t counter = 0x00;
01194   uint8_t frame[6];
01195
01196   /* Prepare Frame to send */
01197   frame[0] = (Cmd | 0x40); /* Construct byte
 1 */
01198   frame[1] = (uint8_t)(Arg >> 24); /* Constr
uct byte 2 */
01199   frame[2] = (uint8_t)(Arg >> 16); /* Constr
uct byte 3 */
01200   frame[3] = (uint8_t)(Arg >> 8); /* Constru
ct byte 4 */
01201   frame[4] = (uint8_t)(Arg); /* Construct by
te 5 */
01202   frame[5] = (Crc); /* Construct CRC: byte 6
 */
01203
01204   /* SD chip select low */
01205   SD_CS_LOW();
01206
01207   /* Send Frame */
01208   for (counter = 0; counter < 6; counter++)
01209   {
01210     SD_IO_WriteByte(frame[counter]); /* Send
 the Cmd bytes */
01211   }
01212
01213   if(Response != SD_NO_RESPONSE_EXPECTED)
01214   {
01215     return SD_IO_WaitResponse(Response);
01216   }
01217
01218   return HAL_OK;
01219 }
01220
01221 /**
01222   * @brief  Wait response from the SD card
```

```
01223   * @param  Response: Expected response from
 the SD card
01224   * @retval  HAL_StatusTypeDef HAL Status
01225   */
01226 HAL_StatusTypeDef SD_IO_WaitResponse(uint8_t
 Response)
01227 {
01228   uint32_t timeout = 0xFFFF;
01229   uint8_t resp = 0;
01230   /* Check if response is got or a timeout i
s happen */
01231   resp = SD_IO_ReadByte();
01232   while ((resp != Response) && timeout)
01233   {
01234     timeout--;
01235     resp = SD_IO_ReadByte();
01236   }
01237
01238   if (timeout == 0)
01239   {
01240     /* After time out */
01241     return HAL_TIMEOUT;
01242   }
01243   else
01244   {
01245     /* Right response got */
01246     return HAL_OK;
01247   }
01248 }
01249
01250 /**
01251   * @brief  Send dummy byte with CS High
01252   * @retval None
01253   */
01254 void SD_IO_WriteDummy(void)
01255 {
01256     /* SD chip select high */
```

```
01257     SD_CS_HIGH();
01258
01259     /* Send Dummy byte 0xFF */
01260     SD_IO_WriteByte(SD_DUMMY_BYTE);
01261 }
01262
01263 #endif /* HAL_SPI_MODULE_ENABLED */
01264
01265 #ifdef HAL_I2C_MODULE_ENABLED
01266 /****************************** LINK I2C
EEPROM *****************************/
01267 /**
01268   * @brief  Initializes peripherals used by
the I2C EEPROM driver.
01269   * @retval None
01270   */
01271 void EEPROM_I2C_IO_Init(void)
01272 {
01273   I2Cx_Init();
01274 }
01275
01276 /**
01277   * @brief  Write data to I2C EEPROM driver
01278   * @param  DevAddress: Target device address

01279   * @param  MemAddress: Internal memory addr
ess
01280   * @param  pBuffer: Pointer to data buffer
01281   * @param  BufferSize: Amount of data to be
 sent
01282   * @retval HAL status
01283   */
01284 HAL_StatusTypeDef EEPROM_I2C_IO_WriteData(ui
nt16_t DevAddress, uint16_t MemAddress, uint8_t* p
Buffer, uint32_t BufferSize)
01285 {
01286   return (I2Cx_WriteBuffer(DevAddress, MemAd
```

```
dress, I2C_MEMADD_SIZE_16BIT, pBuffer, BufferSize)
);
01287 }
01288
01289 /**
01290   * @brief  Read data from I2C EEPROM driver
01291   * @param  DevAddress: Target device address

01292   * @param  MemAddress: Internal memory addr
ess
01293   * @param  pBuffer: Pointer to data buffer
01294   * @param  BufferSize: Amount of data to be
 read
01295   * @retval HAL status
01296   */
01297 HAL_StatusTypeDef EEPROM_I2C_IO_ReadData(uin
t16_t DevAddress, uint16_t MemAddress, uint8_t* pB
uffer, uint32_t BufferSize)
01298 {
01299   return (I2Cx_ReadBuffer(DevAddress, MemAdd
ress, I2C_MEMADD_SIZE_16BIT, pBuffer, BufferSize))
;
01300 }
01301
01302 /**
01303 * @brief  Checks if target device is ready f
or communication.
01304 * @note   This function is used with Memory
devices
01305 * @param  DevAddress: Target device address
01306 * @param  Trials: Number of trials
01307 * @retval HAL status
01308 */
01309 HAL_StatusTypeDef EEPROM_I2C_IO_IsDeviceReady
(uint16_t DevAddress, uint32_t Trials)
01310 {
01311   return (I2Cx_IsDeviceReady(DevAddress, Tri
```

```
als));
01312 }
01313
01314 /********************************** LINK I2C
TEMPERATURE SENSOR ***************************/
01315 /**
01316   * @brief  Initializes peripherals used by
the I2C Temperature Sensor driver.
01317   * @retval None
01318   */
01319 void TSENSOR_IO_Init(void)
01320 {
01321   I2Cx_Init();
01322 }
01323
01324 /**
01325   * @brief  Writes one byte to the TSENSOR.
01326   * @param  DevAddress: Target device address

01327   * @param  pBuffer: Pointer to data buffer
01328   * @param  WriteAddr: TSENSOR's internal ad
dress to write to.
01329   * @param  Length: Number of data to write
01330   * @retval None
01331   */
01332 void TSENSOR_IO_Write(uint16_t DevAddress, u
int8_t* pBuffer, uint8_t WriteAddr, uint16_t Lengt
h)
01333 {
01334   I2Cx_WriteBuffer(DevAddress, WriteAddr, I2
C_MEMADD_SIZE_8BIT, pBuffer, Length);
01335 }
01336
01337 /**
01338   * @brief  Reads one byte from the TSENSOR.
01339   * @param  DevAddress: Target device address
```

```
01340   * @param  pBuffer : pointer to the buffer
that receives the data read from the TSENSOR.
01341   * @param  ReadAddr : TSENSOR's internal ad
dress to read from.
01342   * @param  Length: Number of data to read
01343   * @retval None
01344   */
01345 void TSENSOR_IO_Read(uint16_t DevAddress, ui
nt8_t* pBuffer, uint8_t ReadAddr, uint16_t Length)
01346 {
01347   I2Cx_ReadBuffer(DevAddress, ReadAddr, I2C_
MEMADD_SIZE_8BIT, pBuffer, Length);
01348 }
01349
01350 /**
01351 * @brief  Checks if Temperature Sensor is re
ady for communication.
01352   * @param  DevAddress: Target device address

01353   * @param  Trials: Number of trials
01354 * @retval HAL status
01355 */
01356 uint16_t TSENSOR_IO_IsDeviceReady(uint16_t D
evAddress, uint32_t Trials)
01357 {
01358   return (I2Cx_IsDeviceReady(DevAddress, Tri
als));
01359 }
01360
01361 /****************************** LINK ACCELERO
 ******************************/
01362 /**
01363   * @brief  Configures ACCELEROMETER SPI int
erface.
01364   * @retval None
01365   */
01366 void ACCELERO_IO_Init(void)
```

```
01367 {
01368   /* Initialize the IO functionalities */
01369   BSP_IO_Init();
01370 }
01371
01372
01373 /**
01374   * @brief     Configures ACCELERO INT2 conf
ig.
01375                     EXTI0 is already used by
user button so INT1 is configured here
01376   * @retval   None
01377   */
01378 void ACCELERO_IO_ITConfig(void)
01379 {
01380   BSP_IO_ConfigPin(MEMS_ALL_PINS, IO_MODE_IT
_FALLING_EDGE);
01381 }
01382
01383 /**
01384   * @brief  Writes one byte to the ACCELEROM
ETER.
01385   * @param  pBuffer : pointer to the buffer
 containing the data to be written to the ACCELERO
METER.
01386   * @param  WriteAddr : ACCELEROMETER's inte
rnal address to write to.
01387   * @param  NumByteToWrite: Number of bytes
to write.
01388   * @retval None
01389   */
01390 void ACCELERO_IO_Write(uint8_t* pBuffer, uin
t8_t WriteAddr, uint16_t NumByteToWrite)
01391 {
01392   I2Cx_WriteBuffer(L1S302DL_I2C_ADDRESS, Wri
teAddr, I2C_MEMADD_SIZE_8BIT, pBuffer, NumByteToWr
ite);
```

```
01393 }
01394
01395 /**
01396   * @brief  Reads a block of data from the A
CCELEROMETER.
01397   * @param  pBuffer : pointer to the buffer
that receives the data read from the ACCELEROMETER.

01398   * @param  ReadAddr : ACCELEROMETER's inter
nal address to read from.
01399   * @param  NumByteToRead : number of bytes
to read from the ACCELEROMETER.
01400   * @retval None
01401   */
01402 void ACCELERO_IO_Read(uint8_t* pBuffer, uint
8_t ReadAddr, uint16_t NumByteToRead)
01403 {
01404   I2Cx_ReadBuffer(L1S302DL_I2C_ADDRESS, Read
Addr, I2C_MEMADD_SIZE_8BIT, pBuffer, NumByteToRead
);
01405 }
01406
01407 /********************************** LINK AUDI
O ***********************************/
01408
01409 /**
01410   * @brief  Initializes Audio low level.
01411   * @retval None
01412   */
01413 void AUDIO_IO_Init(void)
01414 {
01415   /* Initialize the IO functionalities */
01416   BSP_IO_Init();
01417
01418   BSP_IO_ConfigPin(AUDIO_RESET_PIN, IO_MODE_
OUTPUT);
01419
```

```
01420   /* Power Down the codec */
01421   BSP_IO_WritePin(AUDIO_RESET_PIN, GPIO_PIN_
RESET);
01422
01423   /* wait for a delay to insure registers er
asing */
01424   HAL_Delay(5);
01425
01426   /* Power on the codec */
01427   BSP_IO_WritePin(AUDIO_RESET_PIN, GPIO_PIN_
SET);
01428
01429   /* wait for a delay to insure registers er
asing */
01430   HAL_Delay(5);
01431
01432 }
01433
01434 /**
01435   * @brief  Writes a single data.
01436   * @param  Addr: I2C address
01437   * @param  Reg: Reg address
01438   * @param  Value: Data to be written
01439   * @retval None
01440   */
01441 void AUDIO_IO_Write (uint8_t Addr, uint8_t R
eg, uint8_t Value)
01442 {
01443   I2Cx_WriteData(Addr, Reg, Value);
01444 }
01445
01446 /**
01447   * @brief  Reads a single data.
01448   * @param  Addr: I2C address
01449   * @param  Reg: Reg address
01450   * @retval Data to be read
01451   */
```

```
01452 uint8_t AUDIO_IO_Read (uint8_t Addr, uint8_t
 Reg)
01453 {
01454   return I2Cx_ReadData(Addr, Reg);
01455 }
01456
01457 #endif /* HAL_I2C_MODULE_ENABLED */
01458
01459 /**
01460   * @}
01461   */
01462
01463 /**
01464   * @}
01465   */
01466
01467 /**
01468   * @}
01469   */
01470
01471 /**
01472   * @}
01473   */
01474
01475 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210C_EVAL BSP User Manual

STM3210C_EVAL IO Expander

STM3210C-EVAL

Modules

- Private Types Definitions
- Private_Defines
- Private_Macros
- Private_Variables
- Private_Function_Prototypes
- Exported_Functions
- Exported_Types
- Exported_Constants
- Exported_Macros

STM3210C_EVAL BSP User Manual

Exported_Types

STM3210C_EVAL LCD

Data Structures

struct	**LCD_DrawPropTypeDef**

STM3210C_EVAL BSP User Manual

stm3210c_eval_lcd.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210c_eval_lcd.h
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm3210c_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 __STM3210C_EVAL_LCD_H
00041 #define __STM3210C_EVAL_LCD_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------------------------------------------*/
00048 #include "stm3210c_eval.h"
00049 #include "../Components/ili9325/ili9325.h"
00050 #include "../Components/ili9320/ili9320.h"
00051 #include "../../../Utilities/Fonts/fonts.h"
00052 /** @addtogroup BSP
00053   * @{
00054   */
00055
00056 /** @addtogroup STM3210C_EVAL
00057   * @{
00058   */
00059
00060 /** @addtogroup STM3210C_EVAL_LCD
00061   * @{
00062   */
00063
00064
00065 /** @defgroup STM3210C_EVAL_LCD_Exported_Types Exported_Types
00066   * @{
00067   */
00068 typedef struct
```

```
00069 {
00070   uint32_t TextColor;
00071   uint32_t BackColor;
00072   sFONT    *pFont;
00073
00074 }LCD_DrawPropTypeDef;
00075 /**
00076   * @}
00077   */
00078
00079 /** @defgroup STM3210C_EVAL_LCD_Exported_Con
stants Exported_Constants
00080   * @{
00081   */
00082 /**
00083   * @brief  LCD status structure definition

00084   */
00085 #define LCD_OK          0x00
00086 #define LCD_ERROR       0x01
00087 #define LCD_TIMEOUT     0x02
00088
00089 typedef struct
00090 {
00091   int16_t X;
00092   int16_t Y;
00093
00094 }Point, * pPoint;
00095
00096 /**
00097   * @brief  Line mode structures definition
00098   */
00099 typedef enum
00100 {
00101   CENTER_MODE             = 0x01,    /*!< Ce
nter mode */
00102   RIGHT_MODE              = 0x02,    /*!< Ri
```

```
ght mode  */
00103   LEFT_MODE                = 0x03     /*!< Le
ft mode   */
00104
00105 }Line_ModeTypdef;
00106
00107 /**
00108   * @brief  LCD color
00109   */
00110 #define LCD_COLOR_BLUE          0x001F
00111 #define LCD_COLOR_GREEN         0x07E0
00112 #define LCD_COLOR_RED           0xF800
00113 #define LCD_COLOR_CYAN          0x07FF
00114 #define LCD_COLOR_MAGENTA       0xF81F
00115 #define LCD_COLOR_YELLOW        0xFFE0
00116 #define LCD_COLOR_LIGHTBLUE     0x841F
00117 #define LCD_COLOR_LIGHTGREEN    0x87F0
00118 #define LCD_COLOR_LIGHTRED      0xFC10
00119 #define LCD_COLOR_LIGHTCYAN     0x87FF
00120 #define LCD_COLOR_LIGHTMAGENTA  0xFC1F
00121 #define LCD_COLOR_LIGHTYELLOW   0xFFF0
00122 #define LCD_COLOR_DARKBLUE      0x0010
00123 #define LCD_COLOR_DARKGREEN     0x0400
00124 #define LCD_COLOR_DARKRED       0x8000
00125 #define LCD_COLOR_DARKCYAN      0x0410
00126 #define LCD_COLOR_DARKMAGENTA   0x8010
00127 #define LCD_COLOR_DARKYELLOW    0x8400
00128 #define LCD_COLOR_WHITE         0xFFFF
00129 #define LCD_COLOR_LIGHTGRAY     0xD69A
00130 #define LCD_COLOR_GRAY          0x8410
00131 #define LCD_COLOR_DARKGRAY      0x4208
00132 #define LCD_COLOR_BLACK         0x0000
00133 #define LCD_COLOR_BROWN         0xA145
00134 #define LCD_COLOR_ORANGE        0xFD20
00135
00136 /**
00137   * @brief LCD default font
```

```
00138   */
00139 #define LCD_DEFAULT_FONT         Font24
00140
00141 /**
00142   * @}
00143   */
00144
00145 /** @addtogroup STM3210C_EVAL_LCD_Exported_F
unctions
00146   * @{
00147   */
00148 uint8_t  BSP_LCD_Init(void);
00149 uint32_t BSP_LCD_GetXSize(void);
00150 uint32_t BSP_LCD_GetYSize(void);
00151
00152 uint16_t BSP_LCD_GetTextColor(void);
00153 uint16_t BSP_LCD_GetBackColor(void);
00154 void     BSP_LCD_SetTextColor(__IO uint16_t
Color);
00155 void     BSP_LCD_SetBackColor(__IO uint16_t
Color);
00156 void     BSP_LCD_SetFont(sFONT *fonts);
00157 sFONT    *BSP_LCD_GetFont(void);
00158
00159 void     BSP_LCD_Clear(uint16_t Color);
00160 void     BSP_LCD_ClearStringLine(uint16_t Li
ne);
00161 void     BSP_LCD_DisplayStringAtLine(uint16_
t Line, uint8_t *ptr);
00162 void     BSP_LCD_DisplayStringAt(uint16_t Xp
os, uint16_t Ypos, uint8_t *Text, Line_ModeTypdef
Mode);
00163 void     BSP_LCD_DisplayChar(uint16_t Xpos,
uint16_t Ypos, uint8_t Ascii);
00164
00165 uint16_t BSP_LCD_ReadPixel(uint16_t Xpos, ui
nt16_t Ypos);
```

```
00166 void     BSP_LCD_DrawHLine(uint16_t Xpos, ui
nt16_t Ypos, uint16_t Length);
00167 void     BSP_LCD_DrawVLine(uint16_t Xpos, ui
nt16_t Ypos, uint16_t Length);
00168 void     BSP_LCD_DrawLine(uint16_t x1, uint1
6_t y1, uint16_t x2, uint16_t y2);
00169 void     BSP_LCD_DrawRect(uint16_t Xpos, uin
t16_t Ypos, uint16_t Width, uint16_t Height);
00170 void     BSP_LCD_DrawCircle(uint16_t Xpos, u
int16_t Ypos, uint16_t Radius);
00171 void     BSP_LCD_DrawPolygon(pPoint Points,
uint16_t PointCount);
00172 void     BSP_LCD_DrawEllipse(int Xpos, int Y
pos, int XRadius, int YRadius);
00173 void     BSP_LCD_DrawBitmap(uint16_t Xpos, u
int16_t Ypos, uint8_t *pbmp);
00174 void     BSP_LCD_DrawRGBImage(uint16_t Xpos,
 uint16_t Ypos, uint16_t Xsize, uint16_t Ysize, ui
nt8_t *pbmp);
00175 void     BSP_LCD_FillRect(uint16_t Xpos, uin
t16_t Ypos, uint16_t Width, uint16_t Height);
00176 void     BSP_LCD_FillCircle(uint16_t Xpos, u
int16_t Ypos, uint16_t Radius);
00177 void     BSP_LCD_FillEllipse(int Xpos, int Y
pos, int XRadius, int YRadius);
00178
00179 void     BSP_LCD_DisplayOff(void);
00180 void     BSP_LCD_DisplayOn(void);
00181
00182 /**
00183   * @}
00184   */
00185
00186 #ifdef __cplusplus
00187 }
00188 #endif
00189
```

```
00190 #endif /* __STM3210C_EVAL_LCD_H */
00191 /**
00192   * @}
00193   */
00194
00195 /**
00196   * @}
00197   */
00198
00199 /**
00200   * @}
00201   */
00202
00203 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210C_EVAL BSP User Manual

stm3210c_eval_lcd.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210c_eval_lcd.c
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file includes the driver for Liquid Crystal Display (LCD) module
00008   *          mounted on STM3210C-EVAL evaluation board.
00009   @verbatim
00010   ================================================================================
00011                          ##### How to use this driver #####
00012   ================================================================================
00013   [..]
00014    (#) This driver is used to drive indirectly an LCD TFT.
00015
```

```
00016    (#) This driver supports the AM-240320L8T
NQW00H (LCD_ILI9320) and AM240320D5TOQW01H
00017        (LCD_ILI9325) LCD mounted on MB785 da
ughter board
00018
00019    (#) The ILI9320 and ILI9325 components dr
iver MUST be included with this driver.
00020
00021    (#) Initialization steps:
00022        (++) Initialize the LCD using the BSP
_LCD_Init() function.
00023
00024    (#) Display on LCD
00025        (++) Clear the hole LCD using yhe BSP
_LCD_Clear() function or only one specified
00026             string line using the BSP_LCD_Cl
earStringLine() function.
00027        (++) Display a character on the speci
fied line and column using the BSP_LCD_DisplayChar
()
00028             function or a complete string li
ne using the BSP_LCD_DisplayStringAtLine() functio
n.
00029        (++) Display a string line on the spe
cified position (x,y in pixel) and align mode
00030             using the BSP_LCD_DisplayStringA
tLine() function.
00031        (++) Draw and fill a basic shapes (do
t, line, rectangle, circle, ellipse, .. bitmap, ra
w picture)
00032             on LCD using a set of functions.

00033   @endverbatim
00034   ******************************************
**********************************
00035   * @attention
00036   *
```

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

```
 OR BUSINESS INTERRUPTION) HOWEVER
00057   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00058   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00059   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00060   *
00061   ******************************************
**********************************
00062   */
00063
00064 /* Includes ---------------------------------
-----------------------------------*/
00065 #include "stm3210c_eval_lcd.h"
00066 #include "../../../Utilities/Fonts/fonts.h"
00067 #include "../../../Utilities/Fonts/font24.c"
00068 #include "../../../Utilities/Fonts/font20.c"
00069 #include "../../../Utilities/Fonts/font16.c"
00070 #include "../../../Utilities/Fonts/font12.c"
00071 #include "../../../Utilities/Fonts/font8.c"
00072
00073 /** @addtogroup BSP
00074   * @{
00075   */
00076
00077 /** @addtogroup STM3210C_EVAL
00078   * @{
00079   */
00080
00081 /** @defgroup STM3210C_EVAL_LCD STM3210C_EVA
L LCD
00082   * @{
00083   */
00084
00085
00086 /** @defgroup STM3210C_EVAL_LCD_Private_Defi
```

```
nes Private Defines
00087   * @{
00088   */
00089 #define POLY_X(Z)              ((int32_t)((P
oints + (Z))->X))
00090 #define POLY_Y(Z)              ((int32_t)((P
oints + (Z))->Y))
00091
00092 #define MAX_HEIGHT_FONT         17
00093 #define MAX_WIDTH_FONT          24
00094 #define OFFSET_BITMAP           54
00095 /**
00096   * @}
00097   */
00098
00099 /** @defgroup STM3210C_EVAL_LCD_Private_Macr
os Private Macros
00100   * @{
00101   */
00102 #define ABS(X)  ((X) > 0 ? (X) : -(X))
00103
00104 /**
00105   * @}
00106   */
00107
00108 /** @defgroup STM3210C_EVAL_LCD_Private_Vari
ables Private Variables
00109   * @{
00110   */
00111 LCD_DrawPropTypeDef DrawProp;
00112
00113 static LCD_DrvTypeDef  *lcd_drv;
00114
00115 /* Max size of bitmap will based on a font24
 (17x24) */
00116 static uint8_t bitmap[MAX_HEIGHT_FONT*MAX_WI
DTH_FONT*2+OFFSET_BITMAP] = {0};
```

```
00117
00118 /**
00119   * @}
00120   */
00121
00122 /** @defgroup STM3210C_EVAL_LCD_Private_Func
tions Private Functions
00123   * @{
00124   */
00125 static void LCD_DrawPixel(uint16_t Xpos, uin
t16_t Ypos, uint16_t RGBCode);
00126 static void LCD_DrawChar(uint16_t Xpos, uint
16_t Ypos, const uint8_t *c);
00127 static void LCD_SetDisplayWindow(uint16_t Xp
os, uint16_t Ypos, uint16_t Width, uint16_t Height
);
00128 /**
00129   * @}
00130   */
00131
00132
00133 /** @defgroup STM3210C_EVAL_LCD_Exported_Fun
ctions Exported Functions
00134   * @{
00135   */
00136
00137 /**
00138   * @brief  Initializes the LCD.
00139   * @retval LCD state
00140   */
00141 uint8_t BSP_LCD_Init(void)
00142 {
00143   uint8_t ret = LCD_ERROR;
00144
00145   /* Default value for draw propriety */
00146   DrawProp.BackColor = 0xFFFF;
00147   DrawProp.pFont     = &Font24;
```

```
00148   DrawProp.TextColor = 0x0000;
00149
00150   if(ili9320_drv.ReadID() == ILI9320_ID)
00151   {
00152     lcd_drv = &ili9320_drv;
00153
00154     /* LCD Init */
00155     lcd_drv->Init();
00156
00157     /* Initialize the font */
00158     BSP_LCD_SetFont(&LCD_DEFAULT_FONT);
00159
00160     ret = LCD_OK;
00161   }
00162   else if(ili9325_drv.ReadID() == ILI9325_ID
)
00163   {
00164     lcd_drv = &ili9325_drv;
00165
00166     /* LCD Init */
00167     lcd_drv->Init();
00168
00169     /* Initialize the font */
00170     BSP_LCD_SetFont(&LCD_DEFAULT_FONT);
00171
00172     ret = LCD_OK;
00173   }
00174
00175   return ret;
00176 }
00177
00178 /**
00179   * @brief  Gets the LCD X size.
00180   * @retval Used LCD X size
00181   */
00182 uint32_t BSP_LCD_GetXSize(void)
00183 {
```

```
00184   return(lcd_drv->GetLcdPixelWidth());
00185 }
00186
00187 /**
00188   * @brief  Gets the LCD Y size.
00189   * @retval Used LCD Y size
00190   */
00191 uint32_t BSP_LCD_GetYSize(void)
00192 {
00193   return(lcd_drv->GetLcdPixelHeight());
00194 }
00195
00196 /**
00197   * @brief  Gets the LCD text color.
00198   * @retval Used text color.
00199   */
00200 uint16_t BSP_LCD_GetTextColor(void)
00201 {
00202   return DrawProp.TextColor;
00203 }
00204
00205 /**
00206   * @brief  Gets the LCD background color.
00207   * @retval Used background color
00208   */
00209 uint16_t BSP_LCD_GetBackColor(void)
00210 {
00211   return DrawProp.BackColor;
00212 }
00213
00214 /**
00215   * @brief  Sets the LCD text color.
00216   * @param  Color: Text color code RGB(5-6-5)

00217   * @retval None
00218   */
00219 void BSP_LCD_SetTextColor(uint16_t Color)
```

```
00220 {
00221   DrawProp.TextColor = Color;
00222 }
00223
00224 /**
00225   * @brief  Sets the LCD background color.
00226   * @param  Color: Background color code RGB
(5-6-5)
00227   * @retval None
00228   */
00229 void BSP_LCD_SetBackColor(uint16_t Color)
00230 {
00231   DrawProp.BackColor = Color;
00232 }
00233
00234 /**
00235   * @brief  Sets the LCD text font.
00236   * @param  pFonts: Font to be used
00237   * @retval None
00238   */
00239 void BSP_LCD_SetFont(sFONT *pFonts)
00240 {
00241   DrawProp.pFont = pFonts;
00242 }
00243
00244 /**
00245   * @brief  Gets the LCD text font.
00246   * @retval Used font
00247   */
00248 sFONT *BSP_LCD_GetFont(void)
00249 {
00250   return DrawProp.pFont;
00251 }
00252
00253 /**
00254   * @brief  Clears the hole LCD.
00255   * @param  Color: Color of the background
```

```
00256   * @retval None
00257   */
00258 void BSP_LCD_Clear(uint16_t Color)
00259 {
00260   uint32_t counter = 0;
00261
00262   uint32_t color_backup = DrawProp.TextColor
;
00263   DrawProp.TextColor = Color;
00264
00265   for(counter = 0; counter < BSP_LCD_GetYSize
(); counter++)
00266   {
00267     BSP_LCD_DrawHLine(0, counter, BSP_LCD_Ge
tXSize());
00268   }
00269
00270   DrawProp.TextColor = color_backup;
00271   BSP_LCD_SetTextColor(DrawProp.TextColor);
00272 }
00273
00274 /**
00275   * @brief  Clears the selected line.
00276   * @param  Line: Line to be cleared
00277   *          This parameter can be one of th
e following values:
00278   *            @arg  0..9: if the Current fo
nts is Font16x24
00279   *            @arg  0..19: if the Current f
onts is Font12x12 or Font8x12
00280   *            @arg  0..29: if the Current f
onts is Font8x8
00281   * @retval None
00282   */
00283 void BSP_LCD_ClearStringLine(uint16_t Line)
00284 {
00285   uint32_t colorbackup = DrawProp.TextColor;
```

```
00286   DrawProp.TextColor = DrawProp.BackColor;;
00287
00288   /* Draw a rectangle with background color
*/
00289   BSP_LCD_FillRect(0, (Line * DrawProp.pFont
->Height), BSP_LCD_GetXSize(), DrawProp.pFont->Hei
ght);
00290
00291   DrawProp.TextColor = colorbackup;
00292   BSP_LCD_SetTextColor(DrawProp.TextColor);
00293 }
00294
00295 /**
00296   * @brief  Displays one character.
00297   * @param  Xpos: Start column address
00298   * @param  Ypos: Line where to display the
character shape.
00299   * @param  Ascii: Character ascii code
00300   *           This parameter must be a numbe
r between Min_Data = 0x20 and Max_Data = 0x7E
00301   * @retval None
00302   */
00303 void BSP_LCD_DisplayChar(uint16_t Xpos, uint
16_t Ypos, uint8_t Ascii)
00304 {
00305   LCD_DrawChar(Xpos, Ypos, &DrawProp.pFont->
table[(Ascii-' ') *\
00306     DrawProp.pFont->Height * ((DrawProp.pFont
->Width + 7) / 8)]);
00307 }
00308
00309 /**
00310   * @brief  Displays characters on the LCD.
00311   * @param  Xpos: X position (in pixel)
00312   * @param  Ypos: Y position (in pixel)
00313   * @param  pText: Pointer to string to disp
```

```
lay on LCD
00314   * @param  Mode: Display mode
00315   *           This parameter can be one of th
e following values:
00316   *             @arg  CENTER_MODE
00317   *             @arg  RIGHT_MODE
00318   *             @arg  LEFT_MODE
00319   * @retval None
00320   */
00321 void BSP_LCD_DisplayStringAt(uint16_t Xpos,
uint16_t Ypos, uint8_t *pText, Line_ModeTypdef Mod
e)
00322 {
00323   uint16_t refcolumn = 1, counter = 0;
00324   uint32_t size = 0, xsize = 0;
00325   uint8_t  *ptr = pText;
00326
00327   /* Get the text size */
00328   while (*ptr++) size ++ ;
00329
00330   /* Characters number per line */
00331   xsize = (BSP_LCD_GetXSize()/DrawProp.pFont
->Width);
00332
00333   switch (Mode)
00334   {
00335   case CENTER_MODE:
00336     {
00337       refcolumn = Xpos + ((xsize - size)* Dr
awProp.pFont->Width) / 2;
00338       break;
00339     }
00340   case LEFT_MODE:
00341     {
00342       refcolumn = Xpos;
00343       break;
00344     }
```

```
00345   case RIGHT_MODE:
00346     {
00347       refcolumn =  - Xpos + ((xsize - size)*
DrawProp.pFont->Width);
00348       break;
00349     }
00350   default:
00351     {
00352       refcolumn = Xpos;
00353       break;
00354     }
00355   }
00356
00357   /* Send the string character by character
on lCD */
00358   while ((*pText != 0) & (((BSP_LCD_GetXSize
() - (counter*DrawProp.pFont->Width)) & 0xFFFF) >=
 DrawProp.pFont->Width))
00359   {
00360     /* Display one character on LCD */
00361     BSP_LCD_DisplayChar(refcolumn, Ypos, *pT
ext);
00362     /* Decrement the column position by 16 */

00363     refcolumn += DrawProp.pFont->Width;
00364     /* Point on the next character */
00365     pText++;
00366     counter++;
00367   }
00368 }
00369
00370 /**
00371   * @brief  Displays a character on the LCD.
00372   * @param  Line: Line where to display the
character shape
00373   *           This parameter can be one of th
e following values:
```

```
00374   *             @arg  0..9: if the Current fo
nts is Font16x24
00375   *             @arg  0..19: if the Current f
onts is Font12x12 or Font8x12
00376   *             @arg  0..29: if the Current f
onts is Font8x8
00377   * @param  pText: Pointer to string to disp
lay on LCD
00378   * @retval None
00379   */
00380 void BSP_LCD_DisplayStringAtLine(uint16_t Li
ne, uint8_t *pText)
00381 {
00382   BSP_LCD_DisplayStringAt(0, LINE(Line),pTex
t, LEFT_MODE);
00383 }
00384
00385 /**
00386   * @brief  Reads an LCD pixel.
00387   * @param  Xpos: X position
00388   * @param  Ypos: Y position
00389   * @retval RGB pixel color
00390   */
00391 uint16_t BSP_LCD_ReadPixel(uint16_t Xpos, ui
nt16_t Ypos)
00392 {
00393   uint16_t ret = 0;
00394
00395   if(lcd_drv->ReadPixel != NULL)
00396   {
00397     ret = lcd_drv->ReadPixel(Xpos, Ypos);
00398   }
00399
00400   return ret;
00401 }
00402
00403 /**
```

```
00404   * @brief  Draws an horizontal line.
00405   * @param  Xpos: X position
00406   * @param  Ypos: Y position
00407   * @param  Length: Line length
00408   * @retval None
00409   */
00410 void BSP_LCD_DrawHLine(uint16_t Xpos, uint16
_t Ypos, uint16_t Length)
00411 {
00412   uint32_t index = 0;
00413
00414   if(lcd_drv->DrawHLine != NULL)
00415   {
00416     lcd_drv->DrawHLine(DrawProp.TextColor, X
pos, Ypos, Length);
00417   }
00418   else
00419   {
00420     for(index = 0; index < Length; index++)
00421     {
00422       LCD_DrawPixel((Xpos + index), Ypos, Dr
awProp.TextColor);
00423     }
00424   }
00425 }
00426
00427 /**
00428   * @brief  Draws a vertical line.
00429   * @param  Xpos: X position
00430   * @param  Ypos: Y position
00431   * @param  Length: Line length
00432   * @retval None
00433   */
00434 void BSP_LCD_DrawVLine(uint16_t Xpos, uint16
_t Ypos, uint16_t Length)
00435 {
00436   uint32_t index = 0;
```

```
00437
00438   if(lcd_drv->DrawVLine != NULL)
00439   {
00440     lcd_drv->DrawVLine(DrawProp.TextColor, X
pos, Ypos, Length);
00441   }
00442   else
00443   {
00444     for(index = 0; index < Length; index++)
00445     {
00446       LCD_DrawPixel(Xpos, Ypos + index, Draw
Prop.TextColor);
00447     }
00448   }
00449 }
00450
00451 /**
00452   * @brief  Draws an uni-line (between two p
oints).
00453   * @param  x1: Point 1 X position
00454   * @param  y1: Point 1 Y position
00455   * @param  x2: Point 2 X position
00456   * @param  y2: Point 2 Y position
00457   * @retval None
00458   */
00459 void BSP_LCD_DrawLine(uint16_t x1, uint16_t
y1, uint16_t x2, uint16_t y2)
00460 {
00461   int16_t deltax = 0, deltay = 0, x = 0, y =
 0, xinc1 = 0, xinc2 = 0,
00462   yinc1 = 0, yinc2 = 0, den = 0, num = 0, nu
madd = 0, numpixels = 0,
00463   curpixel = 0;
00464
00465   deltax = ABS(x2 - x1);        /* The diffe
rence between the x's */
00466   deltay = ABS(y2 - y1);        /* The diffe
```

```
rence between the y's */
00467   x = x1;                         /* Start x o
ff at the first pixel */
00468   y = y1;                         /* Start y o
ff at the first pixel */
00469
00470   if (x2 >= x1)                   /* The x-val
ues are increasing */
00471   {
00472     xinc1 = 1;
00473     xinc2 = 1;
00474   }
00475   else                            /* The x-val
ues are decreasing */
00476   {
00477     xinc1 = -1;
00478     xinc2 = -1;
00479   }
00480
00481   if (y2 >= y1)                   /* The y-val
ues are increasing */
00482   {
00483     yinc1 = 1;
00484     yinc2 = 1;
00485   }
00486   else                            /* The y-val
ues are decreasing */
00487   {
00488     yinc1 = -1;
00489     yinc2 = -1;
00490   }
00491
00492   if (deltax >= deltay)           /* There is
at least one x-value for every y-value */
00493   {
00494     xinc1 = 0;                    /* Don't cha
nge the x when numerator >= denominator */
```

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

```
00519       y += yinc1;
  /* Change the y as appropriate */
00520     }
00521     x += xinc2;
  /* Change the x as appropriate */
00522     y += yinc2;
  /* Change the y as appropriate */
00523   }
00524 }
00525
00526 /**
00527   * @brief  Draws a rectangle.
00528   * @param  Xpos: X position
00529   * @param  Ypos: Y position
00530   * @param  Width: Rectangle width
00531   * @param  Height: Rectangle height
00532   * @retval None
00533   */
00534 void BSP_LCD_DrawRect(uint16_t Xpos, uint16_
t Ypos, uint16_t Width, uint16_t Height)
00535 {
00536   /* Draw horizontal lines */
00537   BSP_LCD_DrawHLine(Xpos, Ypos, Width);
00538   BSP_LCD_DrawHLine(Xpos, (Ypos+ Height), Wi
dth);
00539
00540   /* Draw vertical lines */
00541   BSP_LCD_DrawVLine(Xpos, Ypos, Height);
00542   BSP_LCD_DrawVLine((Xpos + Width), Ypos, He
ight);
00543 }
00544
00545 /**
00546   * @brief  Draws a circle.
00547   * @param  Xpos: X position
00548   * @param  Ypos: Y position
00549   * @param  Radius: Circle radius
```

```
00550   * @retval None
00551   */
00552 void BSP_LCD_DrawCircle(uint16_t Xpos, uint1
6_t Ypos, uint16_t Radius)
00553 {
00554   int32_t  D;        /* Decision Variable */
00555   uint32_t  CurX;    /* Current X Value */
00556   uint32_t  CurY;    /* Current Y Value */
00557
00558   D = 3 - (Radius << 1);
00559   CurX = 0;
00560   CurY = Radius;
00561
00562   while (CurX <= CurY)
00563   {
00564     LCD_DrawPixel((Xpos + CurX), (Ypos - Cur
Y), DrawProp.TextColor);
00565
00566     LCD_DrawPixel((Xpos - CurX), (Ypos - Cur
Y), DrawProp.TextColor);
00567
00568     LCD_DrawPixel((Xpos + CurY), (Ypos - Cur
X), DrawProp.TextColor);
00569
00570     LCD_DrawPixel((Xpos - CurY), (Ypos - Cur
X), DrawProp.TextColor);
00571
00572     LCD_DrawPixel((Xpos + CurX), (Ypos + Cur
Y), DrawProp.TextColor);
00573
00574     LCD_DrawPixel((Xpos - CurX), (Ypos + Cur
Y), DrawProp.TextColor);
00575
00576     LCD_DrawPixel((Xpos + CurY), (Ypos + Cur
X), DrawProp.TextColor);
00577
00578     LCD_DrawPixel((Xpos - CurY), (Ypos + Cur
```

```
X), DrawProp.TextColor);
00579
00580     /* Initialize the font */
00581     BSP_LCD_SetFont(&LCD_DEFAULT_FONT);
00582
00583     if (D < 0)
00584     {
00585       D += (CurX << 2) + 6;
00586     }
00587     else
00588     {
00589       D += ((CurX - CurY) << 2) + 10;
00590       CurY--;
00591     }
00592     CurX++;
00593   }
00594 }
00595
00596 /**
00597   * @brief  Draws an poly-line (between many
 points).
00598   * @param  Points: Pointer to the points ar
ray
00599   * @param  PointCount: Number of points
00600   * @retval None
00601   */
00602 void BSP_LCD_DrawPolygon(pPoint Points, uint
16_t PointCount)
00603 {
00604   int16_t X = 0, Y = 0;
00605
00606   if(PointCount < 2)
00607   {
00608     return;
00609   }
00610
00611   BSP_LCD_DrawLine(Points->X, Points->Y, (Po
```

```
ints+PointCount-1)->X, (Points+PointCount-1)->Y);
00612
00613   while(--PointCount)
00614   {
00615     X = Points->X;
00616     Y = Points->Y;
00617     Points++;
00618     BSP_LCD_DrawLine(X, Y, Points->X, Points
->Y);
00619   }
00620
00621 }
00622
00623 /**
00624   * @brief  Draws an ellipse on LCD.
00625   * @param  Xpos: X position
00626   * @param  Ypos: Y position
00627   * @param  XRadius: Ellipse X radius
00628   * @param  YRadius: Ellipse Y radius
00629   * @retval None
00630   */
00631 void BSP_LCD_DrawEllipse(int Xpos, int Ypos,
int XRadius, int YRadius)
00632 {
00633   int x = 0, y = -YRadius, err = 2-2*XRadius
, e2;
00634   float K = 0, rad1 = 0, rad2 = 0;
00635
00636   rad1 = XRadius;
00637   rad2 = YRadius;
00638
00639   K = (float)(rad2/rad1);
00640
00641   do {
00642     LCD_DrawPixel((Xpos-(uint16_t)(x/K)), (Y
pos+y), DrawProp.TextColor);
00643     LCD_DrawPixel((Xpos+(uint16_t)(x/K)), (Y
```

```
pos+y), DrawProp.TextColor);
00644     LCD_DrawPixel((Xpos+(uint16_t)(x/K)), (Y
pos-y), DrawProp.TextColor);
00645     LCD_DrawPixel((Xpos-(uint16_t)(x/K)), (Y
pos-y), DrawProp.TextColor);
00646
00647     e2 = err;
00648     if (e2 <= x) {
00649       err += ++x*2+1;
00650       if (-y == x && e2 <= y) e2 = 0;
00651     }
00652     if (e2 > y) err += ++y*2+1;
00653   }
00654   while (y <= 0);
00655 }
00656
00657 /**
00658   * @brief  Draws a bitmap picture (16 bpp).
00659   * @param  Xpos: Bmp X position in the LCD
00660   * @param  Ypos: Bmp Y position in the LCD
00661   * @param  pbmp: Pointer to Bmp picture add
ress.
00662   * @retval None
00663   */
00664 void BSP_LCD_DrawBitmap(uint16_t Xpos, uint1
6_t Ypos, uint8_t *pbmp)
00665 {
00666   uint32_t height = 0;
00667   uint32_t width  = 0;
00668
00669
00670   /* Read bitmap width */
00671   width = *(uint16_t *) (pbmp + 18);
00672   width |= (*(uint16_t *) (pbmp + 20)) << 16
;
00673
00674   /* Read bitmap height */
```

```
00675   height = *(uint16_t *) (pbmp + 22);
00676   height |= (*(uint16_t *) (pbmp + 24)) << 1
6;
00677
00678   LCD_SetDisplayWindow(Xpos, Ypos, width, he
ight);
00679
00680   if(lcd_drv->DrawBitmap != NULL)
00681   {
00682     lcd_drv->DrawBitmap(Xpos, Ypos, pbmp);
00683   }
00684   LCD_SetDisplayWindow(0, 0, BSP_LCD_GetXSize
(), BSP_LCD_GetYSize());
00685 }
00686
00687 /**
00688   * @brief  Draws RGB Image (16 bpp).
00689   * @param  Xpos:  X position in the LCD
00690   * @param  Ypos:  Y position in the LCD
00691   * @param  Xsize: X size in the LCD
00692   * @param  Ysize: Y size in the LCD
00693   * @param  pdata: Pointer to the RGB Image
address.
00694   * @retval None
00695   */
00696 void BSP_LCD_DrawRGBImage(uint16_t Xpos, uin
t16_t Ypos, uint16_t Xsize, uint16_t Ysize, uint8_
t *pdata)
00697 {
00698
00699   LCD_SetDisplayWindow(Xpos, Ypos, Xsize, Ys
ize);
00700
00701   if(lcd_drv->DrawRGBImage != NULL)
00702   {
00703     lcd_drv->DrawRGBImage(Xpos, Ypos, Xsize,
 Ysize, pdata);
```

```
00704   }
00705   LCD_SetDisplayWindow(0, 0, BSP_LCD_GetXSize
(), BSP_LCD_GetYSize());
00706 }
00707
00708 /**
00709   * @brief  Draws a full rectangle.
00710   * @param  Xpos: X position
00711   * @param  Ypos: Y position
00712   * @param  Width: Rectangle width
00713   * @param  Height: Rectangle height
00714   * @retval None
00715   */
00716 void BSP_LCD_FillRect(uint16_t Xpos, uint16_
t Ypos, uint16_t Width, uint16_t Height)
00717 {
00718   BSP_LCD_SetTextColor(DrawProp.TextColor);
00719   do
00720   {
00721     BSP_LCD_DrawHLine(Xpos, Ypos++, Width);

00722   }
00723   while(Height--);
00724 }
00725
00726 /**
00727   * @brief  Draws a full circle.
00728   * @param  Xpos: X position
00729   * @param  Ypos: Y position
00730   * @param  Radius: Circle radius
00731   * @retval None
00732   */
00733 void BSP_LCD_FillCircle(uint16_t Xpos, uint1
6_t Ypos, uint16_t Radius)
00734 {
00735   int32_t  D;        /* Decision Variable */
```

```
00736   uint32_t  CurX;    /* Current X Value */
00737   uint32_t  CurY;    /* Current Y Value */
00738
00739   D = 3 - (Radius << 1);
00740
00741   CurX = 0;
00742   CurY = Radius;
00743
00744   BSP_LCD_SetTextColor(DrawProp.TextColor);
00745
00746   while (CurX <= CurY)
00747   {
00748     if(CurY > 0)
00749     {
00750       BSP_LCD_DrawHLine(Xpos - CurY, Ypos +
CurX, 2*CurY);
00751       BSP_LCD_DrawHLine(Xpos - CurY, Ypos -
CurX, 2*CurY);
00752     }
00753
00754     if(CurX > 0)
00755     {
00756       BSP_LCD_DrawHLine(Xpos - CurX, Ypos -
CurY, 2*CurX);
00757       BSP_LCD_DrawHLine(Xpos - CurX, Ypos +
CurY, 2*CurX);
00758     }
00759     if (D < 0)
00760     {
00761       D += (CurX << 2) + 6;
00762     }
00763     else
00764     {
00765       D += ((CurX - CurY) << 2) + 10;
00766       CurY--;
00767     }
00768     CurX++;
```

```
00769   }
00770
00771   BSP_LCD_SetTextColor(DrawProp.TextColor);
00772   BSP_LCD_DrawCircle(Xpos, Ypos, Radius);
00773 }
00774
00775 /**
00776   * @brief  Draws a full ellipse.
00777   * @param  Xpos: X position
00778   * @param  Ypos: Y position
00779   * @param  XRadius: Ellipse X radius
00780   * @param  YRadius: Ellipse Y radius
00781   * @retval None
00782   */
00783 void BSP_LCD_FillEllipse(int Xpos, int Ypos,
int XRadius, int YRadius)
00784 {
00785   int x = 0, y = -YRadius, err = 2-2*XRadius
, e2;
00786   float K = 0, rad1 = 0, rad2 = 0;
00787
00788   rad1 = XRadius;
00789   rad2 = YRadius;
00790
00791   K = (float)(rad2/rad1);
00792
00793   do
00794   {
00795     BSP_LCD_DrawHLine((Xpos-(uint16_t)(x/K))
, (Ypos+y), (2*(uint16_t)(x/K) + 1));
00796     BSP_LCD_DrawHLine((Xpos-(uint16_t)(x/K))
, (Ypos-y), (2*(uint16_t)(x/K) + 1));
00797
00798     e2 = err;
00799     if (e2 <= x)
00800     {
00801       err += ++x*2+1;
```

```
00802       if (-y == x && e2 <= y) e2 = 0;
00803     }
00804     if (e2 > y) err += ++y*2+1;
00805   }
00806   while (y <= 0);
00807 }
00808
00809 /**
00810   * @brief  Enables the display.
00811   * @retval None
00812   */
00813 void BSP_LCD_DisplayOn(void)
00814 {
00815   lcd_drv->DisplayOn();
00816 }
00817
00818 /**
00819   * @brief  Disables the display.
00820   * @retval None
00821   */
00822 void BSP_LCD_DisplayOff(void)
00823 {
00824   lcd_drv->DisplayOff();
00825 }
00826
00827 /**
00828   * @}
00829   */
00830
00831 /** @addtogroup STM3210C_EVAL_LCD_Private_Fu
nctions
00832   * @{
00833   */
00834
00835 /*************************************************
*********************************
00836                                   Static Function
```

```
00837 ********************************************
**********************************/
00838 /**
00839   * @brief  Draws a pixel on LCD.
00840   * @param  Xpos: X position
00841   * @param  Ypos: Y position
00842   * @param  RGBCode: Pixel color in RGB mode
 (5-6-5)
00843   * @retval None
00844   */
00845 static void LCD_DrawPixel(uint16_t Xpos, uin
t16_t Ypos, uint16_t RGBCode)
00846 {
00847   if(lcd_drv->WritePixel != NULL)
00848   {
00849     lcd_drv->WritePixel(Xpos, Ypos, RGBCode)
;
00850   }
00851 }
00852
00853 /**
00854   * @brief  Draws a character on LCD.
00855   * @param  Xpos: Line where to display the
character shape
00856   * @param  Ypos: Start column address
00857   * @param  pChar: Pointer to the character
data
00858   * @retval None
00859   */
00860 static void LCD_DrawChar(uint16_t Xpos, uint
16_t Ypos, const uint8_t *pChar)
00861 {
00862   uint32_t counterh = 0, counterw = 0, index
 = 0;
00863   uint16_t height = 0, width = 0;
00864   uint8_t offset = 0;
00865   uint8_t *pchar = NULL;
```

```
00866   uint32_t line = 0;
00867
00868
00869   height = DrawProp.pFont->Height;
00870   width  = DrawProp.pFont->Width;
00871
00872   /* Fill bitmap header*/
00873   *(uint16_t *) (bitmap + 2) = (uint16_t)(he
ight*width*2+OFFSET_BITMAP);
00874   *(uint16_t *) (bitmap + 4) = (uint16_t)((h
eight*width*2+OFFSET_BITMAP)>>16);
00875   *(uint16_t *) (bitmap + 10) = OFFSET_BITMAP
;
00876   *(uint16_t *) (bitmap + 18) = (uint16_t)(w
idth);
00877   *(uint16_t *) (bitmap + 20) = (uint16_t)((
width)>>16);
00878   *(uint16_t *) (bitmap + 22) = (uint16_t)(h
eight);
00879   *(uint16_t *) (bitmap + 24) = (uint16_t)((
height)>>16);
00880
00881   offset =  8 *((width + 7)/8) -  width ;
00882
00883   for(counterh = 0; counterh < height; count
erh++)
00884   {
00885     pchar = ((uint8_t *)pChar + (width + 7)/
8 * counterh);
00886
00887     if(((width + 7)/8) == 3)
00888     {
00889       line =  (pchar[0]<< 16) | (pchar[1]<<
8) | pchar[2];
00890     }
00891
00892     if(((width + 7)/8) == 2)
```

```
00893     {
00894       line =  (pchar[0]<< 8) | pchar[1];
00895     }
00896
00897     if(((width + 7)/8) == 1)
00898     {
00899       line =  pchar[0];
00900     }
00901
00902     for (counterw = 0; counterw < width; cou
nterw++)
00903     {
00904       /* Image in the bitmap is written from
 the bottom to the top */
00905       /* Need to invert image in the bitmap
*/
00906       index = (((height-counterh-1)*width)+(
counterw))*2+OFFSET_BITMAP;
00907       if(line & (1 << (width- counterw + off
set- 1)))
00908       {
00909         bitmap[index] = (uint8_t)DrawProp.Te
xtColor;
00910         bitmap[index+1] = (uint8_t)(DrawProp.
TextColor >> 8);
00911       }
00912       else
00913       {
00914         bitmap[index] = (uint8_t)DrawProp.Ba
ckColor;
00915         bitmap[index+1] = (uint8_t)(DrawProp.
BackColor >> 8);
00916       }
00917     }
00918   }
00919
00920   BSP_LCD_DrawBitmap(Xpos, Ypos, bitmap);
```

```
00921 }
00922
00923 /**
00924   * @brief  Sets display window.
00925   * @param  Xpos: LCD X position
00926   * @param  Ypos: LCD Y position
00927   * @param  Width: LCD window width
00928   * @param  Height: LCD window height
00929   * @retval None
00930   */
00931 static void LCD_SetDisplayWindow(uint16_t Xp
os, uint16_t Ypos, uint16_t Width, uint16_t Height
)
00932 {
00933   if(lcd_drv->SetDisplayWindow != NULL)
00934   {
00935     lcd_drv->SetDisplayWindow(Xpos, Ypos, Wi
dth, Height);
00936   }
00937 }
00938
00939 /**
00940   * @}
00941   */
00942
00943 /**
00944   * @}
00945   */
00946
00947 /**
00948   * @}
00949   */
00950
00951 /**
00952   * @}
00953   */
00954
```

```
00955 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
00956
```

STM3210C_EVAL BSP User Manual

Data Structures | Defines | Typedefs | Enumerations

Exported_Constants

STM3210C_EVAL LCD

Data Structures

struct	**Point**

Defines

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

LCD default font.

Typedefs

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

Enumerations

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

Define Documentation

#define LCD_COLOR_BLACK 0x0000

Definition at line 132 of file stm3210c_eval_lcd.h.

#define LCD_COLOR_BLUE 0x001F

LCD color.

Definition at line 110 of file stm3210c_eval_lcd.h.

#define LCD_COLOR_BROWN 0xA145

Definition at line 133 of file stm3210c_eval_lcd.h.

#define LCD_COLOR_CYAN 0x07FF

Definition at line 113 of file stm3210c_eval_lcd.h.

#define LCD_COLOR_DARKBLUE 0x0010

Definition at line 122 of file stm3210c_eval_lcd.h.

#define LCD_COLOR_DARKCYAN 0x0410

Definition at line 125 of file stm3210c_eval_lcd.h.

#define LCD_COLOR_DARKGRAY 0x4208

Definition at line **131** of file **stm3210c_eval_lcd.h**.

#define LCD_COLOR_DARKGREEN 0x0400

Definition at line **123** of file **stm3210c_eval_lcd.h**.

#define LCD_COLOR_DARKMAGENTA 0x8010

Definition at line **126** of file **stm3210c_eval_lcd.h**.

#define LCD_COLOR_DARKRED 0x8000

Definition at line **124** of file **stm3210c_eval_lcd.h**.

#define LCD_COLOR_DARKYELLOW 0x8400

Definition at line **127** of file **stm3210c_eval_lcd.h**.

#define LCD_COLOR_GRAY 0x8410

Definition at line **130** of file **stm3210c_eval_lcd.h**.

#define LCD_COLOR_GREEN 0x07E0

Definition at line **111** of file **stm3210c_eval_lcd.h**.

#define LCD_COLOR_LIGHTBLUE 0x841F

Definition at line **116** of file **stm3210c_eval_lcd.h**.

#define LCD_COLOR_LIGHTCYAN 0x87FF

Definition at line 119 of file stm3210c_eval_lcd.h.

#define LCD_COLOR_LIGHTGRAY 0xD69A

Definition at line 129 of file stm3210c_eval_lcd.h.

#define LCD_COLOR_LIGHTGREEN 0x87F0

Definition at line 117 of file stm3210c_eval_lcd.h.

#define LCD_COLOR_LIGHTMAGENTA 0xFC1F

Definition at line 120 of file stm3210c_eval_lcd.h.

#define LCD_COLOR_LIGHTRED 0xFC10

Definition at line 118 of file stm3210c_eval_lcd.h.

#define LCD_COLOR_LIGHTYELLOW 0xFFF0

Definition at line 121 of file stm3210c_eval_lcd.h.

#define LCD_COLOR_MAGENTA 0xF81F

Definition at line 114 of file stm3210c_eval_lcd.h.

#define LCD_COLOR_ORANGE 0xFD20

Definition at line 134 of file stm3210c_eval_lcd.h.

#define LCD_COLOR_RED 0xF800

Definition at line 112 of file stm3210c_eval_lcd.h.

#define LCD_COLOR_WHITE 0xFFFF

Definition at line 128 of file stm3210c_eval_lcd.h.

#define LCD_COLOR_YELLOW 0xFFE0

Definition at line 115 of file stm3210c_eval_lcd.h.

#define LCD_DEFAULT_FONT Font24

LCD default font.

Definition at line 139 of file stm3210c_eval_lcd.h.

Referenced by BSP_LCD_DrawCircle(), and BSP_LCD_Init().

#define LCD_ERROR 0x01

Definition at line 86 of file stm3210c_eval_lcd.h.

Referenced by BSP_LCD_Init().

#define LCD_OK 0x00

LCD status structure definition.

Definition at line **85** of file **stm3210c_eval_lcd.h**.

Referenced by **BSP_LCD_Init()**.

#define LCD_TIMEOUT 0x02

Definition at line **87** of file **stm3210c_eval_lcd.h**.

Typedef Documentation

typedef struct Point * pPoint

Enumeration Type Documentation

enum Line_ModeTypdef

Line mode structures definition.

Enumerator:

CENTER_MODE	Center mode
RIGHT_MODE	Right mode
LEFT_MODE	Left mode

Definition at line **99** of file **stm3210c_eval_lcd.h**.

STM3210C_EVAL BSP User Manual

STM3210C_EVAL SD

STM3210C-EVAL

Modules

STM3210C_EVAL BSP User Manual

Exported_Types

STM3210C_EVAL SD

Data Structures

struct	**SD_CSD** Card Specific Data: CSD Register. More...
struct	**SD_CID** Card Identification Data: CID Register. More...
struct	**SD_CardInfo** SD Card information. More...

Defines

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

Enumerations

enum	**SD_Info** {
	SD_RESPONSE_NO_ERROR = (0x00),
	SD_IN_IDLE_STATE = (0x01), **SD_ERASE_RESET** = (0x02),
	SD_ILLEGAL_COMMAND = (0x04),
	SD_COM_CRC_ERROR = (0x08),
	SD_ERASE_SEQUENCE_ERROR = (0x10),
	SD_ADDRESS_ERROR = (0x20),
	SD_PARAMETER_ERROR = (0x40),
	SD_RESPONSE_FAILURE = (0xFF), **SD_DATA_OK** = (0x05), **SD_DATA_CRC_ERROR** = (0x0B),
	SD_DATA_WRITE_ERROR = (0x0D),
	SD_DATA_OTHER_ERROR = (0xFF)
	}

Define Documentation

#define MSD_ERROR 0x01

Definition at line 71 of file stm3210c_eval_sd.h.

Referenced by BSP_SD_Erase(), BSP_SD_GetCardInfo(), BSP_SD_Init(), BSP_SD_ReadBlocks(), BSP_SD_WriteBlocks(), SD_GetCIDRegister(), SD_GetCSDRegister(), SD_GoIdleState(), and SD_SendCmd().

#define MSD_OK 0x00

SD status structure definition.

Definition at line 70 of file stm3210c_eval_sd.h.

Referenced by BSP_SD_Erase(), BSP_SD_GetStatus(), BSP_SD_ReadBlocks(), BSP_SD_WriteBlocks(), SD_GetCIDRegister(), SD_GetCSDRegister(), SD_GoIdleState(), and SD_SendCmd().

Enumeration Type Documentation

enum SD_Info

Enumerator:

SD_RESPONSE_NO_ERROR	SD reponses and error flags.
SD_IN_IDLE_STATE	
SD_ERASE_RESET	
SD_ILLEGAL_COMMAND	
SD_COM_CRC_ERROR	
SD_ERASE_SEQUENCE_ERROR	
SD_ADDRESS_ERROR	
SD_PARAMETER_ERROR	
SD_RESPONSE_FAILURE	
SD_DATA_OK	Data response error.
SD_DATA_CRC_ERROR	
SD_DATA_WRITE_ERROR	
SD_DATA_OTHER_ERROR	

Definition at line **73** of file **stm3210c_eval_sd.h**.

STM3210C_EVAL BSP User Manual

stm3210c_eval_sd.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210c_eval_sd.h
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm3210c_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 __STM3210C_EVAL_SD_H
00041 #define __STM3210C_EVAL_SD_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------------------------------------------*/
00048 #include "stm3210c_eval.h"
00049
00050 /** @addtogroup BSP
00051   * @{
00052   */
00053
00054 /** @addtogroup STM3210C_EVAL
00055   * @{
00056   */
00057
00058 /** @addtogroup STM3210C_EVAL_SD
00059   * @{
00060   */
00061
00062
00063 /** @defgroup STM3210C_EVAL_SD_Exported_Types Exported_Types
00064   * @{
00065   */
00066
00067 /**
00068   * @brief  SD status structure definition
```

```
00069   */
00070 #define MSD_OK          0x00
00071 #define MSD_ERROR       0x01
00072
00073 typedef enum
00074 {
00075 /**
00076   * @brief  SD reponses and error flags
00077   */
00078   SD_RESPONSE_NO_ERROR      = (0x00),
00079   SD_IN_IDLE_STATE          = (0x01),
00080   SD_ERASE_RESET            = (0x02),
00081   SD_ILLEGAL_COMMAND        = (0x04),
00082   SD_COM_CRC_ERROR          = (0x08),
00083   SD_ERASE_SEQUENCE_ERROR   = (0x10),
00084   SD_ADDRESS_ERROR          = (0x20),
00085   SD_PARAMETER_ERROR        = (0x40),
00086   SD_RESPONSE_FAILURE       = (0xFF),
00087
00088 /**
00089   * @brief  Data response error
00090   */
00091   SD_DATA_OK                = (0x05),
00092   SD_DATA_CRC_ERROR         = (0x0B),
00093   SD_DATA_WRITE_ERROR       = (0x0D),
00094   SD_DATA_OTHER_ERROR       = (0xFF)
00095 }SD_Info;
00096
00097 /**
00098   * @brief  Card Specific Data: CSD Register
00099   */
00100 typedef struct
00101 {
00102   __IO uint8_t  CSDStruct;            /* CSD
 structure */
00103   __IO uint8_t  SysSpecVersion;       /* Sys
tem specification version */
```

```
00104   __IO uint8_t  Reserved1;           /* Res
erved */
00105   __IO uint8_t  TAAC;                /* Dat
a read access-time 1 */
00106   __IO uint8_t  NSAC;                /* Dat
a read access-time 2 in CLK cycles */
00107   __IO uint8_t  MaxBusClkFrec;       /* Max
. bus clock frequency */
00108   __IO uint16_t CardComdClasses;     /* Car
d command classes */
00109   __IO uint8_t  RdBlockLen;          /* Max
. read data block length */
00110   __IO uint8_t  PartBlockRead;       /* Par
tial blocks for read allowed */
00111   __IO uint8_t  WrBlockMisalign;     /* Wri
te block misalignment */
00112   __IO uint8_t  RdBlockMisalign;     /* Rea
d block misalignment */
00113   __IO uint8_t  DSRImpl;             /* DSR
 implemented */
00114   __IO uint8_t  Reserved2;           /* Res
erved */
00115   __IO uint32_t DeviceSize;          /* Dev
ice Size */
00116   __IO uint8_t  MaxRdCurrentVDDMin;  /* Max
. read current @ VDD min */
00117   __IO uint8_t  MaxRdCurrentVDDMax;  /* Max
. read current @ VDD max */
00118   __IO uint8_t  MaxWrCurrentVDDMin;  /* Max
. write current @ VDD min */
00119   __IO uint8_t  MaxWrCurrentVDDMax;  /* Max
. write current @ VDD max */
00120   __IO uint8_t  DeviceSizeMul;       /* Dev
ice size multiplier */
00121   __IO uint8_t  EraseGrSize;         /* Era
se group size */
00122   __IO uint8_t  EraseGrMul;          /* Era
```

```
se group size multiplier */
00123   __IO uint8_t  WrProtectGrSize;      /* Wri
te protect group size */
00124   __IO uint8_t  WrProtectGrEnable;    /* Wri
te protect group enable */
00125   __IO uint8_t  ManDeflECC;           /* Man
ufacturer default ECC */
00126   __IO uint8_t  WrSpeedFact;          /* Wri
te speed factor */
00127   __IO uint8_t  MaxWrBlockLen;        /* Max
. write data block length */
00128   __IO uint8_t  WriteBlockPaPartial;  /* Par
tial blocks for write allowed */
00129   __IO uint8_t  Reserved3;            /* Res
erded */
00130   __IO uint8_t  ContentProtectAppli;  /* Con
tent protection application */
00131   __IO uint8_t  FileFormatGrouop;     /* Fil
e format group */
00132   __IO uint8_t  CopyFlag;             /* Cop
y flag (OTP) */
00133   __IO uint8_t  PermWrProtect;        /* Per
manent write protection */
00134   __IO uint8_t  TempWrProtect;        /* Tem
porary write protection */
00135   __IO uint8_t  FileFormat;           /* Fil
e Format */
00136   __IO uint8_t  ECC;                  /* ECC
 code */
00137   __IO uint8_t  CSD_CRC;              /* CSD
 CRC */
00138   __IO uint8_t  Reserved4;            /* alw
ays 1*/
00139 } SD_CSD;
00140
00141 /**
00142   * @brief  Card Identification Data: CID Re
```

```
gister
00143   */
00144 typedef struct
00145 {
00146   __IO uint8_t  ManufacturerID;       /* Man
ufacturerID */
00147   __IO uint16_t OEM_AppliID;          /* OEM
/Application ID */
00148   __IO uint32_t ProdName1;            /* Pro
duct Name part1 */
00149   __IO uint8_t  ProdName2;            /* Pro
duct Name part2*/
00150   __IO uint8_t  ProdRev;              /* Pro
duct Revision */
00151   __IO uint32_t ProdSN;               /* Pro
duct Serial Number */
00152   __IO uint8_t  Reserved1;            /* Res
erved1 */
00153   __IO uint16_t ManufactDate;         /* Man
ufacturing Date */
00154   __IO uint8_t  CID_CRC;              /* CID
 CRC */
00155   __IO uint8_t  Reserved2;            /* alw
ays 1 */
00156 } SD_CID;
00157
00158 /**
00159   * @brief SD Card information
00160   */
00161 typedef struct
00162 {
00163   SD_CSD Csd;
00164   SD_CID Cid;
00165   uint32_t CardCapacity;  /* Card Capacity */

00166   uint32_t CardBlockSize; /* Card Block Size
 */
```

```
00167 } SD_CardInfo;
00168
00169 /**
00170   * @}
00171   */
00172
00173 /** @defgroup STM3210C_EVAL_SPI_SD_Exported_
Constants Exported_Constants
00174   * @{
00175   */
00176
00177 /**
00178   * @brief  Block Size
00179   */
00180 #define SD_BLOCK_SIZE    0x200
00181
00182 /**
00183   * @brief  Start Data tokens:
00184   *         Tokens (necessary because at nop
/idle (and CS active) only 0xff is
00185   *         on the data/command line)
00186   */
00187 #define SD_START_DATA_SINGLE_BLOCK_READ    0
xFE  /* Data token start byte, Start Single Block
Read */
00188 #define SD_START_DATA_MULTIPLE_BLOCK_READ  0
xFE  /* Data token start byte, Start Multiple Bloc
k Read */
00189 #define SD_START_DATA_SINGLE_BLOCK_WRITE   0
xFE  /* Data token start byte, Start Single Block
Write */
00190 #define SD_START_DATA_MULTIPLE_BLOCK_WRITE 0
xFD  /* Data token start byte, Start Multiple Bloc
k Write */
00191 #define SD_STOP_DATA_MULTIPLE_BLOCK_WRITE  0
xFD  /* Data toke stop byte, Stop Multiple Block W
rite */
```

```
00192
00193 /**
00194   * @brief  SD detection on its memory slot
00195   */
00196 #define SD_PRESENT              ((uint8_t)0
x01)
00197 #define SD_NOT_PRESENT          ((uint8_t)0
x00)
00198
00199 /**
00200   * @brief  Commands: CMDxx = CMD-number | 0
x40
00201   */
00202 #define SD_CMD_GO_IDLE_STATE          0   /*
 CMD0 = 0x40 */
00203 #define SD_CMD_SEND_OP_COND           1   /*
 CMD1 = 0x41 */
00204 #define SD_CMD_SEND_CSD               9   /*
 CMD9 = 0x49 */
00205 #define SD_CMD_SEND_CID               10  /*
 CMD10 = 0x4A */
00206 #define SD_CMD_STOP_TRANSMISSION      12  /*
 CMD12 = 0x4C */
00207 #define SD_CMD_SEND_STATUS            13  /*
 CMD13 = 0x4D */
00208 #define SD_CMD_SET_BLOCKLEN           16  /*
 CMD16 = 0x50 */
00209 #define SD_CMD_READ_SINGLE_BLOCK      17  /*
 CMD17 = 0x51 */
00210 #define SD_CMD_READ_MULT_BLOCK        18  /*
 CMD18 = 0x52 */
00211 #define SD_CMD_SET_BLOCK_COUNT        23  /*
 CMD23 = 0x57 */
00212 #define SD_CMD_WRITE_SINGLE_BLOCK     24  /*
 CMD24 = 0x58 */
00213 #define SD_CMD_WRITE_MULT_BLOCK       25  /*
 CMD25 = 0x59 */
```

```
00214 #define SD_CMD_PROG_CSD                 27  /*
 CMD27 = 0x5B */
00215 #define SD_CMD_SET_WRITE_PROT           28  /*
 CMD28 = 0x5C */
00216 #define SD_CMD_CLR_WRITE_PROT           29  /*
 CMD29 = 0x5D */
00217 #define SD_CMD_SEND_WRITE_PROT          30  /*
 CMD30 = 0x5E */
00218 #define SD_CMD_SD_ERASE_GRP_START       32  /*
 CMD32 = 0x60 */
00219 #define SD_CMD_SD_ERASE_GRP_END         33  /*
 CMD33 = 0x61 */
00220 #define SD_CMD_UNTAG_SECTOR             34  /*
 CMD34 = 0x62 */
00221 #define SD_CMD_ERASE_GRP_START          35  /*
 CMD35 = 0x63 */
00222 #define SD_CMD_ERASE_GRP_END            36  /*
 CMD36 = 0x64 */
00223 #define SD_CMD_UNTAG_ERASE_GROUP        37  /*
 CMD37 = 0x65 */
00224 #define SD_CMD_ERASE                    38  /*
 CMD38 = 0x66 */
00225
00226
00227 /**
00228   * @}
00229   */
00230
00231 /** @defgroup STM3210C_EVAL_SD_Exported_Macr
o Exported_Macro
00232   * @{
00233   */
00234
00235 /**
00236   * @}
00237   */
00238
```

```
00239 /** @addtogroup STM3210C_EVAL_SD_Exported_Fu
nctions
00240   * @{
00241   */
00242 uint8_t BSP_SD_Init(void);
00243 uint8_t  BSP_SD_IsDetected(void);
00244 uint8_t BSP_SD_ReadBlocks(uint32_t* p32Data,
 uint64_t ReadAddr, uint16_t BlockSize, uint32_t N
umberOfBlocks);
00245 uint8_t BSP_SD_WriteBlocks(uint32_t* p32Data
, uint64_t WriteAddr, uint16_t BlockSize, uint32_t
 NumberOfBlocks);
00246 uint8_t BSP_SD_Erase(uint32_t StartAddr, uin
t32_t EndAddr);
00247 uint8_t BSP_SD_GetStatus(void);
00248 uint8_t BSP_SD_GetCardInfo(SD_CardInfo *pCar
dInfo);
00249
00250 /* Link functions for SD Card peripheral*/
00251 void                  SD_IO_Init(void);
00252 void                  SD_IO_WriteByte(uint
8_t Data);
00253 uint8_t               SD_IO_ReadByte(void)
;
00254 HAL_StatusTypeDef     SD_IO_WriteCmd(uint8
_t Cmd, uint32_t Arg, uint8_t Crc, uint8_t Respons
e);
00255 HAL_StatusTypeDef     SD_IO_WaitResponse(u
int8_t Response);
00256 void                  SD_IO_WriteDummy(void
);
00257
00258 /**
00259   * @}
00260   */
00261
00262 /**
```

```
00263   * @}
00264   */
00265
00266 /**
00267   * @}
00268   */
00269
00270 /**
00271   * @}
00272   */
00273
00274 #ifdef __cplusplus
00275 }
00276 #endif
00277
00278 #endif /* __STM3210C_EVAL_SD_H */
00279
00280 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210C_EVAL BSP User Manual

stm3210c_eval_sd.c

Go to the documentation of this file.

```
00001 /**
00002   ********************************************************************************
00003   * @file    stm3210c_eval_sd.c
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file provides a set of functions needed to manage the SPI SD
00008   *          Card memory mounted on STM3210C-EVAL board.
00009   *          It implements a high level communication layer for read and write
00010   *          from/to this memory. The needed STM32F10x hardware resources (SPI and
00011   *          GPIO) are defined in stm3210c_eval.h file, and the initialization is
00012   *          performed in SD_IO_Init() function declared in stm3210c_eval.c
00013   *          file.
00014   *          You can easily tailor this driver to any other development board,
```

```
00015   *          by just adapting the defines fo
r hardware resources and
00016   *          SD_IO_Init() function.
00017   *
00018   *          +-------------------------------
-------------------------+
00019   *          |                        Pin assig
nment                      |
00020   *          +--------------------------+----
-----------+-------------+
00021   *          |  STM32F10x SPI Pins      |
 SD        |    Pin      |
00022   *          +--------------------------+----
-----------+-------------+
00023   *          | SD_SPI_CS_PIN            |   C
hipSelect  |    2        |
00024   *          | SD_SPI_MOSI_PIN / MOSI   |   D
ataIn      |    3        |
00025   *          |                          |   G
ND         |    9 (0 V)  |
00026   *          |                          |   V
DD         |    4 (3.3 V)|
00027   *          | SD_SPI_SCK_PIN / SCLK    |   C
lock       |    5        |
00028   *          |                          |   G
ND         |    6 (0 V)  |
00029   *          | SD_SPI_MISO_PIN / MISO   |   D
ataOut     |    7        |
00030   *          +--------------------------+----
-----------+-------------+
00031   ****************************************
**********************************
00032   * @attention
00033   *
00034   * <h2><center>&copy; COPYRIGHT(c) 2015 STM
icroelectronics</center></h2>
00035   *
```

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

```
00055   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00056   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00057   *
00058   ****************************************
**********************************
00059   */
00060
00061 /* File Info : ----------------------------
-------------------------------------
00062                                     User NOTES

00063 1. How To use this driver:
00064 --------------------------
00065    - This driver is used to drive the micro
SD external card mounted on STM3210C-EVAL
00066      evaluation board.
00067    - This driver does not need a specific co
mponent driver for the micro SD device
00068      to be included with.
00069
00070 2. Driver description:
00071 ---------------------
00072   + Initialization steps:
00073      o Initialize the micro SD card using th
e BSP_SD_Init() function.
00074      o To check the SD card presence you can
 use the function SD_IsDetected() which
00075        returns the detection status
00076      o The function BSP_SD_GetCardInfo() is
used to get the micro SD card information
00077        which is stored in the structure "HAL
_SD_CardInfoTypedef".
00078
00079   + Micro SD card operations
00080      o The micro SD card can be accessed wit
```

```
h read/write block(s) operations once
00081        it is reay for access. The access can
d be performed in polling
00082        mode by calling the functions BSP_SD_
ReadBlocks()/BSP_SD_WriteBlocks()
00083      o The SD erase block(s) is performed us
ing the function BSP_SD_Erase() with specifying
00084        the number of blocks to erase.
00085      o The SD runtime status is returned whe
n calling the function BSP_SD_GetStatus().
00086
00087 ------------------------------------------
-----------------------------------*/
00088
00089 /* Includes ---------------------------------
-----------------------------------*/
00090 #include "stm3210c_eval_sd.h"
00091
00092 /** @addtogroup BSP
00093   * @{
00094   */
00095
00096 /** @addtogroup STM3210C_EVAL
00097   * @{
00098   */
00099
00100 /** @defgroup STM3210C_EVAL_SD STM3210C_EVAL
 SD
00101   * @{
00102   */
00103
00104 /* Private typedef --------------------------
-----------------------------------*/
00105
00106 /** @defgroup STM3210C_EVAL_SD_Private_Types
_Definitions Private_Types_Definitions
00107   * @{
```

```
00108   */
00109
00110 /**
00111   * @}
00112   */
00113
00114 /* Private define ------------------------
-----------------------------------*/
00115
00116 /** @defgroup STM3210C_EVAL_SD_Private_Defin
es Private_Defines
00117   * @{
00118   */
00119 #define SD_DUMMY_BYTE   0xFF
00120 #define SD_NO_RESPONSE_EXPECTED 0x80
00121 /**
00122   * @}
00123   */
00124
00125 /* Private macro -------------------------
-----------------------------------*/
00126
00127 /** @defgroup STM3210C_EVAL_SD_Private_Macro
s Private_Macros
00128   * @{
00129   */
00130
00131 /**
00132   * @}
00133   */
00134
00135 /* Private variables ---------------------
-----------------------------------*/
00136
00137 /** @defgroup STM3210C_EVAL_SD_Private_Varia
bles Private_Variables
00138   * @{
```

```
00139   */
00140 __IO uint8_t SdStatus = SD_PRESENT;
00141
00142 /**
00143   * @}
00144   */
00145
00146 /* Private function prototypes -----------
-----------------------------------*/
00147 static uint8_t SD_GetCIDRegister(SD_CID* Cid
);
00148 static uint8_t SD_GetCSDRegister(SD_CSD* Csd
);
00149 static SD_Info SD_GetDataResponse(void);
00150 static uint8_t SD_GoIdleState(void);
00151 static uint8_t SD_SendCmd(uint8_t Cmd, uint3
2_t Arg, uint8_t Crc, uint8_t Response);
00152
00153 /** @defgroup STM3210C_EVAL_SD_Private_Funct
ion_Prototypes Private_Function_Prototypes
00154   * @{
00155   */
00156 /**
00157   * @}
00158   */
00159
00160 /* Private functions ----------------------
-----------------------------------*/
00161
00162 /** @defgroup STM3210C_EVAL_SD_Exported_Func
tions Exported Functions
00163   * @{
00164   */
00165
00166 /**
00167   * @brief  Initializes the SD/SD communicat
ion.
```

```
00168   * @retval The SD Response:
00169   *         - MSD_ERROR : Sequence failed
00170   *         - MSD_OK    : Sequence succeed
00171   */
00172 uint8_t BSP_SD_Init(void)
00173 {
00174   /* Configure IO functionalities for SD pin
 */
00175   SD_IO_Init();
00176
00177   /* Check SD card detect pin */
00178   if(BSP_SD_IsDetected()==SD_NOT_PRESENT)
00179   {
00180     SdStatus = SD_NOT_PRESENT;
00181     return MSD_ERROR;
00182   }
00183   else
00184   {
00185     SdStatus = SD_PRESENT;
00186   }
00187
00188   /* SD initialized and set to SPI mode prop
erly */
00189   return (SD_GoIdleState());
00190 }
00191
00192 /**
00193  * @brief  Detects if SD card is correctly p
lugged in the memory slot or not.
00194  * @retval Returns if SD is detected or not
00195  */
00196 uint8_t BSP_SD_IsDetected(void)
00197 {
00198   __IO uint8_t status = SD_PRESENT;
00199
00200   /* Check SD card detect pin */
00201   if(HAL_GPIO_ReadPin(SD_DETECT_GPIO_PORT, S
```

```
D_DETECT_PIN) != GPIO_PIN_RESET)
00202   {
00203     status = SD_NOT_PRESENT;
00204   }
00205
00206   return status;
00207 }
00208
00209 /**
00210   * @brief  Returns information about specif
ic card.
00211   * @param  pCardInfo: pointer to a SD_CardI
nfo structure that contains all SD
00212   *         card information.
00213   * @retval The SD Response:
00214   *         - MSD_ERROR : Sequence failed
00215   *         - MSD_OK    : Sequence succeed
00216   */
00217 uint8_t BSP_SD_GetCardInfo(SD_CardInfo *pCar
dInfo)
00218 {
00219   uint8_t status = MSD_ERROR;
00220
00221   SD_GetCSDRegister(&(pCardInfo->Csd));
00222   status = SD_GetCIDRegister(&(pCardInfo->Cid
));
00223   pCardInfo->CardCapacity = (pCardInfo->Csd.
DeviceSize + 1) ;
00224   pCardInfo->CardCapacity *= (1 << (pCardInf
o->Csd.DeviceSizeMul + 2));
00225   pCardInfo->CardBlockSize = 1 << (pCardInfo
->Csd.RdBlockLen);
00226   pCardInfo->CardCapacity *= pCardInfo->Card
BlockSize;
00227
00228   /* Returns the reponse */
00229   return status;
```

```
00230 }
00231
00232 /**
00233   * @brief  Reads block(s) from a specified
address in an SD card, in polling mode.
00234   * @param  p32Data: Pointer to the buffer t
hat will contain the data to transmit
00235   * @param  ReadAddr: Address from where dat
a is to be read
00236   * @param  BlockSize: SD card data block si
ze, that should be 512
00237   * @param  NumberOfBlocks: Number of SD blo
cks to read
00238   * @retval SD status
00239   */
00240 uint8_t BSP_SD_ReadBlocks(uint32_t* p32Data,
 uint64_t ReadAddr, uint16_t BlockSize, uint32_t N
umberOfBlocks)
00241 {
00242   uint32_t counter = 0, offset = 0;
00243   uint8_t rvalue = MSD_ERROR;
00244   uint8_t *pData = (uint8_t *)p32Data;
00245
00246   /* Send CMD16 (SD_CMD_SET_BLOCKLEN) to set
 the size of the block and
00247      Check if the SD acknowledged the set bl
ock length command: R1 response (0x00: no errors)
*/
00248   if (SD_IO_WriteCmd(SD_CMD_SET_BLOCKLEN, Bl
ockSize, 0xFF, SD_RESPONSE_NO_ERROR) != HAL_OK)
00249   {
00250     return MSD_ERROR;
00251   }
00252
00253   /* Data transfer */
00254   while (NumberOfBlocks--)
00255   {
```

```
00256     /* Send dummy byte: 8 Clock pulses of de
lay */
00257     SD_IO_WriteDummy();
00258
00259     /* Send CMD17 (SD_CMD_READ_SINGLE_BLOCK)
 to read one block */
00260     /* Check if the SD acknowledged the read
 block command: R1 response (0x00: no errors) */
00261     if (SD_IO_WriteCmd(SD_CMD_READ_SINGLE_BL
OCK, ReadAddr + offset, 0xFF, SD_RESPONSE_NO_ERROR
) != HAL_OK)
00262     {
00263       return MSD_ERROR;
00264     }
00265
00266     /* Now look for the data token to signif
y the start of the data */
00267     if (SD_IO_WaitResponse(SD_START_DATA_SIN
GLE_BLOCK_READ) == HAL_OK)
00268     {
00269       /* Read the SD block data : read NumBy
teToRead data */
00270       for (counter = 0; counter < BlockSize;
 counter++)
00271       {
00272         /* Read the pointed data */
00273         *pData = SD_IO_ReadByte();
00274         /* Point to the next location where
the byte read will be saved */
00275         pData++;
00276       }
00277       /* Set next read address*/
00278       offset += BlockSize;
00279       /* get CRC bytes (not really needed by
 us, but required by SD) */
00280       SD_IO_ReadByte();
00281       SD_IO_ReadByte();
```

```
00282       /* Set response value to success */
00283       rvalue = MSD_OK;
00284     }
00285     else
00286     {
00287       /* Set response value to failure */
00288       rvalue = MSD_ERROR;
00289     }
00290   }
00291
00292   /* Send dummy byte: 8 Clock pulses of dela
y */
00293   SD_IO_WriteDummy();
00294   /* Returns the reponse */
00295   return rvalue;
00296 }
00297
00298 /**
00299   * @brief  Writes block(s) to a specified a
ddress in an SD card, in polling mode.
00300   * @param  p32Data: Pointer to the buffer t
hat will contain the data to transmit
00301   * @param  WriteAddr: Address from where da
ta is to be written
00302   * @param  BlockSize: SD card data block si
ze, that should be 512
00303   * @param  NumberOfBlocks: Number of SD blo
cks to write
00304   * @retval SD status
00305   */
00306 uint8_t BSP_SD_WriteBlocks(uint32_t* p32Data
, uint64_t WriteAddr, uint16_t BlockSize, uint32_t
 NumberOfBlocks)
00307 {
00308   uint32_t counter = 0, offset = 0;
00309   uint8_t rvalue = MSD_ERROR;
00310   uint8_t *pData = (uint8_t *)p32Data;
```

```
00311
00312   /* Data transfer */
00313   while (NumberOfBlocks--)
00314   {
00315     /* Send CMD24 (SD_CMD_WRITE_SINGLE_BLOCK
) to write blocks  and
00316        Check if the SD acknowledged the writ
e block command: R1 response (0x00: no errors) */
00317     if (SD_IO_WriteCmd(SD_CMD_WRITE_SINGLE_B
LOCK, WriteAddr + offset, 0xFF, SD_RESPONSE_NO_ERR
OR) != HAL_OK)
00318     {
00319       return MSD_ERROR;
00320     }
00321
00322     /* Send dummy byte */
00323     SD_IO_WriteByte(SD_DUMMY_BYTE);
00324
00325     /* Send the data token to signify the st
art of the data */
00326     SD_IO_WriteByte(SD_START_DATA_SINGLE_BLO
CK_WRITE);
00327
00328     /* Write the block data to SD : write co
unt data by block */
00329     for (counter = 0; counter < BlockSize; c
ounter++)
00330     {
00331       /* Send the pointed byte */
00332       SD_IO_WriteByte(*pData);
00333
00334       /* Point to the next location where th
e byte read will be saved */
00335       pData++;
00336     }
00337
00338     /* Set next write address */
```

```
00339     offset += BlockSize;
00340
00341     /* Put CRC bytes (not really needed by u
s, but required by SD) */
00342     SD_IO_ReadByte();
00343     SD_IO_ReadByte();
00344
00345     /* Read data response */
00346     if (SD_GetDataResponse() == SD_DATA_OK)
00347     {
00348       /* Set response value to success */
00349       rvalue = MSD_OK;
00350     }
00351     else
00352     {
00353       /* Set response value to failure */
00354       rvalue = MSD_ERROR;
00355     }
00356   }
00357
00358   /* Send dummy byte: 8 Clock pulses of dela
y */
00359   SD_IO_WriteDummy();
00360
00361   /* Returns the reponse */
00362   return rvalue;
00363 }
00364
00365 /**
00366   * @brief  Read the CSD card register.
00367   *         Reading the contents of the CSD
register in SPI mode is a simple
00368   *         read-block transaction.
00369   * @param  Csd: pointer on an SCD register
structure
00370   * @retval SD status
00371   */
```

```
00372 uint8_t SD_GetCSDRegister(SD_CSD* Csd)
00373 {
00374   uint32_t counter = 0;
00375   uint8_t rvalue = MSD_ERROR;
00376   uint8_t CSD_Tab[16];
00377
00378   /* Send CMD9 (CSD register) or CMD10(CSD r
egister) and Wait for response in the R1 format (0
x00 is no errors) */
00379   if (SD_IO_WriteCmd(SD_CMD_SEND_CSD, 0, 0xF
F, SD_RESPONSE_NO_ERROR) == HAL_OK)
00380   {
00381     if (SD_IO_WaitResponse(SD_START_DATA_SIN
GLE_BLOCK_READ) == HAL_OK)
00382     {
00383       for (counter = 0; counter < 16; counte
r++)
00384       {
00385         /* Store CSD register value on CSD_T
ab */
00386         CSD_Tab[counter] = SD_IO_ReadByte();
00387       }
00388
00389       /* Get CRC bytes (not really needed by
 us, but required by SD) */
00390       SD_IO_WriteByte(SD_DUMMY_BYTE);
00391       SD_IO_WriteByte(SD_DUMMY_BYTE);
00392
00393       /* Set response value to success */
00394       rvalue = MSD_OK;
00395     }
00396   }
00397   /* Send dummy byte: 8 Clock pulses of dela
y */
00398   SD_IO_WriteDummy();
00399
00400   if(rvalue == SD_RESPONSE_NO_ERROR)
```

```
00401   {
00402     /* Byte 0 */
00403     Csd->CSDStruct = (CSD_Tab[0] & 0xC0) >>
6;
00404     Csd->SysSpecVersion = (CSD_Tab[0] & 0x3C
) >> 2;
00405     Csd->Reserved1 = CSD_Tab[0] & 0x03;
00406
00407     /* Byte 1 */
00408     Csd->TAAC = CSD_Tab[1];
00409
00410     /* Byte 2 */
00411     Csd->NSAC = CSD_Tab[2];
00412
00413     /* Byte 3 */
00414     Csd->MaxBusClkFrec = CSD_Tab[3];
00415
00416     /* Byte 4 */
00417     Csd->CardComdClasses = CSD_Tab[4] << 4;
00418
00419     /* Byte 5 */
00420     Csd->CardComdClasses |= (CSD_Tab[5] & 0x
F0) >> 4;
00421     Csd->RdBlockLen = CSD_Tab[5] & 0x0F;
00422
00423     /* Byte 6 */
00424     Csd->PartBlockRead = (CSD_Tab[6] & 0x80)
 >> 7;
00425     Csd->WrBlockMisalign = (CSD_Tab[6] & 0x4
0) >> 6;
00426     Csd->RdBlockMisalign = (CSD_Tab[6] & 0x2
0) >> 5;
00427     Csd->DSRImpl = (CSD_Tab[6] & 0x10) >> 4;
00428     Csd->Reserved2 = 0; /*!< Reserved */
00429
00430     Csd->DeviceSize = (CSD_Tab[6] & 0x03) <<
 10;
```

```
00431
00432     /* Byte 7 */
00433     Csd->DeviceSize |= (CSD_Tab[7]) << 2;
00434
00435     /* Byte 8 */
00436     Csd->DeviceSize |= (CSD_Tab[8] & 0xC0) >
> 6;
00437
00438     Csd->MaxRdCurrentVDDMin = (CSD_Tab[8] &
0x38) >> 3;
00439     Csd->MaxRdCurrentVDDMax = (CSD_Tab[8] &
0x07);
00440
00441     /* Byte 9 */
00442     Csd->MaxWrCurrentVDDMin = (CSD_Tab[9] &
0xE0) >> 5;
00443     Csd->MaxWrCurrentVDDMax = (CSD_Tab[9] &
0x1C) >> 2;
00444     Csd->DeviceSizeMul = (CSD_Tab[9] & 0x03)
 << 1;
00445     /* Byte 10 */
00446     Csd->DeviceSizeMul |= (CSD_Tab[10] & 0x8
0) >> 7;
00447
00448     Csd->EraseGrSize = (CSD_Tab[10] & 0x40)
>> 6;
00449     Csd->EraseGrMul = (CSD_Tab[10] & 0x3F) <
< 1;
00450
00451     /* Byte 11 */
00452     Csd->EraseGrMul |= (CSD_Tab[11] & 0x80)
>> 7;
00453     Csd->WrProtectGrSize = (CSD_Tab[11] & 0x
7F);
00454
00455     /* Byte 12 */
00456     Csd->WrProtectGrEnable = (CSD_Tab[12] &
```

```
0x80) >> 7;
00457     Csd->ManDeflECC = (CSD_Tab[12] & 0x60) >
> 5;
00458     Csd->WrSpeedFact = (CSD_Tab[12] & 0x1C)
>> 2;
00459     Csd->MaxWrBlockLen = (CSD_Tab[12] & 0x03
) << 2;
00460
00461     /* Byte 13 */
00462     Csd->MaxWrBlockLen |= (CSD_Tab[13] & 0xC
0) >> 6;
00463     Csd->WriteBlockPaPartial = (CSD_Tab[13]
& 0x20) >> 5;
00464     Csd->Reserved3 = 0;
00465     Csd->ContentProtectAppli = (CSD_Tab[13]
& 0x01);
00466
00467     /* Byte 14 */
00468     Csd->FileFormatGrouop = (CSD_Tab[14] & 0
x80) >> 7;
00469     Csd->CopyFlag = (CSD_Tab[14] & 0x40) >>
6;
00470     Csd->PermWrProtect = (CSD_Tab[14] & 0x20
) >> 5;
00471     Csd->TempWrProtect = (CSD_Tab[14] & 0x10
) >> 4;
00472     Csd->FileFormat = (CSD_Tab[14] & 0x0C) >
> 2;
00473     Csd->ECC = (CSD_Tab[14] & 0x03);
00474
00475     /* Byte 15 */
00476     Csd->CSD_CRC = (CSD_Tab[15] & 0xFE) >> 1
;
00477     Csd->Reserved4 = 1;
00478   }
00479   /* Return the reponse */
00480   return rvalue;
```

```
00481 }
00482
00483 /**
00484   * @brief  Read the CID card register.
00485   *         Reading the contents of the CID
register in SPI mode is a simple
00486   *         read-block transaction.
00487   * @param  Cid: pointer on an CID register
structure
00488   * @retval SD status
00489   */
00490 static uint8_t SD_GetCIDRegister(SD_CID* Cid
)
00491 {
00492   uint32_t counter = 0;
00493   uint8_t rvalue = MSD_ERROR;
00494   uint8_t CID_Tab[16];
00495
00496   /* Send CMD10 (CID register) and Wait for
response in the R1 format (0x00 is no errors) */
00497   if (SD_IO_WriteCmd(SD_CMD_SEND_CID, 0, 0xF
F, SD_RESPONSE_NO_ERROR) == HAL_OK)
00498   {
00499     if (SD_IO_WaitResponse(SD_START_DATA_SIN
GLE_BLOCK_READ) == HAL_OK)
00500     {
00501       /* Store CID register value on CID_Tab
 */
00502       for (counter = 0; counter < 16; counte
r++)
00503       {
00504         CID_Tab[counter] = SD_IO_ReadByte();
00505       }
00506
00507       /* Get CRC bytes (not really needed by
 us, but required by SD) */
00508       SD_IO_WriteByte(SD_DUMMY_BYTE);
```

```
00509       SD_IO_WriteByte(SD_DUMMY_BYTE);
00510
00511       /* Set response value to success */
00512       rvalue = MSD_OK;
00513     }
00514   }
00515
00516   /* Send dummy byte: 8 Clock pulses of dela
y */
00517   SD_IO_WriteDummy();
00518
00519   if(rvalue == MSD_OK)
00520   {
00521     /* Byte 0 */
00522     Cid->ManufacturerID = CID_Tab[0];
00523
00524     /* Byte 1 */
00525     Cid->OEM_AppliID = CID_Tab[1] << 8;
00526
00527     /* Byte 2 */
00528     Cid->OEM_AppliID |= CID_Tab[2];
00529
00530     /* Byte 3 */
00531     Cid->ProdName1 = CID_Tab[3] << 24;
00532
00533     /* Byte 4 */
00534     Cid->ProdName1 |= CID_Tab[4] << 16;
00535
00536     /* Byte 5 */
00537     Cid->ProdName1 |= CID_Tab[5] << 8;
00538
00539     /* Byte 6 */
00540     Cid->ProdName1 |= CID_Tab[6];
00541
00542     /* Byte 7 */
00543     Cid->ProdName2 = CID_Tab[7];
00544
```

```
00545     /* Byte 8 */
00546     Cid->ProdRev = CID_Tab[8];
00547
00548     /* Byte 9 */
00549     Cid->ProdSN = CID_Tab[9] << 24;
00550
00551     /* Byte 10 */
00552     Cid->ProdSN |= CID_Tab[10] << 16;
00553
00554     /* Byte 11 */
00555     Cid->ProdSN |= CID_Tab[11] << 8;
00556
00557     /* Byte 12 */
00558     Cid->ProdSN |= CID_Tab[12];
00559
00560     /* Byte 13 */
00561     Cid->Reserved1 |= (CID_Tab[13] & 0xF0) >
> 4;
00562     Cid->ManufactDate = (CID_Tab[13] & 0x0F)
 << 8;
00563
00564     /* Byte 14 */
00565     Cid->ManufactDate |= CID_Tab[14];
00566
00567     /* Byte 15 */
00568     Cid->CID_CRC = (CID_Tab[15] & 0xFE) >> 1
;
00569     Cid->Reserved2 = 1;
00570   }
00571   /* Return the reponse */
00572   return rvalue;
00573 }
00574
00575 /**
00576   * @brief  Send 5 bytes command to the SD c
ard and get response
00577   * @param  Cmd: The user expected command t
```

```
o send to SD card.
00578   * @param  Arg: The command argument.
00579   * @param  Crc: The CRC.
00580   * @param  Response: Expected response from
 the SD card
00581   * @retval SD status
00582   */
00583 static uint8_t SD_SendCmd(uint8_t Cmd, uint3
2_t Arg, uint8_t Crc, uint8_t Response)
00584 {
00585   uint8_t status = MSD_ERROR;
00586
00587   if(SD_IO_WriteCmd(Cmd, Arg, Crc, Response)
 == HAL_OK)
00588   {
00589     status = MSD_OK;
00590   }
00591
00592   /* Send Dummy Byte */
00593   SD_IO_WriteDummy();
00594
00595   return status;
00596 }
00597
00598 /**
00599   * @brief  Get SD card data response.
00600   * @retval The SD status: Read data respons
e xxx0<status>1
00601   *         - status 010: Data accecpted
00602   *         - status 101: Data rejected due
to a crc error
00603   *         - status 110: Data rejected due
to a Write error.
00604   *         - status 111: Data rejected due
to other error.
00605   */
00606 static SD_Info SD_GetDataResponse(void)
```

```
00607 {
00608   uint32_t counter = 0;
00609   SD_Info response, rvalue;
00610
00611   while (counter <= 64)
00612   {
00613     /* Read response */
00614     response = (SD_Info)SD_IO_ReadByte();
00615     /* Mask unused bits */
00616     response &= 0x1F;
00617     switch (response)
00618     {
00619       case SD_DATA_OK:
00620       {
00621         rvalue = SD_DATA_OK;
00622         break;
00623       }
00624       case SD_DATA_CRC_ERROR:
00625         return SD_DATA_CRC_ERROR;
00626       case SD_DATA_WRITE_ERROR:
00627         return SD_DATA_WRITE_ERROR;
00628       default:
00629       {
00630         rvalue = SD_DATA_OTHER_ERROR;
00631         break;
00632       }
00633     }
00634     /* Exit loop in case of data ok */
00635     if (rvalue == SD_DATA_OK)
00636       break;
00637     /* Increment loop counter */
00638     counter++;
00639   }
00640
00641   /* Wait null data */
00642   while (SD_IO_ReadByte() == 0);
00643
```

```
00644   /* Return response */
00645   return response;
00646 }
00647
00648 /**
00649   * @brief  Returns the SD status.
00650   * @retval The SD status.
00651   */
00652 uint8_t BSP_SD_GetStatus(void)
00653 {
00654   return MSD_OK;
00655 }
00656
00657 /**
00658   * @brief  Put SD in Idle state.
00659   * @retval SD status
00660   */
00661 static uint8_t SD_GoIdleState(void)
00662 {
00663   /* Send CMD0 (SD_CMD_GO_IDLE_STATE) to put
 SD in SPI mode and
00664      Wait for In Idle State Response (R1 For
mat) equal to 0x01 */
00665   if (SD_SendCmd(SD_CMD_GO_IDLE_STATE, 0, 0x
95, SD_IN_IDLE_STATE) != MSD_OK)
00666   {
00667     /* No Idle State Response: return respon
se failure */
00668     return MSD_ERROR;
00669   }
00670
00671   /*----------Activates the card initializat
ion process-----------*/
00672   /* Send CMD1 (Activates the card process)
until response equal to 0x0 and
00673      Wait for no error Response (R1 Format)
equal to 0x00 */
```

```
00674   while (SD_SendCmd(SD_CMD_SEND_OP_COND, 0,
0xFF, SD_RESPONSE_NO_ERROR) != MSD_OK);
00675
00676   return MSD_OK;
00677 }
00678 /**
00679   * @brief  Erases the specified memory area
 of the given SD card.
00680   * @param  StartAddr: Start byte address
00681   * @param  EndAddr: End byte address
00682   * @retval SD status
00683   */
00684 uint8_t BSP_SD_Erase(uint32_t StartAddr, uin
t32_t EndAddr)
00685 {
00686   uint8_t rvalue = MSD_ERROR;
00687
00688   /* Send CMD32 (Erase group start) and chec
k if the SD acknowledged the erase command: R1 res
ponse (0x00: no errors) */
00689   if (SD_SendCmd(SD_CMD_SD_ERASE_GRP_START,
StartAddr, 0xFF, SD_RESPONSE_NO_ERROR) == MSD_OK)
00690   {
00691     /* Send CMD33 (Erase group end) and Chec
k if the SD acknowledged the erase command: R1 res
ponse (0x00: no errors) */
00692     if (SD_SendCmd(SD_CMD_SD_ERASE_GRP_END,
EndAddr, 0xFF, SD_RESPONSE_NO_ERROR) == MSD_OK)
00693     {
00694       /* Send CMD38 (Erase) and Check if the
 SD acknowledged the erase command: R1 response (0
x00: no errors) */
00695       if (SD_SendCmd(SD_CMD_ERASE, 0, 0xFF,
SD_RESPONSE_NO_ERROR) == MSD_OK)
00696       {
00697           rvalue = MSD_OK;
00698       }
```

```
00699     }
00700   }
00701
00702   /* Return the reponse */
00703   return rvalue;
00704 }
00705 /**
00706   * @}
00707   */
00708
00709 /**
00710   * @}
00711   */
00712
00713 /**
00714   * @}
00715   */
00716
00717 /**
00718   * @}
00719   */
00720
00721 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210C_EVAL BSP User Manual

STM3210C_EVAL Touch Screen

STM3210C-EVAL

Modules

STM3210C_EVAL BSP User Manual

Exported_Types

STM3210C_EVAL Touch Screen

Data Structures

struct	**TS_StateTypeDef**

STM3210C_EVAL BSP User Manual

stm3210c_eval_ts.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210c_eval_ts.h
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm3210c_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 __STM3210C_EVAL_TS_H
00041 #define __STM3210C_EVAL_TS_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------------------------------------------*/
00048 #include "stm3210c_eval.h"
00049 #include "../Components/stmpe811/stmpe811.h"

00050
00051 /** @addtogroup BSP
00052   * @{
00053   */
00054
00055 /** @addtogroup STM3210C_EVAL
00056   * @{
00057   */
00058
00059 /** @addtogroup STM3210C_EVAL_TS
00060   * @{
00061   */
00062
00063 /* Exported types ------------------------------------------------------------*/
00064
00065 /** @defgroup STM3210C_EVAL_TS_Exported_Types Exported_Types
00066   * @{
```

```
00067   */
00068 typedef struct
00069 {
00070   uint16_t TouchDetected;
00071   uint16_t x;
00072   uint16_t y;
00073   uint16_t z;
00074
00075 }TS_StateTypeDef;
00076
00077 /**
00078   * @}
00079   */
00080
00081 /* Exported constants ----------------------
----------------------------------*/
00082
00083
00084 /** @defgroup STM3210C_EVAL_TS_Exported_Cons
tants Exported_Constants
00085   * @{
00086   */
00087 #define TS_SWAP_NONE                      0x00
00088 #define TS_SWAP_X                         0x01
00089 #define TS_SWAP_Y                         0x02
00090 #define TS_SWAP_XY                        0x04
00091
00092 typedef enum
00093 {
00094   TS_OK       = 0x00,
00095   TS_ERROR    = 0x01,
00096   TS_TIMEOUT  = 0x02
00097
00098 }TS_StatusTypeDef;
00099
00100 /**
00101   * @}
```

```
00102   */
00103
00104 /* Exported macro ------------------------
------------------------------------*/
00105
00106 /** @defgroup STM3210C_EVAL_TS_Exported_Macr
os Exported_Macros
00107   * @{
00108   */
00109
00110 /**
00111   * @}
00112   */
00113
00114 /* Exported functions ---------------------
------------------------------------*/
00115
00116 /** @defgroup STM3210C_EVAL_TS_Exported_Func
tions Exported_Functions
00117   * @{
00118   */
00119
00120 uint8_t BSP_TS_Init(uint16_t xSize, uint16_t
 ySize);
00121 uint8_t BSP_TS_GetState(TS_StateTypeDef *TS_
State);
00122 uint8_t BSP_TS_ITConfig(void);
00123 uint8_t BSP_TS_ITGetStatus(void);
00124 void    BSP_TS_ITClear(void);
00125
00126 #ifdef __cplusplus
00127 }
00128 #endif
00129 #endif /* __STM3210C_EVAL_TS_H */
00130
00131 /**
00132   * @}
```

```
00133   */
00134
00135 /**
00136   * @}
00137   */
00138
00139 /**
00140   * @}
00141   */
00142
00143 /**
00144   * @}
00145   */
00146
00147 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210C_EVAL BSP User Manual

stm3210c_eval_ts.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210c_eval_ts.c
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file provides a set of functions needed to manage the touch
00008   *          screen on STM3210C_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 STM3210C-EVAL
00044      evaluation board on the ILI9325 LCD mounted on MB785 daughter board .
00045    - The STMPE811 IO expander device component driver must be included with this
00046      driver in order to run the TS module commanded by the IO expander device
00047      mounted on the evaluation board.
00048
00049 2. Driver description:
00050 ---------------------
00051   + Initialization steps:
00052      o Initialize the TS module using the BSP_TS_Init() function. This
00053        function includes the MSP layer hardware resources initialization and the
00054        communication layer configuration to start the TS use. The LCD size properties
00055        (x and y) are passed as parameters.
00056      o If TS interrupt mode is desired, you must configure the TS interrupt mode
00057        by calling the function BSP_TS_ITConfig(). The TS interrupt mode is generated
00058        as an external interrupt whenever a touch is detected.
```

```
00059
00060   + Touch screen use
00061       o The touch screen state is captured wh
enever the function BSP_TS_GetState() is
00062         used. This function returns informati
on about the last LCD touch occurred
00063         in the TS_StateTypeDef structure.
00064       o If TS interrupt mode is used, the fun
ction BSP_TS_ITGetStatus() is needed to get
00065         the interrupt status. To clear the IT
 pending bits, you should call the
00066         function BSP_TS_ITClear().
00067       o The IT is handled using the correspon
ding external interrupt IRQ handler,
00068         the user IT callback treatment is imp
lemented on the same external interrupt
00069         callback.
00070
00071 ------------------------------------------
-----------------------------------*/
00072
00073 /* Includes ---------------------------------
-----------------------------------*/
00074 #include "stm3210c_eval_ts.h"
00075
00076 /** @addtogroup BSP
00077   * @{
00078   */
00079
00080 /** @addtogroup STM3210C_EVAL
00081   * @{
00082   */
00083
00084 /** @defgroup STM3210C_EVAL_TS STM3210C_EVAL
 Touch Screen
00085   * @{
00086   */
```

```
00087
00088 /* Private typedef -------------------------
-----------------------------------*/
00089
00090 /** @defgroup STM3210C_EVAL_TS_Private_Types
_Definitions Private_Types_Definitions
00091   * @{
00092   */
00093
00094 /**
00095   * @}
00096   */
00097
00098 /* Private define -------------------------
-----------------------------------*/
00099
00100 /** @defgroup STM3210C_EVAL_TS_Private_Defin
es Private_Defines
00101   * @{
00102   */
00103
00104 /**
00105   * @}
00106   */
00107
00108 /* Private macro -------------------------
-----------------------------------*/
00109
00110 /** @defgroup STM3210C_EVAL_TS_Private_Macro
s Private_Macros
00111   * @{
00112   */
00113
00114 /**
00115   * @}
00116   */
00117
```

```
00118 /* Private variables ---------------------
-----------------------------------*/
00119
00120 /** @defgroup STM3210C_EVAL_TS_Private_Varia
bles Private_Variables
00121   * @{
00122   */
00123 static TS_DrvTypeDef *ts_driver;
00124 static uint16_t ts_x_boundary, ts_y_boundary
;
00125 static uint8_t  ts_orientation;
00126
00127 /**
00128   * @}
00129   */
00130
00131 /* Private function prototypes -------------
-----------------------------------*/
00132
00133 /** @defgroup STM3210C_EVAL_TS_Private_Funct
ion_Prototypes Private_Function_Prototypes
00134   * @{
00135   */
00136
00137 /**
00138   * @}
00139   */
00140
00141 /* Private functions ----------------------
-----------------------------------*/
00142
00143 /** @defgroup STM3210C_EVAL_TS_Exported_Func
tions Exported_Functions
00144   * @{
00145   */
00146
00147 /**
```

```
00148   * @brief  Initializes and configures the t
ouch screen functionalities and
00149   *         configures all necessary hardwar
e resources (GPIOs, clocks..).
00150   * @param  xSize: Maximum X size of the TS
area on LCD
00151   * @param  ySize: Maximum Y size of the TS
area on LCD
00152   * @retval TS_OK: if all initializations ar
e OK. Other value if error.
00153   */
00154 uint8_t BSP_TS_Init(uint16_t xSize, uint16_t
 ySize)
00155 {
00156   uint8_t ret = TS_ERROR;
00157
00158   if(stmpe811_ts_drv.ReadID(TS_I2C_ADDRESS)
== STMPE811_ID)
00159   {
00160     /* Initialize the TS driver structure */
00161     ts_driver = &stmpe811_ts_drv;
00162
00163     /* Initialize x and y positions boundari
es */
00164     ts_x_boundary  = xSize;
00165     ts_y_boundary  = ySize;
00166     ts_orientation = TS_SWAP_XY;
00167     ret = TS_OK;
00168   }
00169
00170   if(ret == TS_OK)
00171   {
00172     /* Initialize the LL TS Driver */
00173     ts_driver->Reset(TS_I2C_ADDRESS);
00174     ts_driver->Init(TS_I2C_ADDRESS);
00175     ts_driver->Start(TS_I2C_ADDRESS);
00176   }
```

```
00177
00178   return ret;
00179 }
00180
00181 /**
00182   * @brief  Configures and enables the touch
 screen interrupts.
00183   * @retval TS_OK: if all initializations ar
e OK. Other value if error.
00184   */
00185 uint8_t BSP_TS_ITConfig(void)
00186 {
00187   /* Call component driver to enable TS ITs
*/
00188   ts_driver->EnableIT(TS_I2C_ADDRESS);
00189
00190   return TS_OK;
00191 }
00192
00193 /**
00194   * @brief  Gets the touch screen interrupt
status.
00195   * @retval TS_OK: if all initializations ar
e OK. Other value if error.
00196   */
00197 uint8_t BSP_TS_ITGetStatus(void)
00198 {
00199   /* Call component driver to enable TS ITs
*/
00200   return (ts_driver->GetITStatus(TS_I2C_ADDR
ESS));
00201 }
00202
00203 /**
00204   * @brief  Returns status and positions of
the touch screen.
00205   * @param  TS_State: Pointer to touch scree
```

```
n current state structure
00206   * @retval TS_OK: if all initializations ar
e OK. Other value if error.
00207   */
00208 uint8_t BSP_TS_GetState(TS_StateTypeDef *TS_
State)
00209 {
00210   static uint32_t _x = 0, _y = 0;
00211   uint16_t xDiff, yDiff , x , y;
00212   uint16_t swap;
00213
00214   TS_State->TouchDetected = ts_driver->Detec
tTouch(TS_I2C_ADDRESS);
00215
00216   if(TS_State->TouchDetected)
00217   {
00218     ts_driver->GetXY(TS_I2C_ADDRESS, &x, &y)
;
00219
00220     if(ts_orientation & TS_SWAP_X)
00221     {
00222       x = 4096 - x;
00223     }
00224
00225     if(ts_orientation & TS_SWAP_Y)
00226     {
00227       y = 4096 - y;
00228     }
00229
00230     if(ts_orientation & TS_SWAP_XY)
00231     {
00232       swap = y;
00233       y = x;
00234       x = swap;
00235     }
00236
00237     xDiff = x > _x? (x - _x): (_x - x);
```

```
00238     yDiff = y > _y? (y - _y): (_y - y);
00239
00240     if (xDiff + yDiff > 5)
00241     {
00242       _x = x;
00243       _y = y;
00244     }
00245
00246     TS_State->x = (ts_x_boundary * _x) >> 12
;
00247     TS_State->y = (ts_y_boundary * _y) >> 12
;
00248   }
00249
00250   return TS_OK;
00251 }
00252
00253 /**
00254   * @brief  Clears all touch screen interrup
ts.
00255   * @retval None
00256   */
00257 void BSP_TS_ITClear(void)
00258 {
00259   ts_driver->ClearIT(TS_I2C_ADDRESS);
00260 }
00261
00262 /**
00263   * @}
00264   */
00265
00266 /**
00267   * @}
00268   */
00269
00270 /**
00271   * @}
```

```
00272   */
00273
00274 /**
00275   * @}
00276   */
00277 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM3210C_EVAL BSP User Manual

STM3210C_EVAL_ACCELEROMETER

This file includes the motion sensor driver for ACCELEROMETER motion sensor devices. More...

Modules

Detailed Description

This file includes the motion sensor driver for ACCELEROMETER motion sensor devices.

STM3210C_EVAL BSP User Manual

STM3210C_EVAL_AUDIO

STM3210C-EVAL

This file includes the low layer audio driver available on STM3210C-Eval eval board. More...

Modules

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

Detailed Description

This file includes the low layer audio driver available on STM3210C-Eval eval board.

STM3210C_EVAL BSP User Manual

STM3210C_EVAL_EEPROM

This file includes the I2C and SPI EEPROM driver of STM3210C-EVAL board. More...

Modules

Detailed Description

This file includes the I2C and SPI EEPROM driver of STM3210C-EVAL board.

STM3210C_EVAL BSP User Manual

Exported Types

STM3210C_EVAL_EEPROM

Data Structures

struct	**EEPROM_DrvTypeDef**

STM3210C_EVAL BSP User Manual

stm3210c_eval_eeprom.h

Go to the documentation of this file.

```
00001 /**
00002   ********************************************************************************
00003   * @file    stm3210c_eval_eeprom.h
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file contains all the functions prototypes for
00008   *          the stm3210c_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 __STM3210C_EVAL_EEPROM_H
00041 #define __STM3210C_EVAL_EEPROM_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------
-----------------------------------*/
00048 #include "stm3210c_eval.h"
00049
00050 /** @addtogroup BSP
00051   * @{
00052   */
00053
00054 /** @addtogroup STM3210C_EVAL
00055   * @{
00056   */
00057
00058 /** @addtogroup STM3210C_EVAL_EEPROM
00059   * @{
00060   */
00061
00062 /** @defgroup STM3210C_EVAL_EEPROM_Exported_
Types Exported Types
00063   * @{
00064   */
00065 typedef struct
00066 {
00067   uint32_t  (*Init)(void);
```

```
00068   uint32_t  (*ReadBuffer)(uint8_t* , uint16_
t , uint32_t* );
00069   uint32_t  (*WritePage)(uint8_t* , uint16_t
 , uint32_t* );
00070 }EEPROM_DrvTypeDef;
00071 /**
00072   * @}
00073   */
00074
00075 /** @defgroup STM3210C_EVAL_EEPROM_Exported_
Constants Exported Constants
00076   * @{
00077   */
00078 /* EEPROMs hardware address and page size */

00079 #define EEPROM_ADDRESS_M24C64_32      0xA0
  /* Support the devices: M24C32 and M24C64 */
00080 /* The M24C08W contains 4 blocks (128byte ea
ch) with the adresses below: E2 = 0
00081    EEPROM Addresses defines */
00082 #define EEPROM_ADDRESS_M24C08_BLOCK0      0x
A0
00083 #define EEPROM_ADDRESS_M24C08_BLOCK1      0x
A2
00084 #define EEPROM_ADDRESS_M24C08_BLOCK2      0x
A4
00085 #define EEPROM_ADDRESS_M24C08_BLOCK3      0x
A6
00086
00087 #define EEPROM_PAGESIZE_M24C64_32         32
       /* Support the devices: M24C32 and M24C64 */

00088 #define EEPROM_PAGESIZE_M24C08            16
       /* Support the device: M24C08. */
00089
00090 /* EEPROM BSP return values */
00091 #define EEPROM_OK                      0
```

```
00092 #define EEPROM_FAIL                     1
00093 #define EEPROM_TIMEOUT                  2
00094
00095 /* EEPROM BSP devices definition list suppor
ted */
00096 #define BSP_EEPROM_M24C64_32              1
      /* RF I2C EEPROM M24C32 and M24C64 */
00097 #define BSP_EEPROM_M24C08                 2
      /* RF I2C EEPROM M24C08 */
00098
00099 /* Maximum number of trials for EEPROM_I2C_W
aitEepromStandbyState() function */
00100 #define EEPROM_MAX_TRIALS               300
00101 /**
00102   * @}
00103   */
00104
00105 /** @defgroup STM3210C_EVAL_EEPROM_Exported_
Macros Exported Macros
00106   * @{
00107   */
00108 /**
00109   * @}
00110   */
00111
00112 /** @addtogroup STM3210C_EVAL_EEPROM_Exporte
d_Functions
00113   * @{
00114   */
00115 uint32_t  BSP_EEPROM_Init(void);
00116 void      BSP_EEPROM_SelectDevice(uint8_t De
viceID);
00117 uint32_t  BSP_EEPROM_ReadBuffer(uint8_t* pBu
ffer, uint16_t ReadAddr, uint32_t* NumByteToRead);
00118 uint32_t  BSP_EEPROM_WriteBuffer(uint8_t* pB
uffer, uint16_t WriteAddr, uint32_t NumByteToWrite
);
```

```
00119
00120 /* USER Callbacks: This function is declared
 as __weak in EEPROM driver and
00121    should be implemented into user applicati
on.
00122    BSP_EEPROM_TIMEOUT_UserCallback() functio
n is called whenever a timeout condition
00123    occure during communication (waiting on a
n event that doesn't occur, bus
00124    errors, busy devices ...). */
00125 void BSP_EEPROM_TIMEOUT_UserCallback(void);
00126
00127
00128 /* Link functions for I2C EEPROM peripheral
*/
00129 void                    EEPROM_I2C_IO_Init(v
oid);
00130 HAL_StatusTypeDef       EEPROM_I2C_IO_WriteD
ata(uint16_t DevAddress, uint16_t MemAddress, uint
8_t* pBuffer, uint32_t BufferSize);
00131 HAL_StatusTypeDef       EEPROM_I2C_IO_ReadDa
ta(uint16_t DevAddress, uint16_t MemAddress, uint8
_t* pBuffer, uint32_t BufferSize);
00132 HAL_StatusTypeDef       EEPROM_I2C_IO_IsDevi
ceReady(uint16_t DevAddress, uint32_t Trials);
00133
00134 #ifdef __cplusplus
00135 }
00136 #endif
00137
00138 #endif /* __STM3210C_EVAL_EEPROM_H */
00139 /**
00140   * @}
00141   */
00142
00143 /**
00144   * @}
```

```
00145   */
00146
00147 /**
00148   * @}
00149   */
00150
00151 /**
00152   * @}
00153   */
00154
00155 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210C_EVAL BSP User Manual

stm3210c_eval_eeprom.c

Go to the documentation of this file.

```
00001 /**
00002   ********************************************************************************
00003   * @file    stm3210c_eval_eeprom.c
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file provides a set of functions needed to manage a M24C64
00008   *          I2C EEPROM memory.
00009   *
00010   *          ===================================================================
00011   *          Notes:
00012   *           - This driver is intended for STM32F1xx families devices only.
00013   *           - The I2C EEPROM memory (M24CXX) is available directly
00014   *             on STM3210C EVAL board.
00015   *          ===================================================================
00016   *
```

```
00017   *          It implements a high level comm
unication layer for read and write
00018   *          from/to this memory. The needed
 STM32F10x hardware resources (I2C
00019   *          and GPIO) are defined in stm321
0c_eval.h file,
00020   *          and the initialization is perfo
rmed
00021   *          in EEPROM_I2C_IO_Init() functio
ns
00022   *          declared in stm3210c_eval.c fil
e.
00023   *          You can easily tailor this driv
er to any other development board,
00024   *          by just adapting the defines fo
r hardware resources and
00025   *          EEPROM_I2C_IO_Init() functions.

00026   *
00027   *          @note In this driver, basic rea
d and write functions
00028   *          (BSP_EEPROM_ReadBuffer() and BS
P_EEPROM_WriteBuffer())
00029   *          use Polling mode to perform the
 data transfer to/from EEPROM memories.
00030   *     +-----------------------------------
-------------------------------+
00031   *     |                Pin assignment for M
24CXX EEPROM                  |
00032   *     +-----------------------------------
----+-----------+-------------+
00033   *     |  STM32F1xx I2C Pins
    |   EEPROM  |   Pin       |
00034   *     +-----------------------------------
----+-----------+-------------+
00035   *     | EEPROM_I2C_SDA_PIN/ SDA
    |   SDA     |    5        |
```

```
00036   *     | EEPROM_I2C_SCL_PIN/ SCL
    |   SCL     |    6         |
00037   *     | .
    |   VDD     |    7 (3.3V) |
00038   *     | .
    |   GND     |    8 (0 V)  |
00039   *     +-----------------------------------
----+-----------+-------------+
00040   *
00041   ******************************************
**********************************
00042   * @attention
00043   *
00044   * <h2><center>&copy; COPYRIGHT(c) 2015 STM
icroelectronics</center></h2>
00045   *
00046   * Redistribution and use in source and bin
ary forms, with or without modification,
00047   * are permitted provided that the followin
g conditions are met:
00048   *   1. Redistributions of source code must
 retain the above copyright notice,
00049   *      this list of conditions and the fol
lowing disclaimer.
00050   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00051   *      this list of conditions and the fol
lowing disclaimer in the documentation
00052   *      and/or other materials provided wit
h the distribution.
00053   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00054   *      may be used to endorse or promote p
roducts derived from this software
00055   *      without specific prior written perm
ission.
00056   *
```

```
00057   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00058   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
00059   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00060   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00061   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00062   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
00063   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00064   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00065   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00066   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00067   *
00068   ****************************************
**********************************
00069   */
00070
00071 /* Includes ------------------------------
-----------------------------------*/
00072 #include "stm3210c_eval_eeprom.h"
00073
00074 /** @addtogroup BSP
00075   * @{
00076   */
00077
00078 /** @addtogroup STM3210C_EVAL
00079   * @{
00080   */
00081
```

```
00082 /** @addtogroup STM3210C_EVAL_EEPROM
00083   * @brief      This file includes the I2C a
nd SPI EEPROM driver
00084   *             of STM3210C-EVAL board.
00085   * @{
00086   */
00087
00088 /** @defgroup STM3210C_EVAL_EEPROM_Private_T
ypes Private Types
00089   * @{
00090   */
00091 /**
00092   * @}
00093   */
00094
00095
00096 /** @defgroup STM3210C_EVAL_EEPROM_Private_D
efines Private Defines
00097   * @{
00098   */
00099 /**
00100   * @}
00101   */
00102
00103
00104 /** @defgroup STM3210C_EVAL_EEPROM_Private_M
acros Private Macros
00105   * @{
00106   */
00107 /**
00108   * @}
00109   */
00110
00111
00112 /** @defgroup STM3210C_EVAL_EEPROM_Private_V
ariables Private Variables
00113   * @{
```

```
00114   */
00115 __IO uint16_t  EEPROMAddress = 0;
00116 __IO uint16_t  EEPROMPageSize = 0;
00117 __IO uint16_t  EEPROMDataRead = 0;
00118 __IO uint8_t   EEPROMDataWrite = 0;
00119
00120 static EEPROM_DrvTypeDef *EEPROM_SelectedDev
ice = 0;
00121 /**
00122   * @}
00123   */
00124
00125
00126 /** @defgroup STM3210C_EVAL_EEPROM_Private_F
unction_Prototypes Private Function Prototypes
00127   * @{
00128   */
00129 static uint32_t EEPROM_I2C_Init(void);
00130 static uint32_t EEPROM_I2C_ReadBuffer(uint8_
t* pBuffer, uint16_t ReadAddr, uint32_t* NumByteTo
Read);
00131 static uint32_t EEPROM_I2C_WritePage(uint8_t
* pBuffer, uint16_t WriteAddr, uint32_t* NumByteTo
Write);
00132 static uint32_t EEPROM_I2C_WaitEepromStandby
State(void);
00133
00134 /* EEPROM I2C driver typedef */
00135 EEPROM_DrvTypeDef EEPROM_I2C_Drv =
00136 {
00137   EEPROM_I2C_Init,
00138   EEPROM_I2C_ReadBuffer,
00139   EEPROM_I2C_WritePage
00140 };
00141
00142 /**
00143   * @}
```

```
00144   */
00145
00146 /** @defgroup STM3210C_EVAL_EEPROM_Exported_
Functions Exported Functions
00147   * @{
00148   */
00149
00150 /**
00151   * @brief  Initializes peripherals used by
the EEPROM device selected.
00152   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00153   *         different from EEPROM_OK (0)
00154   */
00155 uint32_t BSP_EEPROM_Init(void)
00156 {
00157   if(EEPROM_SelectedDevice->Init != 0)
00158   {
00159     return (EEPROM_SelectedDevice->Init());
00160   }
00161   else
00162   {
00163     return EEPROM_FAIL;
00164   }
00165 }
00166
00167 /**
00168   * @brief  Select the EEPROM device to comm
unicate.
00169   * @param  DeviceID: Specifies the EEPROM d
evice to be selected.
00170   *   This parameter can be one of following
 parameters:
00171   *     @arg BSP_EEPROM_M24C64_32
00172   *     @arg BSP_EEPROM_M24C08
00173   *
00174   * @retval EEPROM_OK (0) if operation is co
```

```
rrectly performed, else return value
00175   *         different from EEPROM_OK (0)
00176   */
00177 void BSP_EEPROM_SelectDevice(uint8_t DeviceI
D)
00178 {
00179   switch(DeviceID)
00180   {
00181   case BSP_EEPROM_M24C64_32:
00182   case BSP_EEPROM_M24C08:
00183     EEPROM_SelectedDevice = &EEPROM_I2C_Drv;
00184     break;
00185
00186   default:
00187     break;
00188   }
00189 }
00190
00191 /**
00192   * @brief  Reads a block of data from the E
EPROM device selected.
00193   * @param  pBuffer : pointer to the buffer
that receives the data read from
00194   *         the EEPROM.
00195   * @param  ReadAddr : EEPROM's internal add
ress to start reading from.
00196   * @param  NumByteToRead : pointer to the v
ariable holding number of bytes to
00197   *         be read from the EEPROM.
00198   *
00199   *        @note The variable pointed by Num
ByteToRead is reset to 0 when all the
00200   *              data are read from the EEPR
OM. Application should monitor this
00201   *              variable in order know when
 the transfer is complete.
00202   *
```

```
00203   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00204   *         different from EEPROM_OK (0) or
the timeout user callback.
00205   */
00206 uint32_t BSP_EEPROM_ReadBuffer(uint8_t* pBuf
fer, uint16_t ReadAddr, uint32_t* NumByteToRead)
00207 {
00208   if(EEPROM_SelectedDevice->ReadBuffer != 0)
00209   {
00210     return (EEPROM_SelectedDevice->ReadBuffer
(pBuffer, ReadAddr, NumByteToRead));
00211   }
00212   else
00213   {
00214     return EEPROM_FAIL;
00215   }
00216 }
00217
00218 /**
00219   * @brief  Writes buffer of data to the EEP
ROM device selected.
00220   * @param  pBuffer : pointer to the buffer
 containing the data to be written
00221   *         to the EEPROM.
00222   * @param  WriteAddr : EEPROM's internal ad
dress to write to.
00223   * @param  NumByteToWrite : number of bytes
 to write to the EEPROM.
00224   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00225   *         different from EEPROM_OK (0) or
the timeout user callback.
00226   */
00227 uint32_t BSP_EEPROM_WriteBuffer(uint8_t* pBu
ffer, uint16_t WriteAddr, uint32_t NumByteToWrite)
00228 {
```

```
00229   uint16_t numofpage = 0, numofsingle = 0, c
ount = 0;
00230   uint16_t addr = 0;
00231   uint32_t dataindex = 0;
00232   uint32_t status = EEPROM_OK;
00233
00234   addr = WriteAddr % EEPROMPageSize;
00235   count = EEPROMPageSize - addr;
00236   numofpage =  NumByteToWrite / EEPROMPageSi
ze;
00237   numofsingle = NumByteToWrite % EEPROMPageS
ize;
00238
00239   if(EEPROM_SelectedDevice->WritePage == 0)
00240   {
00241     return EEPROM_FAIL;
00242   }
00243
00244   /*!< If WriteAddr is EEPROM_PAGESIZE align
ed  */
00245   if(addr == 0)
00246   {
00247     /*!< If NumByteToWrite < EEPROM_PAGESIZE
 */
00248     if(numofpage == 0)
00249     {
00250       /* Store the number of data to be writ
ten */
00251       dataindex = numofsingle;
00252       /* Start writing data */
00253       status = EEPROM_SelectedDevice->WriteP
age(pBuffer, WriteAddr, (uint32_t*)(&dataindex));
00254       if (status != EEPROM_OK)
00255       {
00256         return status;
00257       }
00258     }
```

```
00259     /*!< If NumByteToWrite > EEPROM_PAGESIZE
 */
00260     else
00261     {
00262       while(numofpage--)
00263       {
00264         /* Store the number of data to be wr
itten */
00265         dataindex = EEPROMPageSize;
00266         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00267         if (status != EEPROM_OK)
00268         {
00269           return status;
00270         }
00271
00272         WriteAddr +=  EEPROMPageSize;
00273         pBuffer += EEPROMPageSize;
00274       }
00275
00276       if(numofsingle!=0)
00277       {
00278         /* Store the number of data to be wr
itten */
00279         dataindex = numofsingle;
00280         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00281         if (status != EEPROM_OK)
00282         {
00283           return status;
00284         }
00285           }
00286         }
00287     }
00288   /*!< If WriteAddr is not EEPROM_PAGESIZE a
```

```
ligned  */
00289   else
00290   {
00291     /*!< If NumByteToWrite < EEPROM_PAGESIZE
 */
00292     if(numofpage== 0)
00293     {
00294       /*!< If the number of data to be writt
en is more than the remaining space
00295       in the current page: */
00296       if (NumByteToWrite > count)
00297       {
00298         /* Store the number of data to be wr
itten */
00299         dataindex = count;
00300         /*!< Write the data contained in sam
e page */
00301         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00302         if (status != EEPROM_OK)
00303         {
00304           return status;
00305         }
00306
00307         /* Store the number of data to be wr
itten */
00308         dataindex = (NumByteToWrite - count)
;
00309         /*!< Write the remaining data in the
 following page */
00310         status = EEPROM_SelectedDevice->Writ
ePage((uint8_t*)(pBuffer + count), (WriteAddr + co
unt), (uint32_t*)(&dataindex));
00311         if (status != EEPROM_OK)
00312         {
00313           return status;
```

```
00314         }
00315       }
00316       else
00317       {
00318         /* Store the number of data to be wr
itten */
00319         dataindex = numofsingle;
00320         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00321         if (status != EEPROM_OK)
00322         {
00323           return status;
00324         }
00325         }
00326       }
00327     /*!< If NumByteToWrite > EEPROM_PAGESIZE
 */
00328     else
00329     {
00330       NumByteToWrite -= count;
00331       numofpage =  NumByteToWrite / EEPROMPa
geSize;
00332       numofsingle = NumByteToWrite % EEPROMP
ageSize;
00333
00334       if(count != 0)
00335       {
00336         /* Store the number of data to be wr
itten */
00337         dataindex = count;
00338         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00339         if (status != EEPROM_OK)
00340         {
00341           return status;
```

```
00342         }
00343         WriteAddr += count;
00344         pBuffer += count;
00345       }
00346
00347       while(numofpage--)
00348       {
00349         /* Store the number of data to be wr
itten */
00350         dataindex = EEPROMPageSize;

00351         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00352         if (status != EEPROM_OK)
00353         {
00354           return status;
00355         }
00356         WriteAddr +=  EEPROMPageSize;
00357         pBuffer += EEPROMPageSize;
00358       }
00359       if(numofsingle != 0)
00360       {
00361         /* Store the number of data to be wr
itten */
00362         dataindex = numofsingle;
00363         status = EEPROM_SelectedDevice->Writ
ePage(pBuffer, WriteAddr, (uint32_t*)(&dataindex))
;
00364         if (status != EEPROM_OK)
00365         {
00366           return status;
00367         }
00368       }
00369     }
00370   }
00371
```

```
00372   /* If all operations OK, return EEPROM_OK
(0) */
00373   return EEPROM_OK;
00374 }
00375
00376 /**
00377   * @brief  Basic management of the timeout
situation.
00378   * @retval None.
00379   */
00380 __weak void BSP_EEPROM_TIMEOUT_UserCallback(
void)
00381 {
00382 }
00383 /**
00384   * @}
00385   */
00386
00387 /** @addtogroup STM3210C_EVAL_EEPROM_Private
_Function_Prototypes
00388   * @{
00389   */
00390
00391 /**
00392   * @brief  Initializes peripherals used by
the I2C EEPROM driver.
00393   * @note There are 2 different versions of
M24CXX (08 or 32 or 64).
00394   *             Then try to connect on 1st o
ne (EEPROM_I2C_ADDRESS_A01)
00395   *             and if problem, check the 2n
d one (EEPROM_I2C_ADDRESS_A02)
00396   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00397   *         different from EEPROM_OK (0)
00398   */
00399 static uint32_t EEPROM_I2C_Init(void)
```

```
00400 {
00401   EEPROM_I2C_IO_Init();
00402
00403   /*Select the EEPROM address for M24C32 or
M24C64 and check if OK*/
00404   EEPROMAddress = EEPROM_ADDRESS_M24C64_32;
00405   EEPROMPageSize = EEPROM_PAGESIZE_M24C64_32
;
00406   if (EEPROM_I2C_IO_IsDeviceReady(EEPROMAddr
ess, EEPROM_MAX_TRIALS) != HAL_OK)
00407   {
00408     EEPROMPageSize = EEPROM_PAGESIZE_M24C08;
00409     /*Select the EEPROM address for M24C08 (
BLOCK0) and check if OK*/
00410     EEPROMAddress = EEPROM_ADDRESS_M24C08_BL
OCK0;
00411     if (EEPROM_I2C_IO_IsDeviceReady(EEPROMAd
dress, EEPROM_MAX_TRIALS) != HAL_OK)
00412     {
00413       /*Select the EEPROM address for M24C08
 (BLOCK1) and check if OK*/
00414       EEPROMAddress = EEPROM_ADDRESS_M24C08_
BLOCK1;
00415       if (EEPROM_I2C_IO_IsDeviceReady(EEPROM
Address, EEPROM_MAX_TRIALS) != HAL_OK)
00416       {
00417         /*Select the EEPROM address for M24C
08 (BLOCK2) and check if OK*/
00418         EEPROMAddress = EEPROM_ADDRESS_M24C0
8_BLOCK2;
00419         if (EEPROM_I2C_IO_IsDeviceReady(EEPR
OMAddress, EEPROM_MAX_TRIALS) != HAL_OK)
00420         {
00421           /*Select the EEPROM address for M2
4C08 (BLOCK3) and check if OK*/
00422           EEPROMAddress = EEPROM_ADDRESS_M24
C08_BLOCK3;
```

```
00423           if (EEPROM_I2C_IO_IsDeviceReady(EE
PROMAddress, EEPROM_MAX_TRIALS) != HAL_OK)
00424           {
00425             return EEPROM_FAIL;
00426           }
00427         }
00428       }
00429     }
00430   }
00431
00432   return EEPROM_OK;
00433 }
00434
00435 /**
00436   * @brief  Reads a block of data from the I
2C EEPROM.
00437   * @param  pBuffer : pointer to the buffer
that receives the data read from
00438   *         the EEPROM.
00439   * @param  ReadAddr : EEPROM's internal add
ress to start reading from.
00440   * @param  NumByteToRead : pointer to the v
ariable holding number of bytes to
00441   *         be read from the EEPROM.
00442   *
00443   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00444   *         different from EEPROM_OK (0) or
the timeout user callback.
00445   */
00446 static uint32_t EEPROM_I2C_ReadBuffer(uint8_
t* pBuffer, uint16_t ReadAddr, uint32_t* NumByteTo
Read)
00447 {
00448   uint32_t buffersize = *NumByteToRead;
00449
00450   if (EEPROM_I2C_IO_ReadData(EEPROMAddress,
```

```
ReadAddr, pBuffer, buffersize) != HAL_OK)
00451   {
00452     return EEPROM_FAIL;
00453   }
00454
00455   /* If all operations OK, return EEPROM_OK
(0) */
00456   return EEPROM_OK;
00457 }
00458
00459 /**
00460   * @brief  Writes more than one byte to the
 EEPROM with a single WRITE cycle.
00461   *
00462   * @note   The number of bytes (combined to
 write start address) must not
00463   *         cross the EEPROM page boundary.
This function can only write into
00464   *         the boundaries of an EEPROM page.

00465   *         This function doesn't check on b
oundaries condition (in this driver
00466   *         the function BSP_EEPROM_WriteBuf
fer() which calls EEPROM_WritePage() is
00467   *         responsible of checking on Page
boundaries).
00468   *
00469   * @param  pBuffer : pointer to the buffer
containing the data to be written to
00470   *         the EEPROM.
00471   * @param  WriteAddr : EEPROM's internal ad
dress to write to.
00472   * @param  NumByteToWrite : pointer to the
variable holding number of bytes to
00473   *         be written into the EEPROM.
00474   *
00475   *       @note The variable pointed by Num
```

```
ByteToWrite is reset to 0 when all the
00476   *               data are written to the EEP
ROM. Application should monitor this
00477   *               variable in order know when
 the transfer is complete.
00478   *
00479   *
00480   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00481   *         different from EEPROM_OK (0) or
the timeout user callback.
00482   */
00483 static uint32_t EEPROM_I2C_WritePage(uint8_t
* pBuffer, uint16_t WriteAddr, uint32_t* NumByteTo
Write)
00484 {
00485   uint32_t buffersize = *NumByteToWrite;
00486
00487   if (EEPROM_I2C_IO_WriteData(EEPROMAddress,
 WriteAddr, pBuffer, buffersize) != HAL_OK)
00488   {
00489     return EEPROM_FAIL;
00490   }
00491
00492   /* Wait for EEPROM Standby state */
00493   if (EEPROM_I2C_WaitEepromStandbyState() !=
EEPROM_OK)
00494   {
00495     return EEPROM_FAIL;
00496   }
00497
00498   return EEPROM_OK;
00499 }
00500
00501 /**
00502   * @brief  Wait for EEPROM I2C Standby stat
e.
```

```
00503   *
00504   * @note  This function allows to wait and
check that EEPROM has finished the
00505   *        last operation. It is mostly used
 after Write operation: after receiving
00506   *        the buffer to be written, the EEP
ROM may need additional time to actually
00507   *        perform the write operation. Duri
ng this time, it doesn't answer to
00508   *        I2C packets addressed to it. Once
 the write operation is complete
00509   *        the EEPROM responds to its addres
s.
00510   *
00511   * @retval EEPROM_OK (0) if operation is co
rrectly performed, else return value
00512   *         different from EEPROM_OK (0) or
the timeout user callback.
00513   */
00514 static uint32_t EEPROM_I2C_WaitEepromStandby
State(void)
00515 {
00516     /* Check if the maximum allowed number o
f trials has bee reached */
00517   if (EEPROM_I2C_IO_IsDeviceReady(EEPROMAddr
ess, EEPROM_MAX_TRIALS) != HAL_OK)
00518   {
00519     /* If the maximum number of trials has b
een reached, exit the function */
00520     BSP_EEPROM_TIMEOUT_UserCallback();
00521     return EEPROM_TIMEOUT;
00522   }
00523   return EEPROM_OK;
00524 }
00525
00526 /**
00527   * @}
```

```
00528   */
00529
00530 /**
00531   * @}
00532   */
00533
00534 /**
00535   * @}
00536   */
00537
00538 /**
00539   * @}
00540   */
00541
00542 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
00543
```

STM3210C_EVAL BSP User Manual

Private Defines

STM3210C-EVAL Common

Defines

#define	**START_BYTE** 0x70
#define	**SET_INDEX** 0x00
#define	**READ_STATUS** 0x01
#define	**LCD_WRITE_REG** 0x02
#define	**LCD_READ_REG** 0x03
#define	**SD_DUMMY_BYTE** 0xFF
#define	**SD_NO_RESPONSE_EXPECTED** 0x80
#define	**__STM3210C_EVAL_BSP_VERSION_MAIN** (0x06) STM3210C EVAL BSP Driver version number V6.0.1.
#define	**__STM3210C_EVAL_BSP_VERSION_SUB1** (0x00)
#define	**__STM3210C_EVAL_BSP_VERSION_SUB2** (0x01)
#define	**__STM3210C_EVAL_BSP_VERSION_RC** (0x00)
#define	**__STM3210C_EVAL_BSP_VERSION**
#define	**TFT_LCD_BASE** ((uint32_t)(0x60000000 | 0x0C000000))
#define	**TFT_LCD** ((**TFT_LCD_TypeDef** *) **TFT_LCD_BASE**)

Define Documentation

#define __STM3210C_EVAL_BSP_VERSION

Value:

```
((__STM3210C_EVAL_BSP_VERSION_MAIN << 24)\
                                                    |(
__STM3210C_EVAL_BSP_VERSION_SUB1 << 16)\
                                                    |(
__STM3210C_EVAL_BSP_VERSION_SUB2 << 8 )\
                                                    |(
__STM3210C_EVAL_BSP_VERSION_RC))
```

Definition at line **92** of file **stm3210c_eval.c**.

Referenced by **BSP_GetVersion()**.

#define __STM3210C_EVAL_BSP_VERSION_MAIN (0x06)

STM3210C EVAL BSP Driver version number V6.0.1.

[31:24] main version

Definition at line **88** of file **stm3210c_eval.c**.

#define __STM3210C_EVAL_BSP_VERSION_RC (0x00)

[7:0] release candidate

Definition at line **91** of file **stm3210c_eval.c**.

#define __STM3210C_EVAL_BSP_VERSION_SUB1 (0x00)

[23:16] sub1 version

Definition at line **89** of file **stm3210c_eval.c**.

#define __STM3210C_EVAL_BSP_VERSION_SUB2 (0x01)

[15:8] sub2 version

Definition at line **90** of file **stm3210c_eval.c**.

#define LCD_READ_REG 0x03

Definition at line **79** of file **stm3210c_eval.c**.

Referenced by **LCD_IO_ReadData()**.

#define LCD_WRITE_REG 0x02

Definition at line **78** of file **stm3210c_eval.c**.

Referenced by **LCD_IO_WriteMultipleData()**.

#define READ_STATUS 0x01

Definition at line **77** of file **stm3210c_eval.c**.

#define SD_DUMMY_BYTE 0xFF

Definition at line **82** of file **stm3210c_eval.c**.

Referenced by **SD_IO_Init()**, and **SD_IO_WriteDummy()**.

#define SD_NO_RESPONSE_EXPECTED 0x80

Definition at line 83 of file stm3210c_eval.c.

Referenced by SD_IO_WriteCmd().

#define SET_INDEX 0x00

Definition at line 76 of file stm3210c_eval.c.

Referenced by LCD_IO_WriteReg().

#define START_BYTE 0x70

Definition at line 75 of file stm3210c_eval.c.

Referenced by LCD_IO_ReadData(), LCD_IO_WriteMultipleData(), and LCD_IO_WriteReg().

#define TFT_LCD ((TFT_LCD_TypeDef *) TFT_LCD_BASE)

Definition at line 100 of file stm3210c_eval.c.

#define TFT_LCD_BASE ((uint32_t)(0x60000000 | 0x0C000000))

Definition at line 99 of file stm3210c_eval.c.

STM3210C_EVAL BSP User Manual

Private Macros

STM3210C_EVAL LCD

Defines

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

Define Documentation

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

Definition at line **102** of file **stm3210c_eval_lcd.c**.

Referenced by **BSP_LCD_DrawLine()**.

STM3210C_EVAL BSP User Manual

Exported Types

STM3210C_EVAL_ACCELEROMETER

Enumerations

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

Enumeration Type Documentation

enum ACCELERO_StatusTypeDef

Enumerator:

ACCELERO_OK

ACCELERO_ERROR

ACCELERO_TIMEOUT

Definition at line **68** of file **stm3210c_eval_accelerometer.h**.

STM3210C_EVAL BSP User Manual

Functions

Link Operations Functions

STM3210C-EVAL Common

Functions

void	**IOE_Init** (void) Initializes IOE low level.
void	**IOE_ITConfig** (void) Configures IOE low level Interrupt.
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_Delay** (uint32_t Delay) IOE delay.
void	**LCD_IO_Init** (void) Configures the LCD_SPI interface.
void	**LCD_IO_WriteMultipleData** (uint8_t *pData, uint32_t Size) Write register value.
void	**LCD_IO_WriteReg** (uint8_t Reg) register address.
uint16_t	**LCD_IO_ReadData** (uint16_t Reg) Read register value.
void	**LCD_Delay** (uint32_t Delay) Wait for loop in ms.
void	**SD_IO_Init** (void) Initializes the SD Card and put it into StandBy State (Ready for data transfer).
void	**SD_IO_WriteByte** (uint8_t Data) Write a byte on the SD.
uint8_t	**SD_IO_ReadByte** (void)

	Read a byte from the SD.
HAL_StatusTypeDef	**SD_IO_WriteCmd** (uint8_t Cmd, uint32_t Arg, uint8_t Crc, uint8_t Response) Send 5 bytes command to the SD card and get response.
HAL_StatusTypeDef	**SD_IO_WaitResponse** (uint8_t Response) Wait response from the SD card.
void	**SD_IO_WriteDummy** (void) Send dummy byte with CS High.
void	**EEPROM_I2C_IO_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_I2C_IO_WriteData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Write data to I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_I2C_IO_ReadData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Read data from I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_I2C_IO_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.
void	**TSENSOR_IO_Init** (void) Initializes peripherals used by the I2C Temperature Sensor driver.
void	**TSENSOR_IO_Write** (uint16_t DevAddress, uint8_t *pBuffer, uint8_t WriteAddr, uint16_t Length) Writes one byte to the TSENSOR.
void	**TSENSOR_IO_Read** (uint16_t DevAddress, uint8_t *pBuffer, uint8_t ReadAddr, uint16_t Length) Reads one byte from the TSENSOR.

uint16_t	**TSENSOR_IO_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if Temperature Sensor is ready for communication.
void	**ACCELERO_IO_Init** (void) Configures ACCELEROMETER SPI interface.
void	**ACCELERO_IO_ITConfig** (void) Configures ACCELERO INT2 config.
void	**ACCELERO_IO_Write** (uint8_t *pBuffer, uint8_t WriteAddr, uint16_t NumByteToWrite) Writes one byte to the ACCELEROMETER.
void	**ACCELERO_IO_Read** (uint8_t *pBuffer, uint8_t ReadAddr, uint16_t NumByteToRead) Reads a block of data from the ACCELEROMETER.
void	**AUDIO_IO_Init** (void) Initializes Audio low level.
void	**AUDIO_IO_Write** (uint8_t Addr, uint8_t Reg, uint8_t Value) Writes a single data.
uint8_t	**AUDIO_IO_Read** (uint8_t Addr, uint8_t Reg) Reads a single data.

Function Documentation

void ACCELERO_IO_Init (void)

Configures ACCELEROMETER SPI interface.

Return values:
None

Definition at line **1366** of file **stm3210c_eval.c**.

References **BSP_IO_Init()**.

void ACCELERO_IO_ITConfig (void)

Configures ACCELERO INT2 config.

EXTI0 is already used by user button so INT1 is configured here

Return values:
None

Definition at line **1378** of file **stm3210c_eval.c**.

References **BSP_IO_ConfigPin()**, and **MEMS_ALL_PINS**.

void ACCELERO_IO_Read (uint8_t * pBuffer, uint8_t ReadAddr, uint16_t NumByteToRead)

Reads a block of data from the ACCELEROMETER.

Parameters:

pBuffer : pointer to the buffer that receives the data read from the ACCELEROMETER.

ReadAddr : ACCELEROMETER's internal address to read from.

NumByteToRead : number of bytes to read from the ACCELEROMETER.

Return values:

None

Definition at line **1402** of file **stm3210c_eval.c**.

References **I2Cx_ReadBuffer()**, and **L1S302DL_I2C_ADDRESS**.

void ACCELERO_IO_Write (uint8_t * pBuffer,
uint8_t WriteAddr,
uint16_t NumByteToWrite
)

Writes one byte to the ACCELEROMETER.

Parameters:

pBuffer : pointer to the buffer containing the data to be written to the ACCELEROMETER.

WriteAddr : ACCELEROMETER's internal address to write to.

NumByteToWrite,: Number of bytes to write.

Return values:

None

Definition at line **1390** of file **stm3210c_eval.c**.

References **I2Cx_WriteBuffer()**, and **L1S302DL_I2C_ADDRESS**.

void AUDIO_IO_Init (void)

Initializes Audio low level.

Return values:

None

Definition at line **1413** of file **stm3210c_eval.c**.

References **AUDIO_RESET_PIN**, **BSP_IO_ConfigPin()**, **BSP_IO_Init()**, and **BSP_IO_WritePin()**.

uint8_t AUDIO_IO_Read (uint8_t Addr,
uint8_t Reg
)

Reads a single data.

Parameters:

Addr,: I2C address

Reg,: Reg address

Return values:

Data to be read

Definition at line **1452** of file **stm3210c_eval.c**.

References **I2Cx_ReadData()**.

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

Writes a single data.

Parameters:

Addr,: I2C address

Reg,: Reg address

Value,: Data to be written

Return values:

None

Definition at line **1441** of file **stm3210c_eval.c**.

References **I2Cx_WriteData()**.

void EEPROM_I2C_IO_Init (void)

Initializes peripherals used by the I2C EEPROM driver.

Return values:

None

Definition at line **1271** of file **stm3210c_eval.c**.

References **I2Cx_Init()**.

Referenced by **EEPROM_I2C_Init()**.

HAL_StatusTypeDef EEPROM_I2C_IO_IsDeviceReady (uint16_t De
uint32_t Tr
)

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 **1309** of file **stm3210c_eval.c**.

References **I2Cx_IsDeviceReady()**.

Referenced by **EEPROM_I2C_Init()**, and **EEPROM_I2C_WaitEepromStandbyState()**.

HAL_StatusTypeDef EEPROM_I2C_IO_ReadData (uint16_t DevAd
uint16_t MemAd
uint8_t * pBuffe
uint32_t BufferS
)

Read data from I2C EEPROM driver.

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 **1297** of file **stm3210c_eval.c**.

References I2Cx_ReadBuffer().

Referenced by EEPROM_I2C_ReadBuffer().

HAL_StatusTypeDef EEPROM_I2C_IO_WriteData (uint16_t DevAd
uint16_t MemAd
uint8_t * pBuffe
uint32_t BufferS
)

Write data to I2C EEPROM driver.

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 1284 of file stm3210c_eval.c.

References I2Cx_WriteBuffer().

Referenced by EEPROM_I2C_WritePage().

void IOE_Delay (uint32_t Delay)

IOE delay.

Parameters:

Delay,: Delay in ms

Return values:
None

Definition at line 971 of file stm3210c_eval.c.

void IOE_Init (void)

Initializes IOE low level.

Return values:
None

Definition at line 916 of file stm3210c_eval.c.

References I2Cx_Init().

void IOE_ITConfig (void)

Configures IOE low level Interrupt.

Return values:
None

Definition at line 925 of file stm3210c_eval.c.

References I2Cx_ITConfig().

uint8_t IOE_Read (uint8_t Addr, uint8_t Reg)

IOE reads single data.

Parameters:

Addr,: I2C address
Reg,: Reg address

Return values:
Read data

Definition at line **948** of file **stm3210c_eval.c**.

References **I2Cx_ReadData()**.

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

Return values:
Number of read data

Definition at line **961** of file **stm3210c_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,: Reg address

Value,: Data to be written

Return values:

None

Definition at line **937** of file **stm3210c_eval.c**.

References **I2Cx_WriteData()**.

void LCD_Delay (uint32_t Delay)

Wait for loop in ms.

Parameters:

Delay in ms.

Return values:

None

Definition at line **1103** of file **stm3210c_eval.c**.

void LCD_IO_Init (void)

Configures the LCD_SPI interface.

Return values:

None

Definition at line **985** of file **stm3210c_eval.c**.

References **LCD_CS_HIGH**, **LCD_CS_LOW**, **LCD_NCS_GPIO_CLK_ENABLE**, **LCD_NCS_GPIO_PORT**, **LCD_NCS_PIN**, and **SPIx_Init()**.

uint16_t LCD_IO_ReadData (uint16_t Reg)

Read register value.

Parameters:

Reg

Return values:

None

Definition at line **1073** of file **stm3210c_eval.c**.

References **LCD_CS_HIGH**, **LCD_CS_LOW**, **LCD_IO_WriteReg()**, **LCD_READ_REG**, **SPIx_Read()**, **SPIx_Write()**, and **START_BYTE**.

void LCD_IO_WriteMultipleData (uint8_t * pData, uint32_t Size)

Write register value.

Parameters:

pData Pointer on the register value

Size Size of byte to transmit to the register

Return values:

None

Definition at line **1012** of file **stm3210c_eval.c**.

References **heval_Spi**, **LCD_CS_HIGH**, **LCD_CS_LOW**, **LCD_WRITE_REG**, **SPIx_Write()**, and **START_BYTE**.

void LCD_IO_WriteReg (uint8_t Reg)

register address.

Parameters:

Reg

Return values:

None

Definition at line **1052** of file **stm3210c_eval.c**.

References **LCD_CS_HIGH**, **LCD_CS_LOW**, **SET_INDEX**, **SPIx_Write()**, and **START_BYTE**.

Referenced by **LCD_IO_ReadData()**.

void SD_IO_Init (void)

Initializes the SD Card and put it into StandBy State (Ready for data transfer).

Return values:

None

Definition at line **1115** of file **stm3210c_eval.c**.

References **SD_CS_GPIO_CLK_ENABLE**, **SD_CS_GPIO_PORT**, **SD_CS_HIGH**, **SD_CS_PIN**, **SD_DETECT_EXTI_IRQn**, **SD_DETECT_GPIO_CLK_ENABLE**, **SD_DETECT_GPIO_PORT**, **SD_DETECT_PIN**, **SD_DUMMY_BYTE**, **SD_IO_WriteByte()**, and **SPIx_Init()**.

Referenced by **BSP_SD_Init()**.

uint8_t SD_IO_ReadByte (void)

Read a byte from the SD.

Return values:

The received byte.

Definition at line **1172** of file **stm3210c_eval.c**.

References **SPIx_Read()**.

Referenced by **BSP_SD_ReadBlocks()**, **BSP_SD_WriteBlocks()**, **SD_GetCIDRegister()**, **SD_GetCSDRegister()**, **SD_GetDataResponse()**, and **SD_IO_WaitResponse()**.

HAL_StatusTypeDef SD_IO_WaitResponse (uint8_t Response)

Wait response from the SD card.

Parameters:

Response,: Expected response from the SD card

Return values:

HAL_StatusTypeDef HAL Status

Definition at line **1226** of file **stm3210c_eval.c**.

References **SD_IO_ReadByte()**.

Referenced by **BSP_SD_ReadBlocks()**, **SD_GetCIDRegister()**, **SD_GetCSDRegister()**, and **SD_IO_WriteCmd()**.

void SD_IO_WriteByte (uint8_t Data)

Write a byte on the SD.

Parameters:

Data,: byte to send.

Return values:

None

Definition at line **1162** of file **stm3210c_eval.c**.

References **SPIx_Write()**.

Referenced by **BSP_SD_WriteBlocks()**, **SD_GetCIDRegister()**, **SD_GetCSDRegister()**, **SD_IO_Init()**, **SD_IO_WriteCmd()**, and **SD_IO_WriteDummy()**.

HAL_StatusTypeDef SD_IO_WriteCmd (uint8_t Cmd,
uint32_t Arg,
uint8_t Crc,
uint8_t Response
)

Send 5 bytes command to the SD card and get response.

Parameters:

Cmd,:	The user expected command to send to SD card.
Arg,:	The command argument.
Crc,:	The CRC.
Response,:	Expected response from the SD card

Return values:

HAL_StatusTypeDef HAL Status

Definition at line **1191** of file **stm3210c_eval.c**.

References SD_CS_LOW, SD_IO_WaitResponse(), SD_IO_WriteByte(), and SD_NO_RESPONSE_EXPECTED.

Referenced by BSP_SD_ReadBlocks(), BSP_SD_WriteBlocks(), SD_GetCIDRegister(), SD_GetCSDRegister(), and SD_SendCmd().

void SD_IO_WriteDummy (void)

Send dummy byte with CS High.

Return values:

None

Definition at line 1254 of file stm3210c_eval.c.

References SD_CS_HIGH, SD_DUMMY_BYTE, and SD_IO_WriteByte().

Referenced by BSP_SD_ReadBlocks(), BSP_SD_WriteBlocks(), SD_GetCIDRegister(), SD_GetCSDRegister(), and SD_SendCmd().

void TSENSOR_IO_Init (void)

Initializes peripherals used by the I2C Temperature Sensor driver.

Return values:

None

Definition at line 1319 of file stm3210c_eval.c.

References I2Cx_Init().

uint16_t TSENSOR_IO_IsDeviceReady (uint16_t DevAddress, uint32_t Trials

)

Checks if Temperature Sensor is ready for communication.

Parameters:

DevAddress,:	Target device address
Trials,:	Number of trials

Return values:

HAL	status

Definition at line **1356** of file **stm3210c_eval.c**.

References **I2Cx_IsDeviceReady()**.

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

Reads one byte from the TSENSOR.

Parameters:

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

Return values:

None	

Definition at line **1345** of file **stm3210c_eval.c**.

References **I2Cx_ReadBuffer()**.

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

Writes one byte to the TSENSOR.

Parameters:

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

Return values:

None

Definition at line **1332** of file **stm3210c_eval.c**.

References **I2Cx_WriteBuffer()**.

STM3210C_EVAL BSP User Manual

Private Variables

STM3210C_EVAL_ACCELEROMETER

Variables

static ACCELERO_DrvTypeDef * **AcceleroDrv**

Variable Documentation

ACCELERO_DrvTypeDef* AcceleroDrv [static]

Definition at line 82 of file stm3210c_eval_accelerometer.c.

Referenced by BSP_ACCELERO_Click_ITClear(), BSP_ACCELERO_Click_ITConfig(), BSP_ACCELERO_GetXYZ(), BSP_ACCELERO_Init(), BSP_ACCELERO_ReadID(), and BSP_ACCELERO_Reset().

STM3210C_EVAL BSP User Manual

Defines

STM3210C_EVAL_COM

Exported Constants

Defines

#define	**COMn** 1
#define	**EVAL_COM1** USART2 Definition for COM port1, connected to USART2.
#define	**EVAL_COM1_CLK_ENABLE**() __HAL_RCC_USART2_CLK_
#define	**EVAL_COM1_CLK_DISABLE**() __HAL_RCC_USART2_CLK
#define	**AFIOCOM1_CLK_ENABLE**() __HAL_RCC_AFIO_CLK_ENA
#define	**AFIOCOM1_CLK_DISABLE**() __HAL_RCC_AFIO_CLK_DIS
#define	**EVAL_COM1_TX_PIN** GPIO_PIN_5 /* PD.05*/
#define	**EVAL_COM1_TX_GPIO_PORT** GPIOD
#define	**EVAL_COM1_TX_GPIO_CLK_ENABLE**() __HAL_RCC_GPI
#define	**EVAL_COM1_TX_GPIO_CLK_DISABLE**() __HAL_RCC_GP
#define	**EVAL_COM1_RX_PIN** GPIO_PIN_6 /* PD.06*/
#define	**EVAL_COM1_RX_GPIO_PORT** GPIOD
#define	**EVAL_COM1_RX_GPIO_CLK_ENABLE**() __HAL_RCC_GPI
#define	**EVAL_COM1_RX_GPIO_CLK_DISABLE**() __HAL_RCC_GP
#define	**EVAL_COM1_IRQn** USART2_IRQn
#define	**COMx_CLK_ENABLE**(__INDEX__) do { if((__INDEX__) == **C** **EVAL_COM1_CLK_ENABLE**();} while(0)
#define	**COMx_CLK_DISABLE**(__INDEX__) (((__INDEX__) == **COM** **EVAL_COM1_CLK_DISABLE**() : 0)
#define	**AFIOCOMx_CLK_ENABLE**(__INDEX__) do { if((__INDEX__ **AFIOCOM1_CLK_ENABLE**();} while(0)
#define	**AFIOCOMx_CLK_DISABLE**(__INDEX__) (((__INDEX__) == **AFIOCOM1_CLK_DISABLE**() : 0)
#define	**AFIOCOMx_REMAP**(__INDEX__) (((__INDEX__) == **COM1**) (AFIO_MAPR_USART2_REMAP)) : 0)
#define	**COMx_TX_GPIO_CLK_ENABLE**(__INDEX__) do { if((__IND **EVAL_COM1_TX_GPIO_CLK_ENABLE**();} while(0)
#define	**COMx_TX_GPIO_CLK_DISABLE**(__INDEX__) (((__INDEX_ **EVAL_COM1_TX_GPIO_CLK_DISABLE**() : 0)
#define	**COMx_RX_GPIO_CLK_ENABLE**(__INDEX__) do { if((__IND **EVAL_COM1_RX_GPIO_CLK_ENABLE**();} while(0)

```
#define COMx_RX_GPIO_CLK_DISABLE(__INDEX__)   (((__INDEX_
EVAL_COM1_RX_GPIO_CLK_DISABLE() : 0)
```

Define Documentation

#define AFIOCOM1_CLK_DISABLE ( ) __HAL_RCC_AFIO_CLK_D

Definition at line 266 of file stm3210c_eval.h.

#define AFIOCOM1_CLK_ENABLE ( ) __HAL_RCC_AFIO_CLK_EI

Definition at line 265 of file stm3210c_eval.h.

#define AFIOCOMx_CLK_DISABLE (__INDEX__) (((__INDEX__)

Definition at line 284 of file stm3210c_eval.h.

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

Definition at line 283 of file stm3210c_eval.h.

Referenced by BSP_COM_Init().

#define AFIOCOMx_REMAP (__INDEX__) (((__INDEX__) == COM

Definition at line 286 of file stm3210c_eval.h.

Referenced by BSP_COM_Init().

#define COMn 1

Definition at line 256 of file stm3210c_eval.h.

#define COMx_CLK_DISABLE (__INDEX__) (((__INDEX__) == C

Definition at line 281 of file stm3210c_eval.h.

#define COMx_CLK_ENABLE (__INDEX__) do { if((__INDEX__) =

Definition at line 280 of file stm3210c_eval.h.

Referenced by BSP_COM_Init().

#define COMx_RX_GPIO_CLK_DISABLE (__INDEX__) (((__INDE

Definition at line 292 of file stm3210c_eval.h.

#define COMx_RX_GPIO_CLK_ENABLE (__INDEX__) do { if((__

Definition at line 291 of file stm3210c_eval.h.

Referenced by BSP_COM_Init().

#define COMx_TX_GPIO_CLK_DISABLE (__INDEX__) (((__INDE

Definition at line 289 of file stm3210c_eval.h.

#define COMx_TX_GPIO_CLK_ENABLE (__INDEX__) do { if((__

Definition at line 288 of file stm3210c_eval.h.

Referenced by BSP_COM_Init().

#define EVAL_COM1 USART2

Definition for COM port1, connected to USART2.

Definition at line **261** of file **stm3210c_eval.h**.

#define EVAL_COM1_CLK_DISABLE ( ) __HAL_RCC_USART2_CL

Definition at line **263** of file **stm3210c_eval.h**.

#define EVAL_COM1_CLK_ENABLE ( ) __HAL_RCC_USART2_CL

Definition at line **262** of file **stm3210c_eval.h**.

#define EVAL_COM1_IRQn USART2_IRQn

Definition at line **278** of file **stm3210c_eval.h**.

#define EVAL_COM1_RX_GPIO_CLK_DISABLE ( ) __HAL_RCC_G

Definition at line **276** of file **stm3210c_eval.h**.

#define EVAL_COM1_RX_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line **275** of file **stm3210c_eval.h**.

#define EVAL_COM1_RX_GPIO_PORT GPIOD

Definition at line **274** of file **stm3210c_eval.h**.

#define EVAL_COM1_RX_PIN GPIO_PIN_6 /* PD.06*/

Definition at line **273** of file **stm3210c_eval.h**.

#define EVAL_COM1_TX_GPIO_CLK_DISABLE ( ) __HAL_RCC_G

Definition at line **271** of file **stm3210c_eval.h**.

#define EVAL_COM1_TX_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line **270** of file **stm3210c_eval.h**.

#define EVAL_COM1_TX_GPIO_PORT GPIOD

Definition at line **269** of file **stm3210c_eval.h**.

#define EVAL_COM1_TX_PIN GPIO_PIN_5 /* PD.05*/

Definition at line **268** of file **stm3210c_eval.h**.

STM3210C_EVAL BSP User Manual

AUDIO_OUT_Exported_Constants

STM3210C_EVAL_AUDIO

Defines

#define	**I2SOUT** SPI2
#define	**I2SOUT_CLK_ENABLE**() __HAL_RCC_SPI2_CLK_ENABLE
#define	**I2SOUT_SCK_SD_CLK_ENABLE**() __HAL_RCC_GPIOB_C
#define	**I2SOUT_MCK_CLK_ENABLE**() __HAL_RCC_GPIOC_CLK_
#define	**I2SOUT_WS_CLK_ENABLE**() __HAL_RCC_GPIOB_CLK_E
#define	**I2SOUT_WS_PIN** GPIO_PIN_12 /* PB.12*/
#define	**I2SOUT_SCK_PIN** GPIO_PIN_13 /* PB.13*/
#define	**I2SOUT_SD_PIN** GPIO_PIN_15 /* PB.15*/
#define	**I2SOUT_MCK_PIN** GPIO_PIN_6 /* PC.06*/
#define	**I2SOUT_SCK_SD_GPIO_PORT** GPIOB
#define	**I2SOUT_WS_GPIO_PORT** GPIOB
#define	**I2SOUT_MCK_GPIO_PORT** GPIOC
#define	**I2SOUT_DMAx_CLK_ENABLE**() __HAL_RCC_DMA1_CLK_
#define	**I2SOUT_DMAx_CHANNEL** DMA1_Channel5
#define	**I2SOUT_DMAx_IRQ** DMA1_Channel5_IRQn
#define	**I2SOUT_DMAx_PERIPH_DATA_SIZE** DMA_PDATAALIGN_
#define	**I2SOUT_DMAx_MEM_DATA_SIZE** DMA_MDATAALIGN_HA
#define	**DMA_MAX_SZE** 0xFFFF
#define	**I2SOUT_IRQHandler** DMA1_Channel5_IRQHandler
#define	**AUDIO_OUT_IRQ_PREPRIO** 5 /* Select the preemption prio
#define	**AUDIO_OK** 0
#define	**AUDIO_ERROR** 1
#define	**AUDIO_TIMEOUT** 2
#define	**INTERNAL_BUFF_SIZE** 128*DEFAULT_AUDIO_IN_FREQ/1

Define Documentation

#define AUDIO_ERROR 1

Definition at line 115 of file stm3210c_eval_audio.h.

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

#define AUDIO_OK 0

Definition at line 114 of file stm3210c_eval_audio.h.

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

#define AUDIO_OUT_IRQ_PREPRIO 5 /* Select the preemption pri

Definition at line 107 of file stm3210c_eval_audio.h.

Referenced by I2SOUT_MspInit().

#define AUDIO_TIMEOUT 2

Definition at line 116 of file stm3210c_eval_audio.h.

#define DMA_MAX_SZE 0xFFFF

Definition at line 102 of file stm3210c_eval_audio.h.

#define I2SOUT SPI2

Definition at line 83 of file stm3210c_eval_audio.h.

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

#define I2SOUT_CLK_ENABLE ( ) __HAL_RCC_SPI2_CLK_ENAB

Definition at line 84 of file stm3210c_eval_audio.h.

Referenced by I2SOUT_MspInit().

#define I2SOUT_DMAx_CHANNEL DMA1_Channel5

Definition at line 98 of file stm3210c_eval_audio.h.

Referenced by I2SOUT_MspInit().

#define I2SOUT_DMAx_CLK_ENABLE ( ) __HAL_RCC_DMA1_CL

Definition at line 97 of file stm3210c_eval_audio.h.

Referenced by I2SOUT_MspInit().

#define I2SOUT_DMAx_IRQ DMA1_Channel5_IRQn

Definition at line **99** of file **stm3210c_eval_audio.h**.

Referenced by **I2SOUT_MspInit()**.

#define I2SOUT_DMAx_MEM_DATA_SIZE DMA_MDATAALIGN_HA

Definition at line **101** of file **stm3210c_eval_audio.h**.

Referenced by **I2SOUT_MspInit()**.

#define I2SOUT_DMAx_PERIPH_DATA_SIZE DMA_PDATAALIGN_

Definition at line **100** of file **stm3210c_eval_audio.h**.

Referenced by **I2SOUT_MspInit()**.

#define I2SOUT_IRQHandler DMA1_Channel5_IRQHandler

Definition at line **104** of file **stm3210c_eval_audio.h**.

#define I2SOUT_MCK_CLK_ENABLE ( ) __HAL_RCC_GPIOC_CLI

Definition at line **86** of file **stm3210c_eval_audio.h**.

Referenced by **I2SOUT_MspInit()**.

#define I2SOUT_MCK_GPIO_PORT GPIOC

Definition at line **94** of file **stm3210c_eval_audio.h**.

Referenced by **I2SOUT_MspInit()**.

#define I2SOUT_MCK_PIN GPIO_PIN_6 /* PC.06*/

Definition at line 91 of file stm3210c_eval_audio.h.

Referenced by I2SOUT_MspInit().

#define I2SOUT_SCK_PIN GPIO_PIN_13 /* PB.13*/

Definition at line 89 of file stm3210c_eval_audio.h.

Referenced by I2SOUT_MspInit().

#define I2SOUT_SCK_SD_CLK_ENABLE ( ) __HAL_RCC_GPIOB_

Definition at line 85 of file stm3210c_eval_audio.h.

Referenced by I2SOUT_MspInit().

#define I2SOUT_SCK_SD_GPIO_PORT GPIOB

Definition at line 92 of file stm3210c_eval_audio.h.

Referenced by I2SOUT_MspInit().

#define I2SOUT_SD_PIN GPIO_PIN_15 /* PB.15*/

Definition at line 90 of file stm3210c_eval_audio.h.

Referenced by I2SOUT_MspInit().

#define I2SOUT_WS_CLK_ENABLE ( ) __HAL_RCC_GPIOB_CLK_

Definition at line **87** of file **stm3210c_eval_audio.h**.

Referenced by **I2SOUT_MspInit()**.

#define I2SOUT_WS_GPIO_PORT GPIOB

Definition at line **93** of file **stm3210c_eval_audio.h**.

Referenced by **I2SOUT_MspInit()**.

#define I2SOUT_WS_PIN GPIO_PIN_12 /* PB.12*/

Definition at line **88** of file **stm3210c_eval_audio.h**.

Referenced by **I2SOUT_MspInit()**.

#define INTERNAL_BUFF_SIZE 128*DEFAULT_AUDIO_IN_FREQ/1

Definition at line **119** of file **stm3210c_eval_audio.h**.

STM3210C_EVAL BSP User Manual

STM3210C_EVAL_COMPONENT

Exported Constants

Defines

#define	**LCD_CS_LOW**() HAL_GPIO_WritePin(**LCD_NCS_GPIO_PO** **LCD_NCS_PIN**, GPIO_PIN_RESET)
#define	**LCD_CS_HIGH**() HAL_GPIO_WritePin(**LCD_NCS_GPIO_PO** **LCD_NCS_PIN**, GPIO_PIN_SET)
#define	**LCD_NCS_PIN** GPIO_PIN_2 /* PB.02*/ LCD Control Interface pins.
#define	**LCD_NCS_GPIO_PORT** GPIOB
#define	**LCD_NCS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_CL
#define	**LCD_NCS_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_C
#define	**SD_CS_LOW**() HAL_GPIO_WritePin(**SD_CS_GPIO_PORT**, GPIO_PIN_RESET)
#define	**SD_CS_HIGH**() HAL_GPIO_WritePin(**SD_CS_GPIO_PORT**, GPIO_PIN_SET)
#define	**SD_CS_PIN** GPIO_PIN_4 /* PA.04*/ SD Control Interface pins.
#define	**SD_CS_GPIO_PORT** GPIOA
#define	**SD_CS_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOA_CLK_
#define	**SD_CS_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOA_CLK_
#define	**SD_DETECT_PIN** GPIO_PIN_0 SD Detect Interface pins.
#define	**SD_DETECT_GPIO_PORT** GPIOE
#define	**SD_DETECT_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOE_
#define	**SD_DETECT_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOE_
#define	**SD_DETECT_EXTI_IRQn** EXTI0_IRQn
#define	**AUDIO_I2C_ADDRESS** 0x94 AUDIO I2C Interface pins.

Define Documentation

#define AUDIO_I2C_ADDRESS 0x94

AUDIO I2C Interface pins.

Definition at line 435 of file stm3210c_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 LCD_CS_HIGH ( ) HAL_GPIO_WritePin(LCD_NCS_GPIO_

Definition at line 399 of file stm3210c_eval.h.

Referenced by LCD_IO_Init(), LCD_IO_ReadData(), LCD_IO_WriteMultipleData(), and LCD_IO_WriteReg().

#define LCD_CS_LOW ( ) HAL_GPIO_WritePin(LCD_NCS_GPIO_I

Definition at line 398 of file stm3210c_eval.h.

Referenced by LCD_IO_Init(), LCD_IO_ReadData(), LCD_IO_WriteMultipleData(), and LCD_IO_WriteReg().

#define LCD_NCS_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOB_

Definition at line 407 of file stm3210c_eval.h.

#define LCD_NCS_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOB_C

Definition at line 406 of file stm3210c_eval.h.

Referenced by LCD_IO_Init().

#define LCD_NCS_GPIO_PORT GPIOB

Definition at line 405 of file stm3210c_eval.h.

Referenced by LCD_IO_Init().

#define LCD_NCS_PIN GPIO_PIN_2 /* PB.02*/

LCD Control Interface pins.

Definition at line 404 of file stm3210c_eval.h.

Referenced by LCD_IO_Init().

#define SD_CS_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOA_CL

Definition at line 420 of file stm3210c_eval.h.

#define SD_CS_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOA_CLK

Definition at line 419 of file stm3210c_eval.h.

Referenced by SD_IO_Init().

#define SD_CS_GPIO_PORT GPIOA

Definition at line 418 of file stm3210c_eval.h.

Referenced by SD_IO_Init().

#define SD_CS_HIGH ( ) HAL_GPIO_WritePin(SD_CS_GPIO_POR

Definition at line 412 of file stm3210c_eval.h.

Referenced by SD_IO_Init(), and SD_IO_WriteDummy().

#define SD_CS_LOW ( ) HAL_GPIO_WritePin(SD_CS_GPIO_POR

Definition at line 411 of file stm3210c_eval.h.

Referenced by SD_IO_WriteCmd().

#define SD_CS_PIN GPIO_PIN_4 /* PA.04*/

SD Control Interface pins.

Definition at line 417 of file stm3210c_eval.h.

Referenced by SD_IO_Init().

#define SD_DETECT_EXTI_IRQn EXTI0_IRQn

Definition at line 429 of file stm3210c_eval.h.

Referenced by SD_IO_Init().

#define SD_DETECT_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIO

Definition at line **428** of file **stm3210c_eval.h**.

#define SD_DETECT_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOE

Definition at line **427** of file **stm3210c_eval.h**.

Referenced by **SD_IO_Init()**.

#define SD_DETECT_GPIO_PORT GPIOE

Definition at line **426** of file **stm3210c_eval.h**.

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

#define SD_DETECT_PIN GPIO_PIN_0

SD Detect Interface pins.

Definition at line **425** of file **stm3210c_eval.h**.

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

STM3210C_EVAL BSP User Manual

STM3210C_EVAL_BUTTON

Exported Constants

Defines

#define	**BUTTONn** 3
#define	**TAMPER_BUTTON_PIN** GPIO_PIN_13 /* PC.13*/ Tamper push-button.
#define	**TAMPER_BUTTON_GPIO_PORT** GPIOC
#define	**TAMPER_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_G
#define	**TAMPER_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_G
#define	**TAMPER_BUTTON_EXTI_IRQn** EXTI15_10_IRQn
#define	**KEY_BUTTON_PIN** GPIO_PIN_9 /* PB.09*/ Key push-button.
#define	**KEY_BUTTON_GPIO_PORT** GPIOB
#define	**KEY_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB
#define	**KEY_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_GPIO
#define	**KEY_BUTTON_EXTI_IRQn** EXTI9_5_IRQn
#define	**WAKEUP_BUTTON_PIN** GPIO_PIN_0 /* PA.00*/ Wake-up push-button.
#define	**WAKEUP_BUTTON_GPIO_PORT** GPIOA
#define	**WAKEUP_BUTTON_GPIO_CLK_ENABLE**() __HAL_RCC_G
#define	**WAKEUP_BUTTON_GPIO_CLK_DISABLE**() __HAL_RCC_G
#define	**WAKEUP_BUTTON_EXTI_IRQn** EXTI0_IRQn
#define	**BUTTONx_GPIO_CLK_ENABLE**(__BUTTON__)
#define	**BUTTONx_GPIO_CLK_DISABLE**(__BUTTON__)
#define	**JOY_SEL_PIN** (**IO2_PIN_7**) /* IO_Expander_2 */ IO Pins definition.
#define	**JOY_DOWN_PIN** (**IO2_PIN_6**) /* IO_Expander_2 */
#define	**JOY_LEFT_PIN** (**IO2_PIN_5**) /* IO_Expander_2 */
#define	**JOY_RIGHT_PIN** (**IO2_PIN_4**) /* IO_Expander_2 */
#define	**JOY_UP_PIN** (**IO2_PIN_3**) /* IO_Expander_2 */
#define	**JOY_NONE_PIN** **JOY_ALL_PINS**
#define	**JOY_ALL_PINS** (**JOY_SEL_PIN** | **JOY_DOWN_PIN** | **JOY_L** **JOY_RIGHT_PIN** | **JOY_UP_PIN**)
#define	**MEMS_INT1_PIN** (**IO1_PIN_3**) /* IO_Expander_1 */ /* Input */

```
#define MEMS_INT2_PIN   (IO1_PIN_2) /* IO_Expander_1 */ /* Input */
#define MEMS_ALL_PINS   (MEMS_INT1_PIN | MEMS_INT2_PIN)
#define AUDIO_RESET_PIN   (IO2_PIN_2) /* IO_Expander_2 */ /* Out
#define MII_INT_PIN   (IO2_PIN_0) /* IO_Expander_2 */ /* Output */
#define VBAT_DIV_PIN   (IO1_PIN_0) /* IO_Expander_1 */ /* Output */
```

Define Documentation

#define AUDIO_RESET_PIN (IO2_PIN_2) /* IO_Expander_2 */ /* Ou

Definition at line **245** of file **stm3210c_eval.h**.

Referenced by **AUDIO_IO_Init()**, and **BSP_AUDIO_OUT_Stop()**.

#define BUTTONn 3

Definition at line **191** of file **stm3210c_eval.h**.

#define BUTTONx_GPIO_CLK_DISABLE (__BUTTON__)

Value:

```
(((__BUTTON__) == BUTTON_TAMPER) TAMPER_BUTTON_GP
IO_CLK_DISABLE()  :\

((__BUTTON__) == BUTTON_KEY) KEY_BUTTON_GPIO_CLK_
DISABLE()  :\

((__BUTTON__) == BUTTON_WAKEUP) WAKEUP_BUTTON_GPI
O_CLK_DISABLE()  : 0 )
```

Definition at line **224** of file **stm3210c_eval.h**.

#define BUTTONx_GPIO_CLK_ENABLE (__BUTTON__)

Value:

```
do { if ((__BUTTON__) == BUTTON_TAMPER) TAMPER_BU
TTON_GPIO_CLK_ENABLE()  ; else \

if ((__BUTTON__) == BUTTON_KEY) KEY_BUTTON_GPIO_C
```

```
LK_ENABLE()  ; else \

if ((__BUTTON__) == BUTTON_WAKEUP) WAKEUP_BUTTON_
GPIO_CLK_ENABLE();} while(0)
```

Definition at line **220** of file **stm3210c_eval.h**.

Referenced by **BSP_PB_Init()**.

#define JOY_ALL_PINS (JOY_SEL_PIN | JOY_DOWN_PIN | JOY_L

Definition at line **238** of file **stm3210c_eval.h**.

Referenced by **BSP_JOY_GetState()**, and **BSP_JOY_Init()**.

#define JOY_DOWN_PIN (IO2_PIN_6) /* IO_Expander_2 */

Definition at line **233** of file **stm3210c_eval.h**.

Referenced by **BSP_JOY_GetState()**.

#define JOY_LEFT_PIN (IO2_PIN_5) /* IO_Expander_2 */

Definition at line **234** of file **stm3210c_eval.h**.

Referenced by **BSP_JOY_GetState()**.

#define JOY_NONE_PIN JOY_ALL_PINS

Definition at line **237** of file **stm3210c_eval.h**.

Referenced by **BSP_JOY_GetState()**.

#define JOY_RIGHT_PIN (IO2_PIN_4) /* IO_Expander_2 */

Definition at line 235 of file stm3210c_eval.h.

Referenced by BSP_JOY_GetState().

#define JOY_SEL_PIN (IO2_PIN_7) /* IO_Expander_2 */

IO Pins definition.

Definition at line 232 of file stm3210c_eval.h.

Referenced by BSP_JOY_GetState().

#define JOY_UP_PIN (IO2_PIN_3) /* IO_Expander_2 */

Definition at line 236 of file stm3210c_eval.h.

Referenced by BSP_JOY_GetState().

#define KEY_BUTTON_EXTI_IRQn EXTI9_5_IRQn

Definition at line 209 of file stm3210c_eval.h.

#define KEY_BUTTON_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPI

Definition at line 208 of file stm3210c_eval.h.

#define KEY_BUTTON_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPI(

Definition at line 207 of file stm3210c_eval.h.

#define KEY_BUTTON_GPIO_PORT GPIOB

Definition at line 206 of file stm3210c_eval.h.

#define KEY_BUTTON_PIN GPIO_PIN_9 /* PB.09*/

Key push-button.

Definition at line 205 of file stm3210c_eval.h.

#define MEMS_ALL_PINS (MEMS_INT1_PIN | MEMS_INT2_PIN)

Definition at line 243 of file stm3210c_eval.h.

Referenced by ACCELERO_IO_ITConfig().

#define MEMS_INT1_PIN (IO1_PIN_3) /* IO_Expander_1 */ /* Input

Definition at line 241 of file stm3210c_eval.h.

#define MEMS_INT2_PIN (IO1_PIN_2) /* IO_Expander_1 */ /* Input

Definition at line 242 of file stm3210c_eval.h.

#define MII_INT_PIN (IO2_PIN_0) /* IO_Expander_2 */ /* Output */

Definition at line 246 of file stm3210c_eval.h.

#define TAMPER_BUTTON_EXTI_IRQn EXTI15_10_IRQn

Definition at line **200** of file **stm3210c_eval.h**.

#define TAMPER_BUTTON_GPIO_CLK_DISABLE ( ) __HAL_RCC_

Definition at line **199** of file **stm3210c_eval.h**.

#define TAMPER_BUTTON_GPIO_CLK_ENABLE ( ) __HAL_RCC_

Definition at line **198** of file **stm3210c_eval.h**.

#define TAMPER_BUTTON_GPIO_PORT GPIOC

Definition at line **197** of file **stm3210c_eval.h**.

#define TAMPER_BUTTON_PIN GPIO_PIN_13 /* PC.13*/

Tamper push-button.

Definition at line **196** of file **stm3210c_eval.h**.

#define VBAT_DIV_PIN (IO1_PIN_0) /* IO_Expander_1 */ /* Output

Definition at line **247** of file **stm3210c_eval.h**.

#define WAKEUP_BUTTON_EXTI_IRQn EXTI0_IRQn

Definition at line **218** of file **stm3210c_eval.h**.

#define WAKEUP_BUTTON_GPIO_CLK_DISABLE ( ) __HAL_RCC

Definition at line **217** of file **stm3210c_eval.h**.

#define WAKEUP_BUTTON_GPIO_CLK_ENABLE ( ) __HAL_RCC_

Definition at line **216** of file **stm3210c_eval.h**.

#define WAKEUP_BUTTON_GPIO_PORT GPIOA

Definition at line **215** of file **stm3210c_eval.h**.

#define WAKEUP_BUTTON_PIN GPIO_PIN_0 /* PA.00*/

Wake-up push-button.

Definition at line **214** of file **stm3210c_eval.h**.

STM3210C_EVAL BSP User Manual

Private Variables

STM3210C_EVAL LCD

Variables

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

Variable Documentation

uint8_t bitmap[MAX_HEIGHT_FONT *MAX_WIDTH_FONT *2+OFFSI

Definition at line 116 of file stm3210c_eval_lcd.c.

Referenced by LCD_DrawChar().

LCD_DrawPropTypeDef DrawProp

Definition at line 111 of file stm3210c_eval_lcd.c.

LCD_DrvTypeDef* lcd_drv [static]

Definition at line 113 of file stm3210c_eval_lcd.c.

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

STM3210C_EVAL BSP User Manual

Exported Functions

STM3210C_EVAL_ACCELEROMETER

Functions

uint8_t	**BSP_ACCELERO_Init** (void) Set ACCELEROMETER Initialization.
uint8_t	**BSP_ACCELERO_ReadID** (void) Read ID of Accelerometer component.
void	**BSP_ACCELERO_Reset** (void) Reboot memory content of ACCELEROMETER.
void	**BSP_ACCELERO_Click_ITConfig** (void) Config Accelerometer click IT.
void	**BSP_ACCELERO_Click_ITClear** (void) Clear Accelerometer click IT.
void	**BSP_ACCELERO_GetXYZ** (int16_t *pDataXYZ) Get XYZ acceleration.

Function Documentation

void BSP_ACCELERO_Click_ITClear (void)

Clear Accelerometer click IT.

Return values:
None

Definition at line **198** of file **stm3210c_eval_accelerometer.c**.

References **AcceleroDrv**.

void BSP_ACCELERO_Click_ITConfig (void)

Config Accelerometer click IT.

Return values:
None

Definition at line **185** of file **stm3210c_eval_accelerometer.c**.

References **AcceleroDrv**.

void BSP_ACCELERO_GetXYZ (int16_t * pDataXYZ)

Get XYZ acceleration.

Parameters:
pDataXYZ,: angular acceleration on X/Y/Z axis

Return values:
None

Definition at line **211** of file **stm3210c_eval_accelerometer.c**.

References **AcceleroDrv**.

uint8_t BSP_ACCELERO_Init (void)

Set ACCELEROMETER Initialization.

Return values:

None

Definition at line **104** of file **stm3210c_eval_accelerometer.c**.

References **ACCELERO_ERROR**, **ACCELERO_OK**, and **AcceleroDrv**.

uint8_t BSP_ACCELERO_ReadID (void)

Read ID of Accelerometer component.

Return values:

ID

Definition at line **158** of file **stm3210c_eval_accelerometer.c**.

References **AcceleroDrv**.

void BSP_ACCELERO_Reset (void)

Reboot memory content of ACCELEROMETER.

Return values:

None

Definition at line **173** of file **stm3210c_eval_accelerometer.c**.

References **AcceleroDrv**.

STM3210C_EVAL BSP User Manual

Functions

AUDIO_OUT_Exported_Functions

STM3210C_EVAL_AUDIO

Functions

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

void	**HAL_I2S_ErrorCallback** (I2S_HandleTypeDef *hi2s) I2S error callbacks.
__weak void	**BSP_AUDIO_OUT_TransferComplete_CallBack** (void) Manages the DMA full Transfer complete event.
__weak void	**BSP_AUDIO_OUT_HalfTransfer_CallBack** (void) Manages the DMA Half Transfer complete event.
__weak void	**BSP_AUDIO_OUT_Error_CallBack** (void) Manages the DMA FIFO error event.
static void	**I2SOUT_MspInit** (void) AUDIO OUT I2S MSP Init.
static void	**I2SOUT_Init** (uint32_t AudioFreq) Initializes the Audio Codec audio interface (I2S)

Function Documentation

void BSP_AUDIO_OUT_ChangeBuffer (uint16_t * pData,
uint16_t Size
)

Sends n-Bytes on the I2S interface.

Parameters:

pData,: pointer on data address

Size,: number of data to be written

Return values:

None

Definition at line **263** of file **stm3210c_eval_audio.c**.

References **hAudioOutI2s**.

void BSP_AUDIO_OUT_Error_CallBack (void)

Manages the DMA FIFO error event.

Return values:

None

Definition at line **488** of file **stm3210c_eval_audio.c**.

Referenced by **HAL_I2S_ErrorCallback()**.

void BSP_AUDIO_OUT_HalfTransfer_CallBack (void)

Manages the DMA Half Transfer complete event.

Return values:

None

Definition at line **480** of file **stm3210c_eval_audio.c**.

Referenced by **HAL_I2S_TxHalfCpltCallback()**.

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

Configure the audio peripherals.

Parameters:

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

Return values:

0 if correct communication, else wrong communication

Definition at line **205** of file **stm3210c_eval_audio.c**.

References **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, **I2SOUT_Init()**, and **pAudioDrv**.

uint8_t BSP_AUDIO_OUT_Pause (void)

This function Pauses the audio file stream.

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

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

Return values:

AUDIO_OK	if correct communication, else wrong communication

Definition at line **276** of file **stm3210c_eval_audio.c**.

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

uint8_t BSP_AUDIO_OUT_Play (uint16_t * pBuffer,
uint32_t Size
)

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

Parameters:

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

Return values:

AUDIO_OK	if correct communication, else wrong communication

Definition at line **241** of file **stm3210c_eval_audio.c**.

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

uint8_t BSP_AUDIO_OUT_Resume (void)

This function Resumes the audio file stream.

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

Return values:

AUDIO_OK	if correct communication, else wrong communication

Definition at line **300** of file **stm3210c_eval_audio.c**.

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

void BSP_AUDIO_OUT_SetFrequency (uint32_t AudioFreq)

Update the audio frequency.

Parameters:

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

Return values:
None

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

Definition at line **418** of file **stm3210c_eval_audio.c**.

References **I2SOUT_Init()**.

uint8_t BSP_AUDIO_OUT_SetMute (uint32_t Cmd)

Enables or disables the MUTE mode by software.

Parameters:

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

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **375** of file **stm3210c_eval_audio.c**.

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

uint8_t BSP_AUDIO_OUT_SetOutputMode (uint8_t Output)

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

Note:

This function modifies a global variable of the audio codec driver: OutputDev.

Parameters:

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

OUTPUT_DEVICE_BOTH or
OUTPUT_DEVICE_AUTO

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **397** of file **stm3210c_eval_audio.c**.

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

uint8_t BSP_AUDIO_OUT_SetVolume (uint8_t Volume)

Controls the current audio volume level.

Parameters:

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

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **355** of file **stm3210c_eval_audio.c**.

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

uint8_t BSP_AUDIO_OUT_Stop (uint32_t Option)

Stops audio playing and Power down the Audio Codec.

Parameters:

Option,: could be one of the following parameters

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

Return values:

AUDIO_OK if correct communication, else wrong communication

Definition at line **323** of file **stm3210c_eval_audio.c**.

References **AUDIO_ERROR**, **AUDIO_I2C_ADDRESS**, **AUDIO_OK**, **AUDIO_RESET_PIN**, **BSP_IO_WritePin()**, **hAudioOutI2s**, and **pAudioDrv**.

void BSP_AUDIO_OUT_TransferComplete_CallBack (void)

Manages the DMA full Transfer complete event.

Return values:

None

Definition at line **472** of file **stm3210c_eval_audio.c**.

Referenced by **HAL_I2S_TxCpltCallback()**.

void HAL_I2S_ErrorCallback (I2S_HandleTypeDef * hi2s)

I2S error callbacks.

Parameters:

hi2s,: I2S handle

Return values:

None

Definition at line **458** of file **stm3210c_eval_audio.c**.

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

void HAL_I2S_TxCpltCallback (I2S_HandleTypeDef * hi2s)

Tx Transfer completed callbacks.

Parameters:

hi2s,: I2S handle

Return values:

None

Definition at line **429** of file **stm3210c_eval_audio.c**.

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

void HAL_I2S_TxHalfCpltCallback (I2S_HandleTypeDef * hi2s)

Tx Transfer Half completed callbacks.

Parameters:

hi2s,: I2S handle

Return values:

None

Definition at line **443** of file **stm3210c_eval_audio.c**.

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

`[static]`

static void I2SOUT_Init (uint32_t AudioFreq)

Initializes the Audio Codec audio interface (I2S)

Parameters:

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

Definition at line **564** of file **stm3210c_eval_audio.c**.

References **hAudioOutI2s**, **I2SOUT**, and **I2SOUT_MspInit()**.

Referenced by **BSP_AUDIO_OUT_Init()**, and **BSP_AUDIO_OUT_SetFrequency()**.

static void I2SOUT_MspInit (void) [static]

AUDIO OUT I2S MSP Init.

Return values:

None

Definition at line **500** of file **stm3210c_eval_audio.c**.

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

Referenced by **I2SOUT_Init()**.

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP
User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

Exported Functions

STM3210C-EVAL Common

Functions

uint32_t	**BSP_GetVersion** (void) This method returns the STM3210C EVAL BSP Driver revision.
void	**BSP_LED_Init** (**Led_TypeDef** Led) Configures LED GPIO.
void	**BSP_LED_On** (**Led_TypeDef** Led) Turns selected LED On.
void	**BSP_LED_Off** (**Led_TypeDef** Led) Turns selected LED Off.
void	**BSP_LED_Toggle** (**Led_TypeDef** Led) Toggles the selected LED.
void	**BSP_PB_Init** (**Button_TypeDef** Button, **ButtonMode_TypeDef** Button_Mode) Configures push button GPIO and EXTI Line.
uint32_t	**BSP_PB_GetState** (**Button_TypeDef** Button) Returns the selected button state.
uint8_t	**BSP_JOY_Init** (**JOYMode_TypeDef** Joy_Mode) Configures joystick GPIO and EXTI modes.
JOYState_TypeDef	**BSP_JOY_GetState** (void) Returns the current joystick status.
void	**BSP_COM_Init** (**COM_TypeDef** COM, UART_HandleTypeDef *huart) Configures COM port.

Function Documentation

void BSP_COM_Init (COM_TypeDef COM,
UART_HandleTypeDef * huart
)

Configures COM port.

Parameters:

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

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

Return values:

None

Definition at line **479** of file **stm3210c_eval.c**.

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

uint32_t BSP_GetVersion (void)

This method returns the STM3210C EVAL BSP Driver revision.

Return values:

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

Definition at line **250** of file **stm3210c_eval.c**.

References **__STM3210C_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 **430** of file **stm3210c_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 **402** of file **stm3210c_eval.c**.

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

void BSP_LED_Init (Led_TypeDef Led)

Configures LED GPIO.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Return values:

None

Definition at line **265** of file **stm3210c_eval.c**.

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

void BSP_LED_Off (Led_TypeDef Led)

Turns selected LED Off.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Return values:

None

Definition at line **308** of file **stm3210c_eval.c**.

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

void BSP_LED_On (Led_TypeDef Led)

Turns selected LED On.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Return values:

None

Definition at line **293** of file **stm3210c_eval.c**.

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

void BSP_LED_Toggle (Led_TypeDef Led)

Toggles the selected LED.

Parameters:

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

- LED1
- LED2
- LED3
- LED4

Return values:

None

Definition at line **323** of file **stm3210c_eval.c**.

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

uint32_t BSP_PB_GetState (Button_TypeDef Button)

Returns the selected button state.

Parameters:

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

- BUTTON_TAMPER: Key/Tamper Push Button

Return values:

Button state

Definition at line **387** of file **stm3210c_eval.c**.

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

void BSP_PB_Init (Button_TypeDef Button,
ButtonMode_TypeDef Button_Mode

)

Configures push button GPIO and EXTI Line.

Parameters:

Button,: Button to be configured. This parameter can be one of the following values:
- BUTTON_WAKEUP: Wakeup Push Button
- BUTTON_TAMPER: Tamper Push Button
- BUTTON_KEY: Key Push Button

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

Return values:

None

Definition at line **342** of file **stm3210c_eval.c**.

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

STM3210C_EVAL BSP User Manual

Exported Functions

STM3210C_EVAL_EEPROM

Functions

uint32_t	**BSP_EEPROM_Init** (void) Initializes peripherals used by the EEPROM device selected.
void	**BSP_EEPROM_SelectDevice** (uint8_t DeviceID) Select the EEPROM device to communicate.
uint32_t	**BSP_EEPROM_ReadBuffer** (uint8_t *pBuffer, uint16_t ReadAddr, uint32_t *NumByteToRead) Reads a block of data from the EEPROM device selected.
uint32_t	**BSP_EEPROM_WriteBuffer** (uint8_t *pBuffer, uint16_t WriteAddr, uint32_t NumByteToWrite) Writes buffer of data to the EEPROM device selected.
__weak void	**BSP_EEPROM_TIMEOUT_UserCallback** (void) Basic management of the timeout situation.
void	**EEPROM_I2C_IO_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_I2C_IO_WriteData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Write data to I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_I2C_IO_ReadData** (uint16_t DevAddress, uint16_t MemAddress, uint8_t *pBuffer, uint32_t BufferSize) Read data from I2C EEPROM driver.
HAL_StatusTypeDef	**EEPROM_I2C_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 EEPROM device selected.

Return values:

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

Definition at line **155** of file **stm3210c_eval_eeprom.c**.

References **EEPROM_FAIL**, and **EEPROM_DrvTypeDef::Init**.

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

Reads a block of data from the EEPROM device selected.

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 **206** of file **stm3210c_eval_eeprom.c**.

References **EEPROM_FAIL**, and **EEPROM_DrvTypeDef::ReadBuffer**.

void BSP_EEPROM_SelectDevice (uint8_t DeviceID)

Select the EEPROM device to communicate.

Parameters:

DeviceID,: Specifies the EEPROM device to be selected. This parameter can be one of following parameters:

- BSP_EEPROM_M24C64_32
- BSP_EEPROM_M24C08

Return values:

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

Definition at line **177** of file **stm3210c_eval_eeprom.c**.

References **BSP_EEPROM_M24C08**, **BSP_EEPROM_M24C64_32**, and **EEPROM_I2C_Drv**.

void BSP_EEPROM_TIMEOUT_UserCallback (void)

Basic management of the timeout situation.

Return values:

None.

Definition at line 380 of file stm3210c_eval_eeprom.c.

Referenced by EEPROM_I2C_WaitEepromStandbyState().

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

Writes buffer of data to the EEPROM device selected.

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.

< If NumByteToWrite < EEPROM_PAGESIZE

< If NumByteToWrite < EEPROM_PAGESIZE

< If the number of data to be written is more than the remaining space in the current page:

< Write the data contained in same page

< Write the remaining data in the following page

Definition at line 227 of file stm3210c_eval_eeprom.c.

References EEPROM_FAIL, EEPROM_OK, EEPROMPageSize, and

EEPROM_DrvTypeDef::WritePage.

void EEPROM_I2C_IO_Init (void)

Initializes peripherals used by the I2C EEPROM driver.

Return values:
None

Definition at line **1271** of file **stm3210c_eval.c**.

References **I2Cx_Init()**.

Referenced by **EEPROM_I2C_Init()**.

HAL_StatusTypeDef EEPROM_I2C_IO_IsDeviceReady (uint16_t De, uint32_t Tr)

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 **1309** of file **stm3210c_eval.c**.

References **I2Cx_IsDeviceReady()**.

Referenced by **EEPROM_I2C_Init()**, and **EEPROM_I2C_WaitEepromStandbyState()**.

HAL_StatusTypeDef EEPROM_I2C_IO_ReadData (uint16_t DevAd
uint16_t MemAd
uint8_t * pBuffe
uint32_t BufferS
)

Read data from I2C EEPROM driver.

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 **1297** of file **stm3210c_eval.c**.

References **I2Cx_ReadBuffer()**.

Referenced by **EEPROM_I2C_ReadBuffer()**.

HAL_StatusTypeDef EEPROM_I2C_IO_WriteData (uint16_t DevAd
uint16_t MemAd
uint8_t * pBuffe
uint32_t BufferS
)

Write data to I2C EEPROM driver.

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 **1284** of file **stm3210c_eval.c**.

References **I2Cx_WriteBuffer()**.

Referenced by **EEPROM_I2C_WritePage()**.

STM3210C_EVAL BSP User Manual

Defines

Exported Constants

STM3210C_EVAL_EEPROM

Defines

#define	**EEPROM_ADDRESS_M24C64_32** 0xA0 /* Support the devices: M24C32 and M24C64 */
#define	**EEPROM_ADDRESS_M24C08_BLOCK0** 0xA0
#define	**EEPROM_ADDRESS_M24C08_BLOCK1** 0xA2
#define	**EEPROM_ADDRESS_M24C08_BLOCK2** 0xA4
#define	**EEPROM_ADDRESS_M24C08_BLOCK3** 0xA6
#define	**EEPROM_PAGESIZE_M24C64_32** 32 /* Support the devices: M24C32 and M24C64 */
#define	**EEPROM_PAGESIZE_M24C08** 16 /* Support the device: M24C08. */
#define	**EEPROM_OK** 0
#define	**EEPROM_FAIL** 1
#define	**EEPROM_TIMEOUT** 2
#define	**BSP_EEPROM_M24C64_32** 1 /* RF I2C EEPROM M24C32 and M24C64 */
#define	**BSP_EEPROM_M24C08** 2 /* RF I2C EEPROM M24C08 */
#define	**EEPROM_MAX_TRIALS** 300

Define Documentation

#define BSP_EEPROM_M24C08 2 /* RF I2C EEPROM M24C08 */

Definition at line 97 of file stm3210c_eval_eeprom.h.

Referenced by BSP_EEPROM_SelectDevice().

#define BSP_EEPROM_M24C64_32 1 /* RF I2C EEPROM M24C32 a

Definition at line 96 of file stm3210c_eval_eeprom.h.

Referenced by BSP_EEPROM_SelectDevice().

#define EEPROM_ADDRESS_M24C08_BLOCK0 0xA0

Definition at line 82 of file stm3210c_eval_eeprom.h.

Referenced by EEPROM_I2C_Init().

#define EEPROM_ADDRESS_M24C08_BLOCK1 0xA2

Definition at line 83 of file stm3210c_eval_eeprom.h.

Referenced by EEPROM_I2C_Init().

#define EEPROM_ADDRESS_M24C08_BLOCK2 0xA4

Definition at line 84 of file stm3210c_eval_eeprom.h.

Referenced by EEPROM_I2C_Init().

#define EEPROM_ADDRESS_M24C08_BLOCK3 0xA6

Definition at line 85 of file stm3210c_eval_eeprom.h.

Referenced by EEPROM_I2C_Init().

#define EEPROM_ADDRESS_M24C64_32 0xA0 /* Support the dev

Definition at line 79 of file stm3210c_eval_eeprom.h.

Referenced by EEPROM_I2C_Init().

#define EEPROM_FAIL 1

Definition at line 92 of file stm3210c_eval_eeprom.h.

Referenced by BSP_EEPROM_Init(), BSP_EEPROM_ReadBuffer(), BSP_EEPROM_WriteBuffer(), EEPROM_I2C_Init(), EEPROM_I2C_ReadBuffer(), and EEPROM_I2C_WritePage().

#define EEPROM_MAX_TRIALS 300

Definition at line 100 of file stm3210c_eval_eeprom.h.

Referenced by EEPROM_I2C_Init(), and EEPROM_I2C_WaitEepromStandbyState().

#define EEPROM_OK 0

Definition at line 91 of file stm3210c_eval_eeprom.h.

Referenced by BSP_EEPROM_WriteBuffer(), EEPROM_I2C_Init(), EEPROM_I2C_ReadBuffer(),

EEPROM_I2C_WaitEepromStandbyState(), and **EEPROM_I2C_WritePage()**.

#define EEPROM_PAGESIZE_M24C08 16 /* Support the device: M

Definition at line **88** of file **stm3210c_eval_eeprom.h**.

Referenced by **EEPROM_I2C_Init()**.

#define EEPROM_PAGESIZE_M24C64_32 32 /* Support the device

Definition at line **87** of file **stm3210c_eval_eeprom.h**.

Referenced by **EEPROM_I2C_Init()**.

#define EEPROM_TIMEOUT 2

Definition at line **93** of file **stm3210c_eval_eeprom.h**.

Referenced by **EEPROM_I2C_WaitEepromStandbyState()**.

STM3210C_EVAL BSP User Manual

Exported_Functions

STM3210C_EVAL IO Expander

Functions

uint8_t	**BSP_IO_Init** (void) Initializes and configures the IO functionalities and configures all necessary hardware resources (GPIOs, clocks..).
uint32_t	**BSP_IO_ITGetStatus** (uint32_t IO_Pin) Gets the selected pins IT status.
void	**BSP_IO_ITClear** (uint32_t IO_Pin) Clears the selected IO IT pending bit.
uint8_t	**BSP_IO_ConfigPin** (uint32_t IO_Pin, IO_ModeTypedef IO_Mode) Configures the IO pin(s) according to IO mode structure value.
void	**BSP_IO_WritePin** (uint32_t IO_Pin, uint8_t PinState) Sets the selected pins state.
uint32_t	**BSP_IO_ReadPin** (uint32_t IO_Pin) Gets the selected pins current state.
void	**BSP_IO_TogglePin** (uint32_t IO_Pin) Toggles the selected pins state.

Function Documentation

uint8_t BSP_IO_ConfigPin (uint32_t IO_Pin, IO_ModeTypedef IO_Mode)

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

Parameters:

IO_Pin,: Output pin to be set or reset. This parameter can be any combination of the IO pins.

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
- IO_MODE_IT_LOW_LEVEL
- IO_MODE_IT_HIGH_LEVEL

Return values:

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

Definition at line **259** of file **stm3210c_eval_io.c**.

References **io1_driver**, **IO1_I2C_ADDRESS**, **IO1_PIN_ALL**, **IO1_PIN_OFFSET**, **io2_driver**, **IO2_I2C_ADDRESS**, **IO2_PIN_ALL**, **IO2_PIN_OFFSET**, and **IO_OK**.

Referenced by **ACCELERO_IO_ITConfig()**, **AUDIO_IO_Init()**, and **BSP_JOY_Init()**.

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 stmpe811 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 **157** of file **stm3210c_eval_io.c**.

References **io1_driver**, **IO1_I2C_ADDRESS**, **IO1_PIN_ALL**, **IO1_PIN_OFFSET**, **io2_driver**, **IO2_I2C_ADDRESS**, **IO2_PIN_ALL**, **IO2_PIN_OFFSET**, **IO_ERROR**, and **IO_OK**.

Referenced by **ACCELERO_IO_Init()**, **AUDIO_IO_Init()**, and **BSP_JOY_Init()**.

void BSP_IO_ITClear (uint32_t IO_Pin)

Clears the selected IO IT pending bit.

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

Return values:
None

Definition at line **224** of file **stm3210c_eval_io.c**.

References io1_driver, IO1_I2C_ADDRESS, IO1_PIN_ALL, IO1_PIN_OFFSET, io2_driver, IO2_I2C_ADDRESS, IO2_PIN_ALL, and IO2_PIN_OFFSET.

uint32_t BSP_IO_ITGetStatus (uint32_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:

Status of the checked IO pin(s).

Definition at line 194 of file stm3210c_eval_io.c.

References io1_driver, IO1_I2C_ADDRESS, IO1_PIN_ALL, IO1_PIN_OFFSET, io2_driver, IO2_I2C_ADDRESS, IO2_PIN_ALL, and IO2_PIN_OFFSET.

uint32_t BSP_IO_ReadPin (uint32_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 316 of file stm3210c_eval_io.c.

References io1_driver, IO1_I2C_ADDRESS, IO1_PIN_ALL,

IO1_PIN_OFFSET, io2_driver, IO2_I2C_ADDRESS, IO2_PIN_ALL, and IO2_PIN_OFFSET.

Referenced by BSP_JOY_GetState().

void BSP_IO_TogglePin (uint32_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.

Return values:

None

Definition at line 346 of file stm3210c_eval_io.c.

References BSP_IO_WritePin(), io1_driver, IO1_I2C_ADDRESS, IO1_PIN_ALL, IO1_PIN_OFFSET, io2_driver, IO2_I2C_ADDRESS, IO2_PIN_ALL, and IO2_PIN_OFFSET.

void BSP_IO_WritePin (uint32_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

Return values:

None

Definition at line **289** of file **stm3210c_eval_io.c**.

References **io1_driver**, **IO1_I2C_ADDRESS**, **IO1_PIN_ALL**, **IO1_PIN_OFFSET**, **io2_driver**, **IO2_I2C_ADDRESS**, **IO2_PIN_ALL**, and **IO2_PIN_OFFSET**.

Referenced by **AUDIO_IO_Init()**, **BSP_AUDIO_OUT_Stop()**, and **BSP_IO_TogglePin()**.

STM3210C_EVAL BSP User Manual

Exported Functions

STM3210C_EVAL LCD

Functions

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

uint16_t	**BSP_LCD_ReadPixel** (uint16_t Xpos, uint16_t Ypos) Reads an LCD pixel.
void	**BSP_LCD_DrawHLine** (uint16_t Xpos, uint16_t Ypos, uint16_t Length) Draws an horizontal line.
void	**BSP_LCD_DrawVLine** (uint16_t Xpos, uint16_t Ypos, uint16_t Length) Draws a vertical line.
void	**BSP_LCD_DrawLine** (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2) Draws an uni-line (between two points).
void	**BSP_LCD_DrawRect** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Draws a rectangle.
void	**BSP_LCD_DrawCircle** (uint16_t Xpos, uint16_t Ypos, uint16_t Radius) Draws a circle.
void	**BSP_LCD_DrawPolygon** (**pPoint** Points, uint16_t PointCount) Draws an poly-line (between many points).
void	**BSP_LCD_DrawEllipse** (int Xpos, int Ypos, int XRadius, int YRadius) Draws an ellipse on LCD.
void	**BSP_LCD_DrawBitmap** (uint16_t Xpos, uint16_t Ypos, uint8_t *pbmp) Draws a bitmap picture (16 bpp).
void	**BSP_LCD_DrawRGBImage** (uint16_t Xpos, uint16_t Ypos, uint16_t Xsize, uint16_t Ysize, uint8_t *pdata) Draws RGB Image (16 bpp).
void	**BSP_LCD_FillRect** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Draws a full rectangle.
void	**BSP_LCD_FillCircle** (uint16_t Xpos, uint16_t Ypos, uint16_t Radius)

	Draws a full circle.
void	**BSP_LCD_FillEllipse** (int Xpos, int Ypos, int XRadius, int YRadius) Draws a full ellipse.
void	**BSP_LCD_DisplayOn** (void) Enables the display.
void	**BSP_LCD_DisplayOff** (void) Disables the display.
void	**BSP_LCD_SetTextColor** (__IO uint16_t Color)
void	**BSP_LCD_SetBackColor** (__IO uint16_t Color)

Function Documentation

void BSP_LCD_Clear (uint16_t Color)

Clears the hole LCD.

Parameters:

Color,: Color of the background

Return values:

None

Definition at line **258** of file **stm3210c_eval_lcd.c**.

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

void BSP_LCD_ClearStringLine (uint16_t Line)

Clears the selected line.

Parameters:

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

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

Return values:

None

Definition at line **283** of file **stm3210c_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**,

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

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

Displays one character.

Parameters:

Xpos,: Start column address

Ypos,: Line where to display the character shape.

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

Return values:

None

Definition at line **303** of file **stm3210c_eval_lcd.c**.

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

Referenced by **BSP_LCD_DisplayStringAt()**.

void BSP_LCD_DisplayOff (void)

Disables the display.

Return values:

None

Definition at line **822** of file **stm3210c_eval_lcd.c**.

References **lcd_drv**.

void BSP_LCD_DisplayOn (void)

Enables the display.

Return values:

None

Definition at line **813** of file **stm3210c_eval_lcd.c**.

References **lcd_drv**.

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

Displays characters on the LCD.

Parameters:

Xpos,: X position (in pixel)

Ypos,: Y position (in pixel)

pText,: Pointer to string to display on LCD

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

- CENTER_MODE
- RIGHT_MODE
- LEFT_MODE

Return values:

None

Definition at line **321** of file **stm3210c_eval_lcd.c**.

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

Referenced by **BSP_LCD_DisplayStringAtLine()**.

void BSP_LCD_DisplayStringAtLine (uint16_t Line,
uint8_t * pText
)

Displays a character on the LCD.

Parameters:

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

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

pText,: Pointer to string to display on LCD

Return values:

None

Definition at line **380** of file **stm3210c_eval_lcd.c**.

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

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

)

Draws a bitmap picture (16 bpp).

Parameters:

Xpos,: Bmp X position in the LCD

Ypos,: Bmp Y position in the LCD

pbmp,: Pointer to Bmp picture address.

Return values:

None

Definition at line **664** of file **stm3210c_eval_lcd.c**.

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

Referenced by **LCD_DrawChar()**.

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

Draws a circle.

Parameters:

Xpos,: X position

Ypos,: Y position

Radius,: Circle radius

Return values:

None

Definition at line **552** of file **stm3210c_eval_lcd.c**.

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

Referenced by **BSP_LCD_FillCircle()**.

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

Draws an ellipse on LCD.

Parameters:

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

Return values:

None

Definition at line **631** of file **stm3210c_eval_lcd.c**.

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

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

Draws an horizontal line.

Parameters:

Xpos,: X position

Ypos,: Y position

Length,: Line length

Return values:

None

Definition at line **410** of file **stm3210c_eval_lcd.c**.

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

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

```
void BSP_LCD_DrawLine ( uint16_t x1,
                        uint16_t y1,
                        uint16_t x2,
                        uint16_t y2
                      )
```

Draws an uni-line (between two points).

Parameters:

x1,: **Point** 1 X position

y1,: **Point** 1 Y position

x2,: **Point** 2 X position

y2,: **Point** 2 Y position

Return values:

None

Definition at line **459** of file **stm3210c_eval_lcd.c**.

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

Referenced by **BSP_LCD_DrawPolygon()**.

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

Return values:

None

Definition at line **602** of file **stm3210c_eval_lcd.c**.

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

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

Draws a rectangle.

Parameters:

Xpos,: X position

Ypos,: Y position

Width,: Rectangle width

Height,: Rectangle height

Return values:

None

Definition at line **534** of file **stm3210c_eval_lcd.c**.

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

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

Draws RGB Image (16 bpp).

Parameters:

Xpos,: X position in the LCD

Ypos,: Y position in the LCD

Xsize,: X size in the LCD

Ysize,: Y size in the LCD

pdata,: Pointer to the RGB Image address.

Return values:

None

Definition at line **696** of file **stm3210c_eval_lcd.c**.

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

void BSP_LCD_DrawVLine (uint16_t Xpos,

```
                                    uint16_t  Ypos,
                                    uint16_t  Length
                                  )
```

Draws a vertical line.

Parameters:

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

Return values:

None

Definition at line **434** of file **stm3210c_eval_lcd.c**.

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

Referenced by **BSP_LCD_DrawRect()**.

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

Draws a full circle.

Parameters:

Xpos,:	X position
Ypos,:	Y position
Radius,:	Circle radius

Return values:

None

Definition at line **733** of file **stm3210c_eval_lcd.c**.

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

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

Draws a full ellipse.

Parameters:

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

Return values:

None

Definition at line **783** of file **stm3210c_eval_lcd.c**.

References **BSP_LCD_DrawHLine()**.

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

Draws a full rectangle.

Parameters:

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

Return values:

None

Definition at line **716** of file **stm3210c_eval_lcd.c**.

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

Referenced by **BSP_LCD_ClearStringLine()**.

uint16_t BSP_LCD_GetBackColor (void)

Gets the LCD background color.

Return values:

Used background color

Definition at line **209** of file **stm3210c_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**.

sFONT * BSP_LCD_GetFont (void)

Gets the LCD text font.

Return values:

Used font

Definition at line **248** of file **stm3210c_eval_lcd.c**.

References **LCD_DrawPropTypeDef::pFont**.

uint16_t BSP_LCD_GetTextColor (void)

Gets the LCD text color.

Return values:

Used text color.

Definition at line **200** of file **stm3210c_eval_lcd.c**.

References **LCD_DrawPropTypeDef::TextColor**.

uint32_t BSP_LCD_GetXSize (void)

Gets the LCD X size.

Return values:

Used LCD X size

Definition at line **182** of file **stm3210c_eval_lcd.c**.

References **lcd_drv**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_ClearStringLine()**, **BSP_LCD_DisplayStringAt()**, **BSP_LCD_DrawBitmap()**, and **BSP_LCD_DrawRGBImage()**.

uint32_t BSP_LCD_GetYSize (void)

Gets the LCD Y size.

Return values:

Used LCD Y size

Definition at line **191** of file **stm3210c_eval_lcd.c**.

References **lcd_drv**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_DrawBitmap()**, and **BSP_LCD_DrawRGBImage()**.

uint8_t BSP_LCD_Init (void)

Initializes the LCD.

Return values:

LCD state

Definition at line **141** of file **stm3210c_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**, **BSP_LCD_SetFont()**, **LCD_DEFAULT_FONT**, **lcd_drv**, **LCD_ERROR**, **LCD_OK**, **LCD_DrawPropTypeDef::pFont**, and **LCD_DrawPropTypeDef::TextColor**.

uint16_t BSP_LCD_ReadPixel (uint16_t Xpos,
uint16_t Ypos
)

Reads an LCD pixel.

Parameters:

Xpos,: X position

Ypos,: Y position

Return values:

RGB pixel color

Definition at line **391** of file **stm3210c_eval_lcd.c**.

References **lcd_drv**.

void BSP_LCD_SetBackColor (__IO uint16_t Color)

void BSP_LCD_SetBackColor (uint16_t Color)

Sets the LCD background color.

Parameters:

Color,: Background color code RGB(5-6-5)

Return values:

None

Definition at line **229** of file **stm3210c_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**.

void BSP_LCD_SetFont (sFONT * pFonts)

Sets the LCD text font.

Parameters:

pFonts,: Font to be used

Return values:

None

Definition at line **239** of file **stm3210c_eval_lcd.c**.

References **LCD_DrawPropTypeDef::pFont**.

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

void BSP_LCD_SetTextColor (__IO uint16_t Color)

void BSP_LCD_SetTextColor (uint16_t Color)

Sets the LCD text color.

Parameters:

Color,: Text color code RGB(5-6-5)

Return values:

None

Definition at line **219** of file **stm3210c_eval_lcd.c**.

References **LCD_DrawPropTypeDef::TextColor**.

Referenced by **BSP_LCD_Clear()**, **BSP_LCD_ClearStringLine()**, **BSP_LCD_FillCircle()**, and **BSP_LCD_FillRect()**.

STM3210C_EVAL BSP User Manual

Exported Functions

STM3210C_EVAL SD

Functions

uint8_t	**BSP_SD_Init** (void) Initializes the SD/SD communication.
uint8_t	**BSP_SD_IsDetected** (void) Detects if SD card is correctly plugged in the memory slot or not.
uint8_t	**BSP_SD_GetCardInfo** (**SD_CardInfo** *pCardInfo) Returns information about specific card.
uint8_t	**BSP_SD_ReadBlocks** (uint32_t *p32Data, uint64_t ReadAddr, uint16_t BlockSize, uint32_t NumberOfBlocks) Reads block(s) from a specified address in an SD card, in polling mode.
uint8_t	**BSP_SD_WriteBlocks** (uint32_t *p32Data, uint64_t WriteAddr, uint16_t BlockSize, uint32_t NumberOfBlocks) Writes block(s) to a specified address in an SD card, in polling mode.
uint8_t	**BSP_SD_GetStatus** (void) Returns the SD status.
uint8_t	**BSP_SD_Erase** (uint32_t StartAddr, uint32_t EndAddr) Erases the specified memory area of the given SD card.
void	**SD_IO_Init** (void) Initializes the SD Card and put it into StandBy State (Ready for data transfer).
void	**SD_IO_WriteByte** (uint8_t Data) Write a byte on the SD.
uint8_t	**SD_IO_ReadByte** (void) Read a byte from the SD.

HAL_StatusTypeDef	**SD_IO_WriteCmd** (uint8_t Cmd, uint32_t Arg, uint8_t Crc, uint8_t Response) Send 5 bytes command to the SD card and get response.
HAL_StatusTypeDef	**SD_IO_WaitResponse** (uint8_t Response) Wait response from the SD card.
void	**SD_IO_WriteDummy** (void) Send dummy byte with CS High.
static uint8_t	**SD_GetCSDRegister** (**SD_CSD** *Csd) Read the CSD card register.
static uint8_t	**SD_GetCIDRegister** (**SD_CID** *Cid) Read the CID card register.
static uint8_t	**SD_SendCmd** (uint8_t Cmd, uint32_t Arg, uint8_t Crc, uint8_t Response) Send 5 bytes command to the SD card and get response.
static **SD_Info**	**SD_GetDataResponse** (void) Get SD card data response.
static uint8_t	**SD_GoIdleState** (void) Put SD in Idle state.

Function Documentation

uint8_t BSP_SD_Erase (uint32_t StartAddr,
uint32_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 **684** of file **stm3210c_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, **SD_CMD_ERASE**, **SD_CMD_SD_ERASE_GRP_END**, **SD_CMD_SD_ERASE_GRP_START**, **SD_RESPONSE_NO_ERROR**, and **SD_SendCmd()**.

uint8_t BSP_SD_GetCardInfo (SD_CardInfo * pCardInfo)

Returns information about specific card.

Parameters:

pCardInfo,: pointer to a **SD_CardInfo** structure that contains all SD card information.

Return values:

The SD Response:

- MSD_ERROR : Sequence failed
- MSD_OK : Sequence succeed

Definition at line 217 of file stm3210c_eval_sd.c.

References SD_CardInfo::CardBlockSize, SD_CardInfo::CardCapacity, SD_CardInfo::Cid, SD_CardInfo::Csd, SD_CSD::DeviceSize, SD_CSD::DeviceSizeMul, MSD_ERROR, SD_CSD::RdBlockLen, SD_GetCIDRegister(), and SD_GetCSDRegister().

uint8_t BSP_SD_GetStatus (void)

Returns the SD status.

Return values:

The SD status.

Definition at line 652 of file stm3210c_eval_sd.c.

References MSD_OK.

uint8_t BSP_SD_Init (void)

Initializes the SD/SD communication.

Return values:

The SD Response:

- MSD_ERROR : Sequence failed
- MSD_OK : Sequence succeed

Definition at line 172 of file stm3210c_eval_sd.c.

References BSP_SD_IsDetected(), MSD_ERROR, SD_GoIdleState(), SD_IO_Init(), SD_NOT_PRESENT, SD_PRESENT, and SdStatus.

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 **196** of file **stm3210c_eval_sd.c**.

References **SD_DETECT_GPIO_PORT**, **SD_DETECT_PIN**, **SD_NOT_PRESENT**, and **SD_PRESENT**.

Referenced by **BSP_SD_Init()**.

```
uint8_t BSP_SD_ReadBlocks ( uint32_t *  p32Data,
                            uint64_t    ReadAddr,
                            uint16_t    BlockSize,
                            uint32_t    NumberOfBlocks
                          )
```

Reads block(s) from a specified address in an SD card, in polling mode.

Parameters:

p32Data,:	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
NumberOfBlocks,:	Number of SD blocks to read

Return values:

SD status

Definition at line **240** of file **stm3210c_eval_sd.c**.

References MSD_ERROR, MSD_OK, SD_CMD_READ_SINGLE_BLOCK, SD_CMD_SET_BLOCKLEN, SD_IO_ReadByte(), SD_IO_WaitResponse(), SD_IO_WriteCmd(), SD_IO_WriteDummy(), SD_RESPONSE_NO_ERROR, and SD_START_DATA_SINGLE_BLOCK_READ.

```
uint8_t BSP_SD_WriteBlocks ( uint32_t *  p32Data,
                             uint64_t    WriteAddr,
                             uint16_t    BlockSize,
                             uint32_t    NumberOfBlocks
                           )
```

Writes block(s) to a specified address in an SD card, in polling mode.

Parameters:

p32Data,:	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
NumberOfBlocks,:	Number of SD blocks to write

Return values:

SD status

Definition at line 306 of file stm3210c_eval_sd.c.

References MSD_ERROR, MSD_OK, SD_CMD_WRITE_SINGLE_BLOCK, SD_DATA_OK, SD_DUMMY_BYTE, SD_GetDataResponse(), SD_IO_ReadByte(), SD_IO_WriteByte(), SD_IO_WriteCmd(), SD_IO_WriteDummy(), SD_RESPONSE_NO_ERROR, and SD_START_DATA_SINGLE_BLOCK_WRITE.

static uint8_t SD_GetCIDRegister (SD_CID * Cid) [static]

Read the CID card register.

Reading the contents of the CID register in SPI mode is a simple read-block transaction.

Parameters:

Cid,: pointer on an CID register structure

Return values:

SD status

Definition at line **490** of file **stm3210c_eval_sd.c**.

References **SD_CID::CID_CRC**, **SD_CID::ManufactDate**, **SD_CID::ManufacturerID**, **MSD_ERROR**, **MSD_OK**, **SD_CID::OEM_AppliID**, **SD_CID::ProdName1**, **SD_CID::ProdName2**, **SD_CID::ProdRev**, **SD_CID::ProdSN**, **SD_CID::Reserved1**, **SD_CID::Reserved2**, **SD_CMD_SEND_CID**, **SD_DUMMY_BYTE**, **SD_IO_ReadByte()**, **SD_IO_WaitResponse()**, **SD_IO_WriteByte()**, **SD_IO_WriteCmd()**, **SD_IO_WriteDummy()**, **SD_RESPONSE_NO_ERROR**, and **SD_START_DATA_SINGLE_BLOCK_READ**.

Referenced by **BSP_SD_GetCardInfo()**.

uint8_t SD_GetCSDRegister (SD_CSD * Csd) [static]

Read the CSD card register.

Reading the contents of the CSD register in SPI mode is a simple read-block transaction.

Parameters:

Csd,: pointer on an SCD register structure

Return values:

SD status

< Reserved

Definition at line **372** of file **stm3210c_eval_sd.c**.

References **SD_CSD::CardComdClasses**, **SD_CSD::ContentProtectAppli**, **SD_CSD::CopyFlag**, **SD_CSD::CSD_CRC**, **SD_CSD::CSDStruct**, **SD_CSD::DeviceSize**, **SD_CSD::DeviceSizeMul**, **SD_CSD::DSRImpl**, **SD_CSD::ECC**, **SD_CSD::EraseGrMul**, **SD_CSD::EraseGrSize**, **SD_CSD::FileFormat**, **SD_CSD::FileFormatGrouop**, **SD_CSD::ManDeflECC**, **SD_CSD::MaxBusClkFrec**, **SD_CSD::MaxRdCurrentVDDMax**, **SD_CSD::MaxRdCurrentVDDMin**, **SD_CSD::MaxWrBlockLen**, **SD_CSD::MaxWrCurrentVDDMax**, **SD_CSD::MaxWrCurrentVDDMin**, **MSD_ERROR**, **MSD_OK**, **SD_CSD::NSAC**, **SD_CSD::PartBlockRead**, **SD_CSD::PermWrProtect**, **SD_CSD::RdBlockLen**, **SD_CSD::RdBlockMisalign**, **SD_CSD::Reserved1**, **SD_CSD::Reserved2**, **SD_CSD::Reserved3**, **SD_CSD::Reserved4**, **SD_CMD_SEND_CSD**, **SD_DUMMY_BYTE**, **SD_IO_ReadByte()**, **SD_IO_WaitResponse()**, **SD_IO_WriteByte()**, **SD_IO_WriteCmd()**, **SD_IO_WriteDummy()**, **SD_RESPONSE_NO_ERROR**, **SD_START_DATA_SINGLE_BLOCK_READ**, **SD_CSD::SysSpecVersion**, **SD_CSD::TAAC**, **SD_CSD::TempWrProtect**, **SD_CSD::WrBlockMisalign**, **SD_CSD::WriteBlockPaPartial**, **SD_CSD::WrProtectGrEnable**, **SD_CSD::WrProtectGrSize**, and **SD_CSD::WrSpeedFact**.

Referenced by **BSP_SD_GetCardInfo()**.

static SD_Info SD_GetDataResponse (void) [static]

Get SD card data response.

Return values:

The SD status: Read data response xxx0<status>1

- status 010: Data accecpted
- status 101: Data rejected due to a crc error
- status 110: Data rejected due to a Write error.
- status 111: Data rejected due to other error.

Definition at line **606** of file **stm3210c_eval_sd.c**.

References **SD_DATA_CRC_ERROR**, **SD_DATA_OK**, **SD_DATA_OTHER_ERROR**, **SD_DATA_WRITE_ERROR**, and **SD_IO_ReadByte()**.

Referenced by **BSP_SD_WriteBlocks()**.

static uint8_t SD_GoIdleState (void) [static]

Put SD in Idle state.

Return values:

SD status

Definition at line **661** of file **stm3210c_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, **SD_CMD_GO_IDLE_STATE**, **SD_CMD_SEND_OP_COND**, **SD_IN_IDLE_STATE**, **SD_RESPONSE_NO_ERROR**, and **SD_SendCmd()**.

Referenced by **BSP_SD_Init()**.

void SD_IO_Init (void)

Initializes the SD Card and put it into StandBy State (Ready for data transfer).

Return values:

None

Definition at line **1115** of file **stm3210c_eval.c**.

References **SD_CS_GPIO_CLK_ENABLE**, **SD_CS_GPIO_PORT**, **SD_CS_HIGH**, **SD_CS_PIN**, **SD_DETECT_EXTI_IRQn**, **SD_DETECT_GPIO_CLK_ENABLE**, **SD_DETECT_GPIO_PORT**, **SD_DETECT_PIN**, **SD_DUMMY_BYTE**, **SD_IO_WriteByte()**, and **SPIx_Init()**.

Referenced by **BSP_SD_Init()**.

uint8_t SD_IO_ReadByte (void)

Read a byte from the SD.

Return values:

The received byte.

Definition at line **1172** of file **stm3210c_eval.c**.

References **SPIx_Read()**.

Referenced by **BSP_SD_ReadBlocks()**, **BSP_SD_WriteBlocks()**, **SD_GetCIDRegister()**, **SD_GetCSDRegister()**, **SD_GetDataResponse()**, and **SD_IO_WaitResponse()**.

HAL_StatusTypeDef SD_IO_WaitResponse (uint8_t Response)

Wait response from the SD card.

Parameters:

Response,: Expected response from the SD card

Return values:

HAL_StatusTypeDef HAL Status

Definition at line **1226** of file **stm3210c_eval.c**.

References **SD_IO_ReadByte()**.

Referenced by **BSP_SD_ReadBlocks()**, **SD_GetCIDRegister()**, **SD_GetCSDRegister()**, and **SD_IO_WriteCmd()**.

void SD_IO_WriteByte (uint8_t Data)

Write a byte on the SD.

Parameters:

Data,: byte to send.

Return values:

None

Definition at line **1162** of file **stm3210c_eval.c**.

References **SPIx_Write()**.

Referenced by **BSP_SD_WriteBlocks()**, **SD_GetCIDRegister()**, **SD_GetCSDRegister()**, **SD_IO_Init()**, **SD_IO_WriteCmd()**, and **SD_IO_WriteDummy()**.

HAL_StatusTypeDef SD_IO_WriteCmd (uint8_t Cmd,
uint32_t Arg,
uint8_t Crc,
uint8_t Response
)

Send 5 bytes command to the SD card and get response.

Parameters:

Cmd,:	The user expected command to send to SD card.
Arg,:	The command argument.
Crc,:	The CRC.
Response,:	Expected response from the SD card

Return values:

HAL_StatusTypeDef	HAL Status

Definition at line **1191** of file **stm3210c_eval.c**.

References **SD_CS_LOW**, **SD_IO_WaitResponse()**, **SD_IO_WriteByte()**, and **SD_NO_RESPONSE_EXPECTED**.

Referenced by **BSP_SD_ReadBlocks()**, **BSP_SD_WriteBlocks()**, **SD_GetCIDRegister()**, **SD_GetCSDRegister()**, and **SD_SendCmd()**.

void SD_IO_WriteDummy (void)

Send dummy byte with CS High.

Return values:

None

Definition at line **1254** of file **stm3210c_eval.c**.

References **SD_CS_HIGH**, **SD_DUMMY_BYTE**, and **SD_IO_WriteByte()**.

Referenced by **BSP_SD_ReadBlocks()**, **BSP_SD_WriteBlocks()**, **SD_GetCIDRegister()**, **SD_GetCSDRegister()**, and **SD_SendCmd()**.

```
static uint8_t SD_SendCmd ( uint8_t  Cmd,
                            uint32_t Arg,
                            uint8_t  Crc,
                            uint8_t  Response
```

) [static]

Send 5 bytes command to the SD card and get response.

Parameters:

Cmd,:	The user expected command to send to SD card.
Arg,:	The command argument.
Crc,:	The CRC.
Response,:	Expected response from the SD card

Return values:

SD status

Definition at line **583** of file **stm3210c_eval_sd.c**.

References **MSD_ERROR**, **MSD_OK**, **SD_IO_WriteCmd()**, and **SD_IO_WriteDummy()**.

Referenced by **BSP_SD_Erase()**, and **SD_GoIdleState()**.

STM3210C_EVAL BSP User Manual

Exported_Functions

STM3210C_EVAL Touch Screen

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_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_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 **208** of file **stm3210c_eval_ts.c**.

References **TS_StateTypeDef::TouchDetected**, **ts_driver**, **TS_I2C_ADDRESS**, **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 **154** of file **stm3210c_eval_ts.c**.

References **ts_driver**, **TS_ERROR**, **TS_I2C_ADDRESS**, **TS_OK**, **ts_orientation**, **TS_SWAP_XY**, **ts_x_boundary**, and **ts_y_boundary**.

void BSP_TS_ITClear (void)

Clears all touch screen interrupts.

Return values:

None

Definition at line **257** of file **stm3210c_eval_ts.c**.

References **ts_driver**, and **TS_I2C_ADDRESS**.

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 **185** of file **stm3210c_eval_ts.c**.

References **ts_driver**, **TS_I2C_ADDRESS**, 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 **197** of file **stm3210c_eval_ts.c**.

References **ts_driver**, and **TS_I2C_ADDRESS**.

STM3210C_EVAL BSP User Manual

Private Variables

STM3210C-EVAL Common

Variables

GPIO_TypeDef *	**LED_PORT** [**LEDn**] LED variables.
const uint16_t	**LED_PIN** [**LEDn**]
GPIO_TypeDef *	**BUTTON_PORT** [**BUTTONn**] BUTTON variables.
const uint16_t	**BUTTON_PIN** [**BUTTONn**]
const uint16_t	**BUTTON_IRQn** [**BUTTONn**]
USART_TypeDef *	**COM_USART** [**COMn**] = {**EVAL_COM1**} COM variables.
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**}
uint32_t	**SpixTimeout** = **EVAL_SPIx_TIMEOUT_MAX** BUS variables.
static SPI_HandleTypeDef	**heval_Spi**
uint32_t	**I2cxTimeout** = **EVAL_I2Cx_TIMEOUT_MAX**
I2C_HandleTypeDef	**heval_I2c**

Variable Documentation

const uint16_t BUTTON_IRQn[BUTTONn]

Initial value:

```
 {WAKEUP_BUTTON_EXTI_IRQn,
                                      TAMPER_BUT
TON_EXTI_IRQn,
                                      KEY_BUTTON
_EXTI_IRQn}
```

Definition at line **134** of file **stm3210c_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 **130** of file **stm3210c_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}
```

BUTTON variables.

Definition at line **126** of file **stm3210c_eval.c**.

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

const uint16_t COM_RX_PIN[COMn] = {EVAL_COM1_RX_PIN}

Definition at line **150** of file **stm3210c_eval.c**.

Referenced by **BSP_COM_Init()**.

GPIO_TypeDef* COM_RX_PORT[COMn] = {EVAL_COM1_RX_GPIO_

Definition at line **146** of file **stm3210c_eval.c**.

Referenced by **BSP_COM_Init()**.

const uint16_t COM_TX_PIN[COMn] = {EVAL_COM1_TX_PIN}

Definition at line **148** of file **stm3210c_eval.c**.

Referenced by **BSP_COM_Init()**.

GPIO_TypeDef* COM_TX_PORT[COMn] = {EVAL_COM1_TX_GPIO_

Definition at line **144** of file **stm3210c_eval.c**.

Referenced by **BSP_COM_Init()**.

USART_TypeDef* COM_USART[COMn] = {EVAL_COM1}

COM variables.

Definition at line **142** of file **stm3210c_eval.c**.

Referenced by **BSP_COM_Init()**.

I2C_HandleTypeDef heval_I2c

Definition at line **162** of file **stm3210c_eval.c**.

Referenced by **I2Cx_Error()**, **I2Cx_Init()**, **I2Cx_IsDeviceReady()**, **I2Cx_ReadBuffer()**, **I2Cx_ReadData()**, **I2Cx_ReadMultiple()**, **I2Cx_WriteBuffer()**, and **I2Cx_WriteData()**.

SPI_HandleTypeDef heval_Spi [static]

Definition at line **157** of file **stm3210c_eval.c**.

Referenced by **LCD_IO_WriteMultipleData()**, **SPIx_Error()**, **SPIx_Init()**, **SPIx_Read()**, and **SPIx_Write()**.

uint32_t I2cxTimeout = EVAL_I2Cx_TIMEOUT_MAX

Definition at line **161** of file **stm3210c_eval.c**.

Referenced by **I2Cx_IsDeviceReady()**, **I2Cx_ReadBuffer()**, **I2Cx_ReadData()**, **I2Cx_ReadMultiple()**, **I2Cx_WriteBuffer()**, and **I2Cx_WriteData()**.

const uint16_t LED_PIN[LEDn]

Initial value:

```
{LED1_PIN,
```

```
                                    LED2_PIN,
                                    LED3_PIN,
                                    LED4_PIN}
```

Definition at line **118** of file **stm3210c_eval.c**.

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

GPIO_TypeDef* LED_PORT[LEDn]

Initial value:

```
 {LED1_GPIO_PORT,
                                    LED2_GPIO_PORT,
                                    LED3_GPIO_PORT,
                                    LED4_GPIO_PORT}
```

LED variables.

Definition at line **113** of file **stm3210c_eval.c**.

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

uint32_t SpixTimeout = EVAL_SPIx_TIMEOUT_MAX

BUS variables.

Definition at line **156** of file **stm3210c_eval.c**.

Referenced by **SPIx_Read()**, and **SPIx_Write()**.

STM3210C_EVAL BSP User Manual

Exported Types

STM3210C-EVAL Common

Enumerations

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

enum	**Button_TypeDef** { **BUTTON_WAKEUP** = 0, **BUTTON_TAMPER** = 1, **BUTTON_KEY** = 2 } BUTTON Types Definition. More...
enum	**ButtonMode_TypeDef** { **BUTTON_MODE_GPIO** = 0, **BUTTON_MODE_EXTI** = 1 }

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

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

Enumeration Type Documentation

enum Button_TypeDef

BUTTON Types Definition.

Enumerator:

BUTTON_WAKEUP

BUTTON_TAMPER

BUTTON_KEY

Definition at line **89** of file **stm3210c_eval.h**.

enum ButtonMode_TypeDef

Enumerator:

BUTTON_MODE_GPIO

BUTTON_MODE_EXTI

Definition at line **97** of file **stm3210c_eval.h**.

enum COM_TypeDef

COM Types Definition.

Enumerator:

COM1

COM2

Definition at line **127** of file **stm3210c_eval.h**.

enum JOYMode_TypeDef

Enumerator:

JOY_MODE_GPIO

JOY_MODE_EXTI

Definition at line **117** of file **stm3210c_eval.h**.

enum JOYState_TypeDef

JOYSTICK Types Definition.

Enumerator:

JOY_SEL

JOY_LEFT

JOY_RIGHT

JOY_DOWN

JOY_UP

JOY_NONE

Definition at line **106** of file **stm3210c_eval.h**.

enum Led_TypeDef

LED Types Definition.

Enumerator:

LED1

LED2

LED3

LED4

LED_GREEN

LED_ORANGE

LED_RED

LED_BLUE

Definition at line **72** of file **stm3210c_eval.h**.

STM3210C_EVAL BSP User Manual

AUDIO_Exported_Macros

STM3210C_EVAL_AUDIO

Defines

#define	**DMA_MAX**(_X_) (((_X_) <= **DMA_MAX_SZE**)? (_X_):**DMA_MAX_SZE**)

Define Documentation

#define DMA_MAX (_X_) (((_X_) <= DMA_MAX_SZE)? (_X_):DMA

Definition at line **133** of file **stm3210c_eval_audio.h**.

Referenced by **BSP_AUDIO_OUT_Play()**.

STM3210C_EVAL BSP User Manual

Private Function Prototypes

STM3210C_EVAL_EEPROM

Functions

static uint32_t	**EEPROM_I2C_Init** (void) Initializes peripherals used by the I2C EEPROM driver.
static uint32_t	**EEPROM_I2C_ReadBuffer** (uint8_t *pBuffer, uint16_t ReadAddr, uint32_t *NumByteToRead) Reads a block of data from the I2C EEPROM.
static uint32_t	**EEPROM_I2C_WritePage** (uint8_t *pBuffer, uint16_t WriteAddr, uint32_t *NumByteToWrite) Writes more than one byte to the EEPROM with a single WRITE cycle.
static uint32_t	**EEPROM_I2C_WaitEepromStandbyState** (void) Wait for EEPROM I2C Standby state.

Variables

EEPROM_DrvTypeDef	EEPROM_I2C_Drv

Function Documentation

static uint32_t EEPROM_I2C_Init (void) [static]

Initializes peripherals used by the I2C EEPROM driver.

Note:
There are 2 different versions of M24CXX (08 or 32 or 64). 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 **399** of file **stm3210c_eval_eeprom.c**.

References **EEPROM_ADDRESS_M24C08_BLOCK0**, **EEPROM_ADDRESS_M24C08_BLOCK1**, **EEPROM_ADDRESS_M24C08_BLOCK2**, **EEPROM_ADDRESS_M24C08_BLOCK3**, **EEPROM_ADDRESS_M24C64_32**, **EEPROM_FAIL**, **EEPROM_I2C_IO_Init()**, **EEPROM_I2C_IO_IsDeviceReady()**, **EEPROM_MAX_TRIALS**, **EEPROM_OK**, **EEPROM_PAGESIZE_M24C08**, **EEPROM_PAGESIZE_M24C64_32**, **EEPROMAddress**, and **EEPROMPageSize**.

static uint32_t EEPROM_I2C_ReadBuffer (uint8_t * pBuffer, uint16_t ReadAddr, uint32_t * NumByteToR[illegible]) [static]

Reads a block of data from the I2C 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.

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 **446** of file **stm3210c_eval_eeprom.c**.

References **EEPROM_FAIL**, **EEPROM_I2C_IO_ReadData()**, **EEPROM_OK**, and **EEPROMAddress**.

static uint32_t EEPROM_I2C_WaitEepromStandbyState (void) [st

Wait for EEPROM I2C 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 **514** of file **stm3210c_eval_eeprom.c**.

References **BSP_EEPROM_TIMEOUT_UserCallback()**, **EEPROM_I2C_IO_IsDeviceReady()**, **EEPROM_MAX_TRIALS**, **EEPROM_OK**, **EEPROM_TIMEOUT**, and **EEPROMAddress**.

Referenced by **EEPROM_I2C_WritePage()**.

static uint32_t EEPROM_I2C_WritePage (uint8_t * pBuffer,
uint16_t WriteAddr,
uint32_t * NumByteToWri
) [static]

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 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.

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 **483** of file **stm3210c_eval_eeprom.c**.

References **EEPROM_FAIL**, **EEPROM_I2C_IO_WriteData()**, **EEPROM_I2C_WaitEepromStandbyState()**, **EEPROM_OK**, and **EEPROMAddress**.

Variable Documentation

EEPROM_DrvTypeDef EEPROM_I2C_Drv

Initial value:

```
{
  EEPROM_I2C_Init,
  EEPROM_I2C_ReadBuffer,
  EEPROM_I2C_WritePage
}
```

Definition at line **135** of file **stm3210c_eval_eeprom.c**.

Referenced by **BSP_EEPROM_SelectDevice()**.

STM3210C_EVAL BSP User Manual

Private Variables

STM3210C_EVAL_EEPROM

Variables

__IO uint16_t	**EEPROMAddress** = 0
__IO uint16_t	**EEPROMPageSize** = 0
__IO uint16_t	**EEPROMDataRead** = 0
__IO uint8_t	**EEPROMDataWrite** = 0
static **EEPROM_DrvTypeDef** *	**EEPROM_SelectedDevice** = 0

Variable Documentation

EEPROM_DrvTypeDef* EEPROM_SelectedDevice = 0 [static]

Definition at line 120 of file stm3210c_eval_eeprom.c.

__IO uint16_t EEPROMAddress = 0

Definition at line 115 of file stm3210c_eval_eeprom.c.

Referenced by EEPROM_I2C_Init(), EEPROM_I2C_ReadBuffer(), EEPROM_I2C_WaitEepromStandbyState(), and EEPROM_I2C_WritePage().

__IO uint16_t EEPROMDataRead = 0

Definition at line 117 of file stm3210c_eval_eeprom.c.

__IO uint8_t EEPROMDataWrite = 0

Definition at line 118 of file stm3210c_eval_eeprom.c.

__IO uint16_t EEPROMPageSize = 0

Definition at line 116 of file stm3210c_eval_eeprom.c.

Referenced by BSP_EEPROM_WriteBuffer(), and EEPROM_I2C_Init().

User Manual by  1.7.6.1

STM3210C_EVAL BSP User Manual

STM3210C_EVAL_BUS

Exported Constants

Defines

#define	**IOE_IT_PIN** GPIO_PIN_14 IO Expander Interrupt line on EXTI.
#define	**IOE_IT_GPIO_PORT** GPIOB
#define	**IOE_IT_GPIO_CLK_ENABLE**() __HAL_RCC_GPIOB_CLK_I
#define	**IOE_IT_GPIO_CLK_DISABLE**() __HAL_RCC_GPIOB_CLK_
#define	**IOE_IT_EXTI_IRQn** EXTI15_10_IRQn
#define	**IOE_IT_EXTI_IRQHANDLER** EXTI15_10_IRQHandler
#define	**IO1_I2C_ADDRESS** 0x82
#define	**IO2_I2C_ADDRESS** 0x88
#define	**TS_I2C_ADDRESS** 0x82
#define	**L1S302DL_I2C_ADDRESS** 0x38
#define	**READWRITE_CMD** ((uint8_t)0x80)
#define	**MULTIPLEBYTE_CMD** ((uint8_t)0x40)
#define	**EVAL_I2Cx_SCL_PIN** GPIO_PIN_6 /* PB.06*/
#define	**EVAL_I2Cx_SCL_GPIO_PORT** GPIOB
#define	**EVAL_I2Cx_SDA_PIN** GPIO_PIN_7 /* PB.07*/
#define	**EVAL_I2Cx_SDA_GPIO_PORT** GPIOB
#define	**EVAL_I2Cx** I2C1
#define	**EVAL_I2Cx_CLK_ENABLE**() __HAL_RCC_I2C1_CLK_ENAI
#define	**EVAL_I2Cx_SDA_GPIO_CLK_ENABLE**() __HAL_RCC_GPI
#define	**EVAL_I2Cx_SCL_GPIO_CLK_ENABLE**() __HAL_RCC_GPI
#define	**EVAL_I2Cx_FORCE_RESET**() __HAL_RCC_I2C1_FORCE_
#define	**EVAL_I2Cx_RELEASE_RESET**() __HAL_RCC_I2C1_RELE/
#define	**EVAL_I2Cx_EV_IRQn** I2C1_EV_IRQn
#define	**EVAL_I2Cx_EV_IRQHandler** I2C1_EV_IRQHandler
#define	**EVAL_I2Cx_ER_IRQn** I2C1_ER_IRQn
#define	**EVAL_I2Cx_ER_IRQHandler** I2C1_ER_IRQHandler
#define	**EVAL_I2Cx_TIMEOUT_MAX** 3000
#define	**EVAL_SPIx** SPI3
#define	**EVAL_SPIx_CLK_ENABLE**() __HAL_RCC_SPI3_CLK_ENA
#define	**EVAL_SPIx_SCK_GPIO_PORT** GPIOC /* PC.10*/

```
#define EVAL_SPIx_SCK_PIN GPIO_PIN_10
#define EVAL_SPIx_SCK_GPIO_CLK_ENABLE() __HAL_RCC_GPI
#define EVAL_SPIx_SCK_GPIO_CLK_DISABLE() __HAL_RCC_GP
#define EVAL_SPIx_MISO_MOSI_GPIO_PORT GPIOC
#define EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE() __HAL_R
#define EVAL_SPIx_MISO_MOSI_GPIO_CLK_DISABLE() __HAL_R
#define EVAL_SPIx_MISO_PIN GPIO_PIN_11 /* PC.11*/
#define EVAL_SPIx_MOSI_PIN GPIO_PIN_12 /* PC.12*/
#define EVAL_SPIx_TIMEOUT_MAX 1000
```

Define Documentation

#define EVAL_I2Cx I2C1

Definition at line 340 of file stm3210c_eval.h.

Referenced by I2Cx_Init(), and I2Cx_MspInit().

#define EVAL_I2Cx_CLK_ENABLE ( ) __HAL_RCC_I2C1_CLK_EN

Definition at line 341 of file stm3210c_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_ER_IRQHandler I2C1_ER_IRQHandler

Definition at line 352 of file stm3210c_eval.h.

#define EVAL_I2Cx_ER_IRQn I2C1_ER_IRQn

Definition at line 351 of file stm3210c_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_EV_IRQHandler I2C1_EV_IRQHandler

Definition at line 350 of file stm3210c_eval.h.

#define EVAL_I2Cx_EV_IRQn I2C1_EV_IRQn

Definition at line 349 of file stm3210c_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_FORCE_RESET ( ) __HAL_RCC_I2C1_FORCE

Definition at line 345 of file stm3210c_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_RELEASE_RESET ( ) __HAL_RCC_I2C1_REL

Definition at line 346 of file stm3210c_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SCL_GPIO_CLK_ENABLE ( ) __HAL_RCC_GI

Definition at line 343 of file stm3210c_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SCL_GPIO_PORT GPIOB

Definition at line 335 of file stm3210c_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SCL_PIN GPIO_PIN_6 /* PB.06*/

Definition at line 334 of file stm3210c_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SDA_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line 342 of file stm3210c_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SDA_GPIO_PORT GPIOB

Definition at line 337 of file stm3210c_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_SDA_PIN GPIO_PIN_7 /* PB.07*/

Definition at line 336 of file stm3210c_eval.h.

Referenced by I2Cx_MspInit().

#define EVAL_I2Cx_TIMEOUT_MAX 3000

Definition at line 365 of file stm3210c_eval.h.

#define EVAL_SPIx SPI3

Definition at line 368 of file stm3210c_eval.h.

Referenced by SPIx_Init().

#define EVAL_SPIx_CLK_ENABLE ( ) __HAL_RCC_SPI3_CLK_EN

Definition at line 369 of file stm3210c_eval.h.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_MISO_MOSI_GPIO_CLK_DISABLE ( ) __HAL_

Definition at line **378** of file **stm3210c_eval.h**.

#define EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE ( ) __HAL_

Definition at line **377** of file **stm3210c_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_MISO_MOSI_GPIO_PORT GPIOC

Definition at line **376** of file **stm3210c_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_MISO_PIN GPIO_PIN_11 /* PC.11*/

Definition at line **379** of file **stm3210c_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_MOSI_PIN GPIO_PIN_12 /* PC.12*/

Definition at line **380** of file **stm3210c_eval.h**.

Referenced by **SPIx_MspInit()**.

#define EVAL_SPIx_SCK_GPIO_CLK_DISABLE ( ) __HAL_RCC_G

Definition at line 374 of file stm3210c_eval.h.

#define EVAL_SPIx_SCK_GPIO_CLK_ENABLE ( ) __HAL_RCC_G

Definition at line 373 of file stm3210c_eval.h.

Referenced by SPIx_MspInit().

#define EVAL_SPIx_SCK_GPIO_PORT GPIOC /* PC.10*/

Definition at line 371 of file stm3210c_eval.h.

Referenced by SPIx_MspInit().

#define EVAL_SPIx_SCK_PIN GPIO_PIN_10

Definition at line 372 of file stm3210c_eval.h.

Referenced by SPIx_MspInit().

#define EVAL_SPIx_TIMEOUT_MAX 1000

Definition at line 386 of file stm3210c_eval.h.

#define IO1_I2C_ADDRESS 0x82

Definition at line 313 of file stm3210c_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 IO2_I2C_ADDRESS 0x88

Definition at line 314 of file stm3210c_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 IOE_IT_EXTI_IRQHANDLER EXTI15_10_IRQHandler

Definition at line 310 of file stm3210c_eval.h.

#define IOE_IT_EXTI_IRQn EXTI15_10_IRQn

Definition at line 309 of file stm3210c_eval.h.

Referenced by I2Cx_ITConfig().

#define IOE_IT_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOB_CLI

Definition at line 308 of file stm3210c_eval.h.

#define IOE_IT_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOB_CLK

Definition at line 307 of file stm3210c_eval.h.

Referenced by I2Cx_ITConfig().

#define IOE_IT_GPIO_PORT GPIOB

Definition at line 306 of file stm3210c_eval.h.

Referenced by **I2Cx_ITConfig()**.

#define IOE_IT_PIN GPIO_PIN_14

IO Expander Interrupt line on EXTI.

Definition at line **305** of file **stm3210c_eval.h**.

Referenced by **I2Cx_ITConfig()**.

#define L1S302DL_I2C_ADDRESS 0x38

Definition at line **321** of file **stm3210c_eval.h**.

Referenced by **ACCELERO_IO_Read()**, and **ACCELERO_IO_Write()**.

#define MULTIPLEBYTE_CMD ((uint8_t)0x40)

Definition at line **328** of file **stm3210c_eval.h**.

#define READWRITE_CMD ((uint8_t)0x80)

Definition at line **326** of file **stm3210c_eval.h**.

#define TS_I2C_ADDRESS 0x82

Definition at line **315** of file **stm3210c_eval.h**.

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

STM3210C_EVAL BSP User Manual

AUDIO_Private_Variables

STM3210C_EVAL_AUDIO

Variables

static AUDIO_DrvTypeDef *	**pAudioDrv**
I2S_HandleTypeDef	**hAudioOutI2s**

Variable Documentation

I2S_HandleTypeDef hAudioOutI2s

Definition at line 177 of file stm3210c_eval_audio.c.

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

AUDIO_DrvTypeDef* pAudioDrv [static]

Definition at line 176 of file stm3210c_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().

STM3210C_EVAL BSP User Manual

Functions

Bus Operations Functions

STM3210C-EVAL Common

Functions

static void	**I2Cx_MspInit** (I2C_HandleTypeDef *hi2c) Eval I2Cx MSP Initialization.
static void	**I2Cx_Init** (void) Eval I2Cx Bus initialization.
static void	**I2Cx_ITConfig** (void) Configures I2C Interrupt.
static HAL_StatusTypeDef	**I2Cx_ReadMultiple** (uint8_t Addr, uint16_t Reg, uint16_t MemAddress, uint8_t *Buffer, uint16_t Length) Reads multiple data.
static void	**I2Cx_WriteData** (uint16_t Addr, uint8_t Reg, uint8_t Value) Write a value in a register of the device through BUS.
static HAL_StatusTypeDef	**I2Cx_WriteBuffer** (uint16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuffer, uint16_t Length) Write a value in a register of the device through BUS.
static uint8_t	**I2Cx_ReadData** (uint16_t Addr, uint8_t Reg) Read a value in a register of the device through BUS.
static HAL_StatusTypeDef	**I2Cx_ReadBuffer** (uint16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t *pBuffer, uint16_t Length) Reads multiple data on the BUS.
static HAL_StatusTypeDef	**I2Cx_IsDeviceReady** (uint16_t DevAddress, uint32_t Trials) Checks if target device is ready for communication.
static void	**I2Cx_Error** (uint8_t Addr)

	Manages error callback by re-initializing I2C.
static void	**SPIx_MspInit** (SPI_HandleTypeDef *hspi) Initializes SPI MSP.
static void	**SPIx_Init** (void) Initializes SPI HAL.
static uint32_t	**SPIx_Read** (void) SPI Read 4 bytes from device.
static void	**SPIx_Write** (uint8_t Value) SPI Write a byte to device.
static void	**SPIx_Error** (void) SPI error treatment function.

Function Documentation

static void I2Cx_Error (uint8_t Addr) [static]

Manages error callback by re-initializing I2C.

Parameters:

Addr,: I2C Address

Return values:

None

Definition at line **763** of file **stm3210c_eval.c**.

References **heval_I2c**, and **I2Cx_Init()**.

Referenced by **I2Cx_ReadBuffer()**, **I2Cx_ReadData()**, **I2Cx_ReadMultiple()**, **I2Cx_WriteBuffer()**, and **I2Cx_WriteData()**.

static void I2Cx_Init (void) [static]

Eval I2Cx Bus initialization.

Return values:

None

Definition at line **582** of file **stm3210c_eval.c**.

References **EVAL_I2Cx**, **heval_I2c**, and **I2Cx_MspInit()**.

Referenced by **EEPROM_I2C_IO_Init()**, **I2Cx_Error()**, **IOE_Init()**, and **TSENSOR_IO_Init()**.

static HAL_StatusTypeDef I2Cx_IsDeviceReady (uint16_t DevAddr

uint32_t Trials **[static]**

)

Checks if target device is ready for communication.

Note:
This function is used with Memory devices

Parameters:

DevAddress,: Target device address

Trials,: Number of trials

Return values:

HAL status

Definition at line **753** of file **stm3210c_eval.c**.

References **heval_I2c**, and **I2cxTimeout**.

Referenced by **EEPROM_I2C_IO_IsDeviceReady()**, and **TSENSOR_IO_IsDeviceReady()**.

static void I2Cx_ITConfig (void) [static]

Configures I2C Interrupt.

Return values:

None

Definition at line **606** of file **stm3210c_eval.c**.

References **IOE_IT_EXTI_IRQn**, **IOE_IT_GPIO_CLK_ENABLE**, **IOE_IT_GPIO_PORT**, and **IOE_IT_PIN**.

Referenced by **IOE_ITConfig()**.

static void I2Cx_MspInit (I2C_HandleTypeDef * hi2c) [static]

Eval I2Cx MSP Initialization.

Parameters:

hi2c,: I2C handle

Return values:

None

Definition at line **532** of file **stm3210c_eval.c**.

References **EVAL_I2Cx**, **EVAL_I2Cx_CLK_ENABLE**, **EVAL_I2Cx_ER_IRQn**, **EVAL_I2Cx_EV_IRQn**, **EVAL_I2Cx_FORCE_RESET**, **EVAL_I2Cx_RELEASE_RESET**, **EVAL_I2Cx_SCL_GPIO_CLK_ENABLE**, **EVAL_I2Cx_SCL_GPIO_PORT**, **EVAL_I2Cx_SCL_PIN**, **EVAL_I2Cx_SDA_GPIO_CLK_ENABLE**, **EVAL_I2Cx_SDA_GPIO_PORT**, and **EVAL_I2Cx_SDA_PIN**.

Referenced by **I2Cx_Init()**.

static HAL_StatusTypeDef I2Cx_ReadBuffer (uint16_t Addr, uint8_t Reg, uint16_t RegSize, uint8_t * pBuffer, uint16_t Length) [static]

Reads multiple data on the BUS.

Parameters:

Addr,: I2C Address

Reg,: Reg Address

RegSize : The target register size (can be 8BIT or 16BIT)

pBuffer,: pointer to read data buffer
Length,: length of the data

Return values:
0 if no problems to read multiple data

Definition at line **731** of file **stm3210c_eval.c**.

References **heval_I2c**, **I2Cx_Error()**, and **I2cxTimeout**.

Referenced by **ACCELERO_IO_Read()**, **EEPROM_I2C_IO_ReadData()**, and **TSENSOR_IO_Read()**.

static uint8_t I2Cx_ReadData (uint16_t Addr,
uint8_t Reg
) [static]

Read a value in a register of the device through BUS.

Parameters:
Addr,: Device address on BUS Bus.
Reg,: The target register address to write

Return values:
Data read at register @

Definition at line **705** of file **stm3210c_eval.c**.

References **heval_I2c**, **I2Cx_Error()**, and **I2cxTimeout**.

Referenced by **AUDIO_IO_Read()**, and **IOE_Read()**.

static HAL_StatusTypeDef I2Cx_ReadMultiple (uint8_t Addr,
uint16_t Reg,
uint16_t MemAddre

```
                                        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 **639** of file **stm3210c_eval.c**.

References **heval_I2c**, **I2Cx_Error()**, and **I2cxTimeout**.

Referenced by **IOE_ReadMultiple()**.

```
static HAL_StatusTypeDef I2Cx_WriteBuffer ( uint16_t   Addr,
                                            uint8_t    Reg,
                                            uint16_t   RegSize,
                                            uint8_t *  pBuffer,
                                            uint16_t   Length
                                          )            [static]
```

Write a value in a register of the device through BUS.

Parameters:

Addr,:	Device address on BUS Bus.
Reg,:	The target register address to write

RegSize,: The target register size (can be 8BIT or 16BIT)
pBuffer,: The target register value to be written
Length,: buffer size to be written

Return values:
None

Definition at line **684** of file **stm3210c_eval.c**.

References **heval_I2c**, **I2Cx_Error()**, and **I2cxTimeout**.

Referenced by **ACCELERO_IO_Write()**, **EEPROM_I2C_IO_WriteData()**, and **TSENSOR_IO_Write()**.

static void I2Cx_WriteData (uint16_t Addr,
uint8_t Reg,
uint8_t Value
) [static]

Write a value in a register of the device through BUS.

Parameters:
Addr,: Device address on BUS Bus.
Reg,: The target register address to write
Value,: The target register value to be written

Return values:
None

Definition at line **661** of file **stm3210c_eval.c**.

References **heval_I2c**, **I2Cx_Error()**, and **I2cxTimeout**.

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

static void SPIx_Error (void) [static]

SPI error treatment function.

Return values:
None

Definition at line 887 of file stm3210c_eval.c.

References heval_Spi, and SPIx_Init().

Referenced by SPIx_Read(), and SPIx_Write().

static void SPIx_Init (void) [static]

Initializes SPI HAL.

Return values:
None

Definition at line 814 of file stm3210c_eval.c.

References EVAL_SPIx, heval_Spi, and SPIx_MspInit().

Referenced by LCD_IO_Init(), SD_IO_Init(), and SPIx_Error().

static void SPIx_MspInit (SPI_HandleTypeDef * hspi) [static]

Initializes SPI MSP.

Return values:
None

Definition at line 780 of file stm3210c_eval.c.

References EVAL_SPIx_CLK_ENABLE,

EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE, EVAL_SPIx_MISO_MOSI_GPIO_PORT, EVAL_SPIx_MISO_PIN, EVAL_SPIx_MOSI_PIN, EVAL_SPIx_SCK_GPIO_CLK_ENABLE, EVAL_SPIx_SCK_GPIO_PORT, and EVAL_SPIx_SCK_PIN.

Referenced by SPIx_Init().

static uint32_t SPIx_Read (void) [static]

SPI Read 4 bytes from device.

Return values:

Read data

Definition at line 846 of file stm3210c_eval.c.

References heval_Spi, SPIx_Error(), and SpixTimeout.

Referenced by LCD_IO_ReadData(), and SD_IO_ReadByte().

static void SPIx_Write (uint8_t Value) [static]

SPI Write a byte to device.

Parameters:

Value,: value to be written

Return values:

None

Definition at line 869 of file stm3210c_eval.c.

References heval_Spi, SPIx_Error(), and SpixTimeout.

Referenced by LCD_IO_ReadData(), LCD_IO_WriteMultipleData(), LCD_IO_WriteReg(), and SD_IO_WriteByte().

STM3210C_EVAL BSP User Manual

Private_Variables

STM3210C_EVAL IO Expander

Variables

static IO_DrvTypeDef *	**io1_driver**
static IO_DrvTypeDef *	**io2_driver**

Variable Documentation

IO_DrvTypeDef* io1_driver [static]

Definition at line 122 of file stm3210c_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().

IO_DrvTypeDef* io2_driver [static]

Definition at line 123 of file stm3210c_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().

STM3210C_EVAL BSP User Manual

Exported_Constants

STM3210C_EVAL IO Expander

Defines

#define	**IO1_PIN_OFFSET** 0
#define	**IO2_PIN_OFFSET** 8
#define	**IO1_PIN_0** (uint32_t)(0x00000001 << IO1_PIN_OFFSET)
#define	**IO1_PIN_1** (uint32_t)(0x00000002 << IO1_PIN_OFFSET)
#define	**IO1_PIN_2** (uint32_t)(0x00000004 << IO1_PIN_OFFSET)
#define	**IO1_PIN_3** (uint32_t)(0x00000008 << IO1_PIN_OFFSET)
#define	**IO1_PIN_4** (uint32_t)(0x00000010 << IO1_PIN_OFFSET)
#define	**IO1_PIN_5** (uint32_t)(0x00000020 << IO1_PIN_OFFSET)
#define	**IO1_PIN_6** (uint32_t)(0x00000040 << IO1_PIN_OFFSET)
#define	**IO1_PIN_7** (uint32_t)(0x00000080 << IO1_PIN_OFFSET)
#define	**IO1_PIN_ALL** (uint32_t)(0x000000FF << IO1_PIN_OFFSET)
#define	**IO2_PIN_0** (uint32_t)(0x00000001 << IO2_PIN_OFFSET)
#define	**IO2_PIN_1** (uint32_t)(0x00000002 << IO2_PIN_OFFSET)
#define	**IO2_PIN_2** (uint32_t)(0x00000004 << IO2_PIN_OFFSET)
#define	**IO2_PIN_3** (uint32_t)(0x00000008 << IO2_PIN_OFFSET)
#define	**IO2_PIN_4** (uint32_t)(0x00000010 << IO2_PIN_OFFSET)
#define	**IO2_PIN_5** (uint32_t)(0x00000020 << IO2_PIN_OFFSET)
#define	**IO2_PIN_6** (uint32_t)(0x00000040 << IO2_PIN_OFFSET)
#define	**IO2_PIN_7** (uint32_t)(0x00000080 << IO2_PIN_OFFSET)
#define	**IO2_PIN_ALL** (uint32_t)(0x000000FF << IO2_PIN_OFFSET)

Define Documentation

#define IO1_PIN_0 (uint32_t)(0x00000001 << IO1_PIN_OFFSET)

Definition at line 90 of file stm3210c_eval_io.h.

#define IO1_PIN_1 (uint32_t)(0x00000002 << IO1_PIN_OFFSET)

Definition at line 91 of file stm3210c_eval_io.h.

#define IO1_PIN_2 (uint32_t)(0x00000004 << IO1_PIN_OFFSET)

Definition at line 92 of file stm3210c_eval_io.h.

#define IO1_PIN_3 (uint32_t)(0x00000008 << IO1_PIN_OFFSET)

Definition at line 93 of file stm3210c_eval_io.h.

#define IO1_PIN_4 (uint32_t)(0x00000010 << IO1_PIN_OFFSET)

Definition at line 94 of file stm3210c_eval_io.h.

#define IO1_PIN_5 (uint32_t)(0x00000020 << IO1_PIN_OFFSET)

Definition at line 95 of file stm3210c_eval_io.h.

#define IO1_PIN_6 (uint32_t)(0x00000040 << IO1_PIN_OFFSET)

Definition at line 96 of file stm3210c_eval_io.h.

#define IO1_PIN_7 (uint32_t)(0x00000080 << IO1_PIN_OFFSET)

Definition at line 97 of file stm3210c_eval_io.h.

#define IO1_PIN_ALL (uint32_t)(0x000000FF << IO1_PIN_OFFSET)

Definition at line 98 of file stm3210c_eval_io.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 IO1_PIN_OFFSET 0

Definition at line 85 of file stm3210c_eval_io.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 IO2_PIN_0 (uint32_t)(0x00000001 << IO2_PIN_OFFSET)

Definition at line 101 of file stm3210c_eval_io.h.

#define IO2_PIN_1 (uint32_t)(0x00000002 << IO2_PIN_OFFSET)

Definition at line 102 of file stm3210c_eval_io.h.

#define IO2_PIN_2 (uint32_t)(0x00000004 << IO2_PIN_OFFSET)

Definition at line **103** of file **stm3210c_eval_io.h**.

#define IO2_PIN_3 (uint32_t)(0x00000008 << IO2_PIN_OFFSET)

Definition at line **104** of file **stm3210c_eval_io.h**.

#define IO2_PIN_4 (uint32_t)(0x00000010 << IO2_PIN_OFFSET)

Definition at line **105** of file **stm3210c_eval_io.h**.

#define IO2_PIN_5 (uint32_t)(0x00000020 << IO2_PIN_OFFSET)

Definition at line **106** of file **stm3210c_eval_io.h**.

#define IO2_PIN_6 (uint32_t)(0x00000040 << IO2_PIN_OFFSET)

Definition at line **107** of file **stm3210c_eval_io.h**.

#define IO2_PIN_7 (uint32_t)(0x00000080 << IO2_PIN_OFFSET)

Definition at line **108** of file **stm3210c_eval_io.h**.

#define IO2_PIN_ALL (uint32_t)(0x000000FF << IO2_PIN_OFFSET)

Definition at line **109** of file **stm3210c_eval_io.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 IO2_PIN_OFFSET 8

Definition at line **87** of file **stm3210c_eval_io.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()**.

STM3210C_EVAL BSP User Manual

Exported_Types

STM3210C_EVAL IO Expander

Enumerations

enum	**IO_StatusTypeDef** { **IO_OK** = 0x00, **IO_ERROR** = 0x01, **IO_TIMEOUT** = 0x02 }

Enumeration Type Documentation

enum IO_StatusTypeDef

Enumerator:

IO_OK

IO_ERROR

IO_TIMEOUT

Definition at line **69** of file **stm3210c_eval_io.h**.

STM3210C_EVAL BSP User Manual

Private Functions

STM3210C_EVAL LCD

Functions

static void	**LCD_DrawPixel** (uint16_t Xpos, uint16_t Ypos, uint16_t RGBCode) Draws a pixel on LCD.
static void	**LCD_DrawChar** (uint16_t Xpos, uint16_t Ypos, const uint8_t *pChar) Draws a character on LCD.
static void	**LCD_SetDisplayWindow** (uint16_t Xpos, uint16_t Ypos, uint16_t Width, uint16_t Height) Sets display window.

Function Documentation

static void LCD_DrawChar (uint16_t Xpos,
uint16_t Ypos,
const uint8_t * pChar
) [static]

Draws a character on LCD.

Parameters:

Xpos,: Line where to display the character shape
Ypos,: Start column address
pChar,: Pointer to the character data

Return values:

None

Definition at line **860** of file **stm3210c_eval_lcd.c**.

References **LCD_DrawPropTypeDef::BackColor**, **bitmap**, **BSP_LCD_DrawBitmap()**, **OFFSET_BITMAP**, **LCD_DrawPropTypeDef::pFont**, and **LCD_DrawPropTypeDef::TextColor**.

Referenced by **BSP_LCD_DisplayChar()**.

static void LCD_DrawPixel (uint16_t Xpos,
uint16_t Ypos,
uint16_t RGBCode
) [static]

Draws a pixel on LCD.

Parameters:

Xpos,: X position
Ypos,: Y position
RGBCode,: Pixel color in RGB mode (5-6-5)

Return values:
None

Definition at line **845** of file **stm3210c_eval_lcd.c**.

References **lcd_drv**.

Referenced by **BSP_LCD_DrawCircle()**, **BSP_LCD_DrawEllipse()**, **BSP_LCD_DrawHLine()**, **BSP_LCD_DrawLine()**, and **BSP_LCD_DrawVLine()**.

static void LCD_SetDisplayWindow (uint16_t Xpos,
uint16_t Ypos,
uint16_t Width,
uint16_t Height
) [static]

Sets display window.

Parameters:
Xpos,: LCD X position
Ypos,: LCD Y position
Width,: LCD window width
Height,: LCD window height

Return values:
None

Definition at line **931** of file **stm3210c_eval_lcd.c**.

References **lcd_drv**.

Referenced by **BSP_LCD_DrawBitmap()**, and **BSP_LCD_DrawRGBImage()**.

STM3210C_EVAL BSP User Manual

STM3210C_EVAL_LED

Exported Constants

Define for STM3210C_EVAL board. More...

Defines

```
#define LEDn 4
#define LED1_PIN GPIO_PIN_7 /* PD.07*/
#define LED1_GPIO_PORT GPIOD
#define LED1_GPIO_CLK_ENABLE() __HAL_RCC_GPIOD_CLK_E
#define LED1_GPIO_CLK_DISABLE() __HAL_RCC_GPIOD_CLK_D
#define LED2_PIN GPIO_PIN_13 /* PD.13*/
#define LED2_GPIO_PORT GPIOD
#define LED2_GPIO_CLK_ENABLE() __HAL_RCC_GPIOD_CLK_E
#define LED2_GPIO_CLK_DISABLE() __HAL_RCC_GPIOD_CLK_D
#define LED3_PIN GPIO_PIN_3 /* PD.03*/
#define LED3_GPIO_PORT GPIOD
#define LED3_GPIO_CLK_ENABLE() __HAL_RCC_GPIOD_CLK_E
#define LED3_GPIO_CLK_DISABLE() __HAL_RCC_GPIOD_CLK_D
#define LED4_PIN GPIO_PIN_4 /* PD.04*/
#define LED4_GPIO_PORT GPIOD
#define LED4_GPIO_CLK_ENABLE() __HAL_RCC_GPIOD_CLK_E
#define LED4_GPIO_CLK_DISABLE() __HAL_RCC_GPIOD_CLK_D
#define LEDx_GPIO_CLK_ENABLE(__LED__)
#define LEDx_GPIO_CLK_DISABLE(__LED__)
```

Detailed Description

Define for STM3210C_EVAL board.

Define Documentation

#define LED1_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOD_CLK

Definition at line **155** of file **stm3210c_eval.h**.

#define LED1_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOD_CLK_

Definition at line **154** of file **stm3210c_eval.h**.

#define LED1_GPIO_PORT GPIOD

Definition at line **153** of file **stm3210c_eval.h**.

#define LED1_PIN GPIO_PIN_7 /* PD.07*/

Definition at line **152** of file **stm3210c_eval.h**.

#define LED2_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOD_CLK

Definition at line **160** of file **stm3210c_eval.h**.

#define LED2_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOD_CLK_

Definition at line **159** of file **stm3210c_eval.h**.

#define LED2_GPIO_PORT GPIOD

Definition at line **158** of file **stm3210c_eval.h**.

#define LED2_PIN GPIO_PIN_13 /* PD.13*/

Definition at line 157 of file stm3210c_eval.h.

#define LED3_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOD_CLK

Definition at line 166 of file stm3210c_eval.h.

#define LED3_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOD_CLK_

Definition at line 165 of file stm3210c_eval.h.

#define LED3_GPIO_PORT GPIOD

Definition at line 164 of file stm3210c_eval.h.

#define LED3_PIN GPIO_PIN_3 /* PD.03*/

Definition at line 163 of file stm3210c_eval.h.

#define LED4_GPIO_CLK_DISABLE ( ) __HAL_RCC_GPIOD_CLK

Definition at line 172 of file stm3210c_eval.h.

#define LED4_GPIO_CLK_ENABLE ( ) __HAL_RCC_GPIOD_CLK_

Definition at line 171 of file stm3210c_eval.h.

#define LED4_GPIO_PORT GPIOD

Definition at line 170 of file stm3210c_eval.h.

#define LED4_PIN GPIO_PIN_4 /* PD.04*/

Definition at line 169 of file stm3210c_eval.h.

#define LEDn 4

Definition at line 150 of file stm3210c_eval.h.

#define LEDx_GPIO_CLK_DISABLE (__LED__)

Value:

```
(((__LED__) == LED1) ? LED1_GPIO_CLK_DISABLE() :\
                                                ((__LED
__) == LED2) ? LED2_GPIO_CLK_DISABLE() :\
                                                ((__LED
__) == LED3) ? LED3_GPIO_CLK_DISABLE() :\
                                                ((__LED
__) == LED4) ? LED4_GPIO_CLK_DISABLE() : 0 )
```

Definition at line 179 of file stm3210c_eval.h.

#define LEDx_GPIO_CLK_ENABLE (__LED__)

Value:

```
do { if ((__LED__) == LED1) LED1_GPIO_CLK_ENABLE(
); else \
                                                  if
((__LED__) == LED2) LED2_GPIO_CLK_ENABLE(); else
\
```

```
                                                                  if
((__LED__) == LED3) LED3_GPIO_CLK_ENABLE(); else
\
                                                                  if
((__LED__) == LED4) LED4_GPIO_CLK_ENABLE();} whil
e(0)
```

Definition at line **174** of file **stm3210c_eval.h**.

Referenced by **BSP_LED_Init()**.

STM3210C_EVAL BSP User Manual

Private Defines

STM3210C_EVAL LCD

Defines

#define	**POLY_X**(Z) ((int32_t)((Points + (Z))->X))
#define	**POLY_Y**(Z) ((int32_t)((Points + (Z))->Y))
#define	**MAX_HEIGHT_FONT** 17
#define	**MAX_WIDTH_FONT** 24
#define	**OFFSET_BITMAP** 54

Define Documentation

#define MAX_HEIGHT_FONT 17

Definition at line 92 of file stm3210c_eval_lcd.c.

#define MAX_WIDTH_FONT 24

Definition at line 93 of file stm3210c_eval_lcd.c.

#define OFFSET_BITMAP 54

Definition at line 94 of file stm3210c_eval_lcd.c.

Referenced by LCD_DrawChar().

#define POLY_X (Z) ((int32_t)((Points + (Z))->X))

Definition at line 89 of file stm3210c_eval_lcd.c.

#define POLY_Y (Z) ((int32_t)((Points + (Z))->Y))

Definition at line 90 of file stm3210c_eval_lcd.c.

STM3210C_EVAL BSP User Manual

Exported_Constants

STM3210C_EVAL SD

Defines

#define	**SD_BLOCK_SIZE** 0x200 Block Size.
#define	**SD_START_DATA_SINGLE_BLOCK_READ** 0xFE /* Data token start byte, Start Single Block Read */ Start Data tokens: Tokens (necessary because at nop/idle (and CS active) only 0xff is on the data/command line)
#define	**SD_START_DATA_MULTIPLE_BLOCK_READ** 0xFE /* Data token start byte, Start Multiple Block Read */
#define	**SD_START_DATA_SINGLE_BLOCK_WRITE** 0xFE /* Data token start byte, Start Single Block Write */
#define	**SD_START_DATA_MULTIPLE_BLOCK_WRITE** 0xFD /* Data token start byte, Start Multiple Block Write */
#define	**SD_STOP_DATA_MULTIPLE_BLOCK_WRITE** 0xFD /* Data toke stop byte, Stop Multiple Block Write */
#define	**SD_PRESENT** ((uint8_t)0x01) SD detection on its memory slot.
#define	**SD_NOT_PRESENT** ((uint8_t)0x00)
#define	**SD_CMD_GO_IDLE_STATE** 0 /* CMD0 = 0x40 */ Commands: CMDxx = CMD-number | 0x40.
#define	**SD_CMD_SEND_OP_COND** 1 /* CMD1 = 0x41 */
#define	**SD_CMD_SEND_CSD** 9 /* CMD9 = 0x49 */
#define	**SD_CMD_SEND_CID** 10 /* CMD10 = 0x4A */
#define	**SD_CMD_STOP_TRANSMISSION** 12 /* CMD12 = 0x4C */
#define	**SD_CMD_SEND_STATUS** 13 /* CMD13 = 0x4D */
#define	**SD_CMD_SET_BLOCKLEN** 16 /* CMD16 = 0x50 */
#define	**SD_CMD_READ_SINGLE_BLOCK** 17 /* CMD17 = 0x51 */
#define	**SD_CMD_READ_MULT_BLOCK** 18 /* CMD18 = 0x52 */
#define	**SD_CMD_SET_BLOCK_COUNT** 23 /* CMD23 = 0x57 */
#define	**SD_CMD_WRITE_SINGLE_BLOCK** 24 /* CMD24 = 0x58 */
#define	**SD_CMD_WRITE_MULT_BLOCK** 25 /* CMD25 = 0x59 */
#define	**SD_CMD_PROG_CSD** 27 /* CMD27 = 0x5B */

```
#define SD_CMD_SET_WRITE_PROT    28 /* CMD28 = 0x5C */
#define SD_CMD_CLR_WRITE_PROT    29 /* CMD29 = 0x5D */
#define SD_CMD_SEND_WRITE_PROT   30 /* CMD30 = 0x5E */
#define SD_CMD_SD_ERASE_GRP_START   32 /* CMD32 = 0x60 */
#define SD_CMD_SD_ERASE_GRP_END   33 /* CMD33 = 0x61 */
#define SD_CMD_UNTAG_SECTOR   34 /* CMD34 = 0x62 */
#define SD_CMD_ERASE_GRP_START   35 /* CMD35 = 0x63 */
#define SD_CMD_ERASE_GRP_END   36 /* CMD36 = 0x64 */
#define SD_CMD_UNTAG_ERASE_GROUP   37 /* CMD37 = 0x65 */
#define SD_CMD_ERASE   38 /* CMD38 = 0x66 */
```

Define Documentation

#define SD_BLOCK_SIZE 0x200

Block Size.

Definition at line 180 of file stm3210c_eval_sd.h.

#define SD_CMD_CLR_WRITE_PROT 29 /* CMD29 = 0x5D */

Definition at line 216 of file stm3210c_eval_sd.h.

#define SD_CMD_ERASE 38 /* CMD38 = 0x66 */

Definition at line 224 of file stm3210c_eval_sd.h.

Referenced by BSP_SD_Erase().

#define SD_CMD_ERASE_GRP_END 36 /* CMD36 = 0x64 */

Definition at line 222 of file stm3210c_eval_sd.h.

#define SD_CMD_ERASE_GRP_START 35 /* CMD35 = 0x63 */

Definition at line 221 of file stm3210c_eval_sd.h.

#define SD_CMD_GO_IDLE_STATE 0 /* CMD0 = 0x40 */

Commands: CMDxx = CMD-number | 0x40.

Definition at line 202 of file stm3210c_eval_sd.h.

Referenced by **SD_GoIdleState()**.

#define SD_CMD_PROG_CSD 27 /* CMD27 = 0x5B */

Definition at line **214** of file **stm3210c_eval_sd.h**.

#define SD_CMD_READ_MULT_BLOCK 18 /* CMD18 = 0x52 */

Definition at line **210** of file **stm3210c_eval_sd.h**.

#define SD_CMD_READ_SINGLE_BLOCK 17 /* CMD17 = 0x51 */

Definition at line **209** of file **stm3210c_eval_sd.h**.

Referenced by **BSP_SD_ReadBlocks()**.

#define SD_CMD_SD_ERASE_GRP_END 33 /* CMD33 = 0x61 */

Definition at line **219** of file **stm3210c_eval_sd.h**.

Referenced by **BSP_SD_Erase()**.

#define SD_CMD_SD_ERASE_GRP_START 32 /* CMD32 = 0x60 */

Definition at line **218** of file **stm3210c_eval_sd.h**.

Referenced by **BSP_SD_Erase()**.

#define SD_CMD_SEND_CID 10 /* CMD10 = 0x4A */

Definition at line **205** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCIDRegister()**.

#define SD_CMD_SEND_CSD 9 /* CMD9 = 0x49 */

Definition at line **204** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GetCSDRegister()**.

#define SD_CMD_SEND_OP_COND 1 /* CMD1 = 0x41 */

Definition at line **203** of file **stm3210c_eval_sd.h**.

Referenced by **SD_GoIdleState()**.

#define SD_CMD_SEND_STATUS 13 /* CMD13 = 0x4D */

Definition at line **207** of file **stm3210c_eval_sd.h**.

#define SD_CMD_SEND_WRITE_PROT 30 /* CMD30 = 0x5E */

Definition at line **217** of file **stm3210c_eval_sd.h**.

#define SD_CMD_SET_BLOCK_COUNT 23 /* CMD23 = 0x57 */

Definition at line **211** of file **stm3210c_eval_sd.h**.

#define SD_CMD_SET_BLOCKLEN 16 /* CMD16 = 0x50 */

Definition at line **208** of file **stm3210c_eval_sd.h**.

Referenced by **BSP_SD_ReadBlocks()**.

#define SD_CMD_SET_WRITE_PROT 28 /* CMD28 = 0x5C */

Definition at line 215 of file stm3210c_eval_sd.h.

#define SD_CMD_STOP_TRANSMISSION 12 /* CMD12 = 0x4C */

Definition at line 206 of file stm3210c_eval_sd.h.

#define SD_CMD_UNTAG_ERASE_GROUP 37 /* CMD37 = 0x65 */

Definition at line 223 of file stm3210c_eval_sd.h.

#define SD_CMD_UNTAG_SECTOR 34 /* CMD34 = 0x62 */

Definition at line 220 of file stm3210c_eval_sd.h.

#define SD_CMD_WRITE_MULT_BLOCK 25 /* CMD25 = 0x59 */

Definition at line 213 of file stm3210c_eval_sd.h.

#define SD_CMD_WRITE_SINGLE_BLOCK 24 /* CMD24 = 0x58 */

Definition at line 212 of file stm3210c_eval_sd.h.

Referenced by BSP_SD_WriteBlocks().

#define SD_NOT_PRESENT ((uint8_t)0x00)

Definition at line 197 of file stm3210c_eval_sd.h.

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

#define SD_PRESENT ((uint8_t)0x01)

SD detection on its memory slot.

Definition at line **196** of file **stm3210c_eval_sd.h**.

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

#define SD_START_DATA_MULTIPLE_BLOCK_READ 0xFE /* Data

Definition at line **188** of file **stm3210c_eval_sd.h**.

#define SD_START_DATA_MULTIPLE_BLOCK_WRITE 0xFD /* Dat

Definition at line **190** of file **stm3210c_eval_sd.h**.

#define SD_START_DATA_SINGLE_BLOCK_READ 0xFE /* Data to

Start Data tokens: Tokens (necessary because at nop/idle (and CS active) only 0xff is on the data/command line)

Definition at line **187** of file **stm3210c_eval_sd.h**.

Referenced by **BSP_SD_ReadBlocks()**, **SD_GetCIDRegister()**, and **SD_GetCSDRegister()**.

#define SD_START_DATA_SINGLE_BLOCK_WRITE 0xFE /* Data t

Definition at line **189** of file **stm3210c_eval_sd.h**.

Referenced by **BSP_SD_WriteBlocks()**.

#define SD_STOP_DATA_MULTIPLE_BLOCK_WRITE 0xFD /* Data

Definition at line **191** of file **stm3210c_eval_sd.h**.

STM3210C_EVAL BSP User Manual

Private_Defines

STM3210C_EVAL SD

Defines

#define	**SD_DUMMY_BYTE** 0xFF
#define	**SD_NO_RESPONSE_EXPECTED** 0x80

Define Documentation

#define SD_DUMMY_BYTE 0xFF

Definition at line **119** of file **stm3210c_eval_sd.c**.

Referenced by **BSP_SD_WriteBlocks()**, **SD_GetCIDRegister()**, and **SD_GetCSDRegister()**.

#define SD_NO_RESPONSE_EXPECTED 0x80

Definition at line **120** of file **stm3210c_eval_sd.c**.

STM3210C_EVAL BSP User Manual

Private_Variables

STM3210C_EVAL SD

Variables

__IO uint8_t **SdStatus** = **SD_PRESENT**

Variable Documentation

__IO uint8_t SdStatus = SD_PRESENT

Definition at line **140** of file **stm3210c_eval_sd.c**.

Referenced by **BSP_SD_Init()**.

STM3210C_EVAL BSP User Manual

Private_Variables

STM3210C_EVAL Touch Screen

Variables

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

Variable Documentation

TS_DrvTypeDef* ts_driver [static]

Definition at line 123 of file stm3210c_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 125 of file stm3210c_eval_ts.c.

Referenced by BSP_TS_GetState(), and BSP_TS_Init().

uint16_t ts_x_boundary [static]

Definition at line 124 of file stm3210c_eval_ts.c.

Referenced by BSP_TS_GetState(), and BSP_TS_Init().

uint16_t ts_y_boundary [static]

Definition at line 124 of file stm3210c_eval_ts.c.

Referenced by BSP_TS_GetState(), and BSP_TS_Init().

STM3210C_EVAL BSP User Manual

Exported_Constants

STM3210C_EVAL Touch Screen

Defines

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

Enumerations

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

Define Documentation

#define TS_SWAP_NONE 0x00

Definition at line 87 of file stm3210c_eval_ts.h.

#define TS_SWAP_X 0x01

Definition at line 88 of file stm3210c_eval_ts.h.

Referenced by BSP_TS_GetState().

#define TS_SWAP_XY 0x04

Definition at line 90 of file stm3210c_eval_ts.h.

Referenced by BSP_TS_GetState(), and BSP_TS_Init().

#define TS_SWAP_Y 0x02

Definition at line 89 of file stm3210c_eval_ts.h.

Referenced by BSP_TS_GetState().

Enumeration Type Documentation

enum TS_StatusTypeDef

Enumerator:

TS_OK

TS_ERROR

TS_TIMEOUT

Definition at line **92** of file **stm3210c_eval_ts.h**.

STM3210C_EVAL BSP User Manual

Drivers Directory Reference

Directories

directory	**BSP**

STM3210C_EVAL BSP User Manual

BSP Directory Reference

Directories

directory	**STM3210C_EVAL**

STM3210C_EVAL BSP User Manual

STM3210C_EVAL Directory Reference

Files

file **stm3210c_eval.c** [code]

This file provides a set of firmware functions to manage Leds, push-button and COM ports for STM3210C_EVAL.

file **stm3210c_eval.h** [code]

This file contains definitions for STM3210C_EVAL's LEDs, push-buttons and COM ports hardware resources.

file **stm3210c_eval_accelerometer.c** [code]

This file provides a set of functions needed to manage the ACCELEROMETER MEMS accelerometer available on STM3210C_EVAL board.

file **stm3210c_eval_accelerometer.h** [code]

This file contains all the functions prototypes for the **stm3210c_eval_accelerometer.c** firmware driver.

file **stm3210c_eval_audio.c** [code]

This file provides the Audio driver for the STM3210C-Eval board.

file **stm3210c_eval_audio.h** [code]

This file contains the common defines and functions prototypes for **stm3210c_eval_audio.c** driver.

file **stm3210c_eval_eeprom.c** [code]

This file provides a set of functions needed to manage a M24C64 I2C EEPROM memory.

file **stm3210c_eval_eeprom.h** [code]

This file contains all the functions prototypes for the **stm3210c_eval_eeprom.c** firmware driver.

file **stm3210c_eval_io.c** [code]

This file provides a set of functions needed to manage the IO pins on STM3210C-EVAL evaluation board.

file **stm3210c_eval_io.h** [code]

This file contains the common defines and functions prototypes for the **stm3210c_eval_io.c** driver.

file **stm3210c_eval_lcd.c** [code]

This file includes the driver for Liquid Crystal Display (LCD) module mounted on STM3210C-EVAL evaluation board.

file **stm3210c_eval_lcd.h** [code]

This file contains the common defines and functions prototypes for the **stm3210c_eval_lcd.c** driver.

file **stm3210c_eval_sd.c** [code]

This file provides a set of functions needed to manage the SPI SD Card memory mounted on STM3210C-EVAL board. It implements a high level communication layer for read and write from/to this memory. The needed STM32F10x hardware resources (SPI and GPIO) are defined in **stm3210c_eval.h** file, and the initialization is performed in **SD_IO_Init()** function declared in **stm3210c_eval.c** file. You can easily tailor this driver to any other development board, by just adapting the defines for hardware resources and **SD_IO_Init()** function.

file **stm3210c_eval_sd.h** [code]

This file contains the common defines and functions prototypes for the **stm3210c_eval_sd.c** driver.

file **stm3210c_eval_ts.c** [code]

This file provides a set of functions needed to manage the touch screen on STM3210C_EVAL evaluation board.

file **stm3210c_eval_ts.h** [code]

This file contains the common defines and functions prototypes for the **stm3210c_eval_ts.c** driver.

STM3210C_EVAL BSP User Manual

stm3210c_eval.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210c_eval.h
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file contains definitions for STM3210C_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 /** @addtogroup BSP
00040   * @{
00041   */
00042
00043 /** @addtogroup STM3210C_EVAL
00044   * @{
00045   */
00046
00047 /** @addtogroup STM3210C_EVAL_COMMON
00048   * @{
00049   */
00050
00051 /* Define to prevent recursive inclusion ---
-----------------------------------*/
00052 #ifndef __STM3210C_EVAL_H
00053 #define __STM3210C_EVAL_H
00054
00055 #ifdef __cplusplus
00056  extern "C" {
00057 #endif
00058
00059 /* Includes ------------------------------
-----------------------------------*/
00060 #include "stm32f1xx_hal.h"
00061 #ifdef HAL_I2C_MODULE_ENABLED
00062 #include "stm3210c_eval_io.h"
00063 #endif /* HAL_I2C_MODULE_ENABLED */
00064
00065 /** @defgroup STM3210C_EVAL_Exported_Types E
xported Types
00066   * @{
00067   */
```

```
00068
00069 /**
00070  * @brief LED Types Definition
00071  */
00072 typedef enum
00073 {
00074   LED1 = 0,
00075   LED2 = 1,
00076   LED3 = 2,
00077   LED4 = 3,
00078
00079   LED_GREEN  = LED1,
00080   LED_ORANGE = LED2,
00081   LED_RED    = LED3,
00082   LED_BLUE   = LED4
00083
00084 } Led_TypeDef;
00085
00086 /**
00087  * @brief BUTTON Types Definition
00088  */
00089 typedef enum
00090 {
00091   BUTTON_WAKEUP = 0,
00092   BUTTON_TAMPER = 1,
00093   BUTTON_KEY    = 2,
00094
00095 } Button_TypeDef;
00096
00097 typedef enum
00098 {
00099   BUTTON_MODE_GPIO = 0,
00100   BUTTON_MODE_EXTI = 1
00101 } ButtonMode_TypeDef;
00102
00103 /**
00104  * @brief JOYSTICK Types Definition
```

```
00105  */
00106 typedef enum
00107 {
00108   JOY_SEL   = 0,
00109   JOY_LEFT  = 1,
00110   JOY_RIGHT = 2,
00111   JOY_DOWN  = 3,
00112   JOY_UP    = 4,
00113   JOY_NONE  = 5
00114
00115 }JOYState_TypeDef;
00116
00117 typedef enum
00118 {
00119   JOY_MODE_GPIO = 0,
00120   JOY_MODE_EXTI = 1
00121
00122 }JOYMode_TypeDef;
00123
00124 /**
00125  * @brief COM Types Definition
00126  */
00127 typedef enum
00128 {
00129   COM1 = 0,
00130   COM2 = 1
00131 } COM_TypeDef;
00132 /**
00133   * @}
00134   */
00135
00136 /** @defgroup STM3210C_EVAL_Exported_Constan
ts Exported Constants
00137   * @{
00138   */
00139
00140 /**
```

```
00141   * @brief  Define for STM3210C_EVAL board
00142   */
00143 #if !defined (USE_STM3210C_EVAL)
00144  #define USE_STM3210C_EVAL
00145 #endif
00146
00147 /** @addtogroup STM3210C_EVAL_LED
00148   * @{
00149   */
00150 #define LEDn                                    4
00151
00152 #define LED1_PIN                                GPI
O_PIN_7             /* PD.07*/
00153 #define LED1_GPIO_PORT                          GPI
OD
00154 #define LED1_GPIO_CLK_ENABLE()                  __H
AL_RCC_GPIOD_CLK_ENABLE()
00155 #define LED1_GPIO_CLK_DISABLE()                 __H
AL_RCC_GPIOD_CLK_DISABLE()
00156
00157 #define LED2_PIN                                GPI
O_PIN_13            /* PD.13*/
00158 #define LED2_GPIO_PORT                          GPI
OD
00159 #define LED2_GPIO_CLK_ENABLE()                  __H
AL_RCC_GPIOD_CLK_ENABLE()
00160 #define LED2_GPIO_CLK_DISABLE()                 __H
AL_RCC_GPIOD_CLK_DISABLE()
00161
00162
00163 #define LED3_PIN                                GPI
O_PIN_3             /* PD.03*/
00164 #define LED3_GPIO_PORT                          GPI
OD
00165 #define LED3_GPIO_CLK_ENABLE()                  __H
AL_RCC_GPIOD_CLK_ENABLE()
00166 #define LED3_GPIO_CLK_DISABLE()                 __H
```

```
AL_RCC_GPIOD_CLK_DISABLE()
00167
00168
00169 #define LED4_PIN                                GPI
O_PIN_4             /* PD.04*/
00170 #define LED4_GPIO_PORT                          GPI
OD
00171 #define LED4_GPIO_CLK_ENABLE()                  __H
AL_RCC_GPIOD_CLK_ENABLE()
00172 #define LED4_GPIO_CLK_DISABLE()                 __H
AL_RCC_GPIOD_CLK_DISABLE()
00173
00174 #define LEDx_GPIO_CLK_ENABLE(__LED__)    do
{ if ((__LED__) == LED1) LED1_GPIO_CLK_ENABLE(); e
lse \
00175
  if ((__LED__) == LED2) LED2_GPIO_CLK_ENABLE(); e
lse \
00176
  if ((__LED__) == LED3) LED3_GPIO_CLK_ENABLE(); e
lse \
00177
  if ((__LED__) == LED4) LED4_GPIO_CLK_ENABLE();}
while(0)
00178
00179 #define LEDx_GPIO_CLK_DISABLE(__LED__)   (((
__LED__) == LED1) ? LED1_GPIO_CLK_DISABLE() :\
00180                                             ((
__LED__) == LED2) ? LED2_GPIO_CLK_DISABLE() :\
00181                                             ((
__LED__) == LED3) ? LED3_GPIO_CLK_DISABLE() :\
00182                                             ((
__LED__) == LED4) ? LED4_GPIO_CLK_DISABLE() : 0 )
00183
00184 /**
00185   * @}
00186   */
```

```
00187
00188 /** @addtogroup STM3210C_EVAL_BUTTON
00189   * @{
00190   */
00191 #define BUTTONn                          3
00192
00193 /**
00194  * @brief Tamper push-button
00195  */
00196 #define TAMPER_BUTTON_PIN
GPIO_PIN_13             /* PC.13*/
00197 #define TAMPER_BUTTON_GPIO_PORT
GPIOC
00198 #define TAMPER_BUTTON_GPIO_CLK_ENABLE()
__HAL_RCC_GPIOC_CLK_ENABLE()
00199 #define TAMPER_BUTTON_GPIO_CLK_DISABLE()
__HAL_RCC_GPIOC_CLK_DISABLE()
00200 #define TAMPER_BUTTON_EXTI_IRQn
EXTI15_10_IRQn
00201
00202 /**
00203  * @brief Key push-button
00204  */
00205 #define KEY_BUTTON_PIN
GPIO_PIN_9              /* PB.09*/
00206 #define KEY_BUTTON_GPIO_PORT
GPIOB
00207 #define KEY_BUTTON_GPIO_CLK_ENABLE()
__HAL_RCC_GPIOB_CLK_ENABLE()
00208 #define KEY_BUTTON_GPIO_CLK_DISABLE()
__HAL_RCC_GPIOB_CLK_DISABLE()
00209 #define KEY_BUTTON_EXTI_IRQn
EXTI9_5_IRQn
00210
00211 /**
00212  * @brief Wake-up push-button
00213  */
```

```
00214 #define WAKEUP_BUTTON_PIN
GPIO_PIN_0             /* PA.00*/
00215 #define WAKEUP_BUTTON_GPIO_PORT
GPIOA
00216 #define WAKEUP_BUTTON_GPIO_CLK_ENABLE()
__HAL_RCC_GPIOA_CLK_ENABLE()
00217 #define WAKEUP_BUTTON_GPIO_CLK_DISABLE()
__HAL_RCC_GPIOA_CLK_DISABLE()
00218 #define WAKEUP_BUTTON_EXTI_IRQn
EXTI0_IRQn
00219
00220 #define BUTTONx_GPIO_CLK_ENABLE(__BUTTON__)
do { if ((__BUTTON__) == BUTTON_TAMPER) TAMPER_BUT
TON_GPIO_CLK_ENABLE()  ; else \
00221
     if ((__BUTTON__) == BUTTON_KEY) KEY_BUTTON_GP
IO_CLK_ENABLE()  ; else \
00222
     if ((__BUTTON__) == BUTTON_WAKEUP) WAKEUP_BUT
TON_GPIO_CLK_ENABLE();} while(0)
00223
00224 #define BUTTONx_GPIO_CLK_DISABLE(__BUTTON__)
    (((__BUTTON__) == BUTTON_TAMPER) TAMPER_BUTTON
_GPIO_CLK_DISABLE()  :\
00225
     ((__BUTTON__) == BUTTON_KEY) KEY_BUTTON_GPIO_
CLK_DISABLE()  :\
00226
     ((__BUTTON__) == BUTTON_WAKEUP) WAKEUP_BUTTON
_GPIO_CLK_DISABLE()  : 0 )
00227
00228 /**
00229   * @brief IO Pins definition
00230   */
00231 /* Joystick */
00232 #define JOY_SEL_PIN                    (IO2_PI
N_7) /* IO_Expander_2 */
```

```
00233 #define JOY_DOWN_PIN                    (IO2_PI
N_6) /* IO_Expander_2 */
00234 #define JOY_LEFT_PIN                    (IO2_PI
N_5) /* IO_Expander_2 */
00235 #define JOY_RIGHT_PIN                   (IO2_PI
N_4) /* IO_Expander_2 */
00236 #define JOY_UP_PIN                      (IO2_PI
N_3) /* IO_Expander_2 */
00237 #define JOY_NONE_PIN                    JOY_ALL
_PINS
00238 #define JOY_ALL_PINS                    (JOY_SE
L_PIN | JOY_DOWN_PIN | JOY_LEFT_PIN | JOY_RIGHT_PI
N | JOY_UP_PIN)
00239
00240 /* MEMS */
00241 #define MEMS_INT1_PIN                   (IO1_PI
N_3) /* IO_Expander_1 */ /* Input */
00242 #define MEMS_INT2_PIN                   (IO1_PI
N_2) /* IO_Expander_1 */ /* Input */
00243 #define MEMS_ALL_PINS                   (MEMS_I
NT1_PIN | MEMS_INT2_PIN)
00244
00245 #define AUDIO_RESET_PIN                 (IO2_PI
N_2) /* IO_Expander_2 */ /* Output */
00246 #define MII_INT_PIN                     (IO2_PI
N_0) /* IO_Expander_2 */ /* Output */
00247 #define VBAT_DIV_PIN                    (IO1_PI
N_0) /* IO_Expander_1 */ /* Output */
00248
00249 /**
00250   * @}
00251   */
00252
00253 /** @addtogroup STM3210C_EVAL_COM
00254   * @{
00255   */
00256 #define COMn                                1
```

```
00257
00258 /**
00259  * @brief Definition for COM port1, connecte
d to USART2
00260  */
00261 #define EVAL_COM1                           USA
RT2
00262 #define EVAL_COM1_CLK_ENABLE()              __H
AL_RCC_USART2_CLK_ENABLE()
00263 #define EVAL_COM1_CLK_DISABLE()             __H
AL_RCC_USART2_CLK_DISABLE()
00264
00265 #define AFIOCOM1_CLK_ENABLE()               __H
AL_RCC_AFIO_CLK_ENABLE()
00266 #define AFIOCOM1_CLK_DISABLE()              __H
AL_RCC_AFIO_CLK_DISABLE()
00267
00268 #define EVAL_COM1_TX_PIN                    GPI
O_PIN_5             /* PD.05*/
00269 #define EVAL_COM1_TX_GPIO_PORT              GPI
OD
00270 #define EVAL_COM1_TX_GPIO_CLK_ENABLE()   __H
AL_RCC_GPIOD_CLK_ENABLE()
00271 #define EVAL_COM1_TX_GPIO_CLK_DISABLE()  __H
AL_RCC_GPIOD_CLK_DISABLE()
00272
00273 #define EVAL_COM1_RX_PIN                    GPI
O_PIN_6             /* PD.06*/
00274 #define EVAL_COM1_RX_GPIO_PORT              GPI
OD
00275 #define EVAL_COM1_RX_GPIO_CLK_ENABLE()   __H
AL_RCC_GPIOD_CLK_ENABLE()
00276 #define EVAL_COM1_RX_GPIO_CLK_DISABLE()  __H
AL_RCC_GPIOD_CLK_DISABLE()
00277
00278 #define EVAL_COM1_IRQn                      USA
RT2_IRQn
```

```
00279
00280 #define COMx_CLK_ENABLE(__INDEX__)
    do { if((__INDEX__) == COM1) EVAL_COM1_CLK_ENA
BLE();} while(0)
00281 #define COMx_CLK_DISABLE(__INDEX__)
    (((__INDEX__) == COM1) ? EVAL_COM1_CLK_DISABLE
() : 0)
00282
00283 #define AFIOCOMx_CLK_ENABLE(__INDEX__)
    do { if((__INDEX__) == COM1) AFIOCOM1_CLK_ENAB
LE();} while(0)
00284 #define AFIOCOMx_CLK_DISABLE(__INDEX__)
    (((__INDEX__) == COM1) ? AFIOCOM1_CLK_DISABLE(
) : 0)
00285
00286 #define AFIOCOMx_REMAP(__INDEX__)
    (((__INDEX__) == COM1) ? (AFIO->MAPR |= (AFIO_
MAPR_USART2_REMAP)) : 0)
00287
00288 #define COMx_TX_GPIO_CLK_ENABLE(__INDEX__)
    do { if((__INDEX__) == COM1) EVAL_COM1_TX_GPIO
_CLK_ENABLE();} while(0)
00289 #define COMx_TX_GPIO_CLK_DISABLE(__INDEX__)
    (((__INDEX__) == COM1) ? EVAL_COM1_TX_GPIO_CLK
_DISABLE() : 0)
00290
00291 #define COMx_RX_GPIO_CLK_ENABLE(__INDEX__)
    do { if((__INDEX__) == COM1) EVAL_COM1_RX_GPIO
_CLK_ENABLE();} while(0)
00292 #define COMx_RX_GPIO_CLK_DISABLE(__INDEX__)
    (((__INDEX__) == COM1) ? EVAL_COM1_RX_GPIO_CLK
_DISABLE() : 0)
00293
00294 /**
00295   * @}
00296   */
00297
```

```
00298 /** @addtogroup STM3210C_EVAL_BUS
00299   * @{
00300   */
00301
00302 /**
00303   * @brief  IO Expander Interrupt line on EX
TI
00304   */
00305 #define IOE_IT_PIN                        GPI
O_PIN_14
00306 #define IOE_IT_GPIO_PORT                  GPI
OB
00307 #define IOE_IT_GPIO_CLK_ENABLE()          __H
AL_RCC_GPIOB_CLK_ENABLE()
00308 #define IOE_IT_GPIO_CLK_DISABLE()         __H
AL_RCC_GPIOB_CLK_DISABLE()
00309 #define IOE_IT_EXTI_IRQn                  EXT
I15_10_IRQn
00310 #define IOE_IT_EXTI_IRQHANDLER            EXT
I15_10_IRQHandler
00311
00312 /* Exported constant IO ------------------
-----------------------------------*/
00313 #define IO1_I2C_ADDRESS
  0x82
00314 #define IO2_I2C_ADDRESS
  0x88
00315 #define TS_I2C_ADDRESS
  0x82
00316
00317 /*The Slave ADdress (SAD) associated to the
LIS302DL is 001110xb. SDO pad can be used
00318 to modify less significant bit of the device
 address. If SDO pad is connected to voltage
00319 supply LSb is 1 (address 0011101b) else if
 SDO pad is connected to ground LSb value is
00320 0 (address 0011100b).*/
```

```
00321 #define L1S302DL_I2C_ADDRESS
  0x38
00322
00323
00324 /*###################### ACCELEROMETER ######
######################*/
00325 /* Read/Write command */
00326 #define READWRITE_CMD                     ((
uint8_t)0x80)
00327 /* Multiple byte read/write command */
00328 #define MULTIPLEBYTE_CMD                  ((
uint8_t)0x40)
00329
00330 /*###################### I2Cx ###############
######################*/
00331 /* User can use this section to tailor I2Cx
instance used and associated
00332    resources */
00333 /* Definition for I2Cx Pins */
00334 #define EVAL_I2Cx_SCL_PIN
    GPIO_PIN_6        /* PB.06*/
00335 #define EVAL_I2Cx_SCL_GPIO_PORT
    GPIOB
00336 #define EVAL_I2Cx_SDA_PIN
    GPIO_PIN_7        /* PB.07*/
00337 #define EVAL_I2Cx_SDA_GPIO_PORT
    GPIOB
00338
00339 /* Definition for I2Cx clock resources */
00340 #define EVAL_I2Cx
    I2C1
00341 #define EVAL_I2Cx_CLK_ENABLE()
    __HAL_RCC_I2C1_CLK_ENABLE()
00342 #define EVAL_I2Cx_SDA_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOB_CLK_ENABLE()
00343 #define EVAL_I2Cx_SCL_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOB_CLK_ENABLE()
```

```
00344
00345 #define EVAL_I2Cx_FORCE_RESET()
    __HAL_RCC_I2C1_FORCE_RESET()
00346 #define EVAL_I2Cx_RELEASE_RESET()
    __HAL_RCC_I2C1_RELEASE_RESET()
00347
00348 /* Definition for I2Cx's NVIC */
00349 #define EVAL_I2Cx_EV_IRQn
    I2C1_EV_IRQn
00350 #define EVAL_I2Cx_EV_IRQHandler
    I2C1_EV_IRQHandler
00351 #define EVAL_I2Cx_ER_IRQn
    I2C1_ER_IRQn
00352 #define EVAL_I2Cx_ER_IRQHandler
    I2C1_ER_IRQHandler
00353
00354 /* I2C clock speed configuration (in Hz) */
00355 #ifndef BSP_I2C_SPEED
00356  #define BSP_I2C_SPEED
      400000
00357 #endif /* I2C_SPEED */
00358
00359
00360 /* Maximum Timeout values for flags waiting
loops. These timeouts are not based
00361    on accurate values, they just guarantee t
hat the application will not remain
00362    stuck if the I2C communication is corrupt
ed.
00363    You may modify these timeout values depen
ding on CPU frequency and application
00364    conditions (interrupts routines ...). */

00365 #define EVAL_I2Cx_TIMEOUT_MAX
    3000
00366
00367 /*###################### SPI3 ###############
```

```
####################*/
00368 #define EVAL_SPIx
    SPI3
00369 #define EVAL_SPIx_CLK_ENABLE()
    __HAL_RCC_SPI3_CLK_ENABLE()
00370
00371 #define EVAL_SPIx_SCK_GPIO_PORT
    GPIOC              /* PC.10*/
00372 #define EVAL_SPIx_SCK_PIN
    GPIO_PIN_10
00373 #define EVAL_SPIx_SCK_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOC_CLK_ENABLE()
00374 #define EVAL_SPIx_SCK_GPIO_CLK_DISABLE()
    __HAL_RCC_GPIOC_CLK_DISABLE()
00375
00376 #define EVAL_SPIx_MISO_MOSI_GPIO_PORT
    GPIOC
00377 #define EVAL_SPIx_MISO_MOSI_GPIO_CLK_ENABLE(
)   __HAL_RCC_GPIOC_CLK_ENABLE()
00378 #define EVAL_SPIx_MISO_MOSI_GPIO_CLK_DISABLE
()  __HAL_RCC_GPIOC_CLK_DISABLE()
00379 #define EVAL_SPIx_MISO_PIN
    GPIO_PIN_11        /* PC.11*/
00380 #define EVAL_SPIx_MOSI_PIN
    GPIO_PIN_12        /* PC.12*/
00381 /* Maximum Timeout values for flags waiting
loops. These timeouts are not based
00382    on accurate values, they just guarantee t
hat the application will not remain
00383    stuck if the SPI communication is corrupt
ed.
00384    You may modify these timeout values depen
ding on CPU frequency and application
00385    conditions (interrupts routines ...). */

00386 #define EVAL_SPIx_TIMEOUT_MAX
    1000
```

```
00387
00388 /**
00389   * @}
00390   */
00391
00392 /** @addtogroup STM3210C_EVAL_COMPONENT
00393   * @{
00394   */
00395
00396 /*#################### LCD ################
####################*/
00397 /* Chip Select macro definition */
00398 #define LCD_CS_LOW()       HAL_GPIO_WritePin
(LCD_NCS_GPIO_PORT, LCD_NCS_PIN, GPIO_PIN_RESET)
00399 #define LCD_CS_HIGH()      HAL_GPIO_WritePin
(LCD_NCS_GPIO_PORT, LCD_NCS_PIN, GPIO_PIN_SET)
00400
00401 /**
00402   * @brief  LCD Control Interface pins
00403   */
00404 #define LCD_NCS_PIN
    GPIO_PIN_2         /* PB.02*/
00405 #define LCD_NCS_GPIO_PORT
    GPIOB
00406 #define LCD_NCS_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOB_CLK_ENABLE()
00407 #define LCD_NCS_GPIO_CLK_DISABLE()
    __HAL_RCC_GPIOB_CLK_DISABLE()
00408
00409 /*#################### SD ################
####################*/
00410 /* Chip Select macro definition */
00411 #define SD_CS_LOW()       HAL_GPIO_WritePin(
SD_CS_GPIO_PORT, SD_CS_PIN, GPIO_PIN_RESET)
00412 #define SD_CS_HIGH()      HAL_GPIO_WritePin(
SD_CS_GPIO_PORT, SD_CS_PIN, GPIO_PIN_SET)
00413
```

```
00414 /**
00415   * @brief  SD Control Interface pins
00416   */
00417 #define SD_CS_PIN
    GPIO_PIN_4         /* PA.04*/
00418 #define SD_CS_GPIO_PORT
    GPIOA
00419 #define SD_CS_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOA_CLK_ENABLE()
00420 #define SD_CS_GPIO_CLK_DISABLE()
    __HAL_RCC_GPIOA_CLK_DISABLE()
00421
00422 /**
00423   * @brief  SD Detect Interface pins
00424   */
00425 #define SD_DETECT_PIN
    GPIO_PIN_0
00426 #define SD_DETECT_GPIO_PORT
    GPIOE
00427 #define SD_DETECT_GPIO_CLK_ENABLE()
    __HAL_RCC_GPIOE_CLK_ENABLE()
00428 #define SD_DETECT_GPIO_CLK_DISABLE()
    __HAL_RCC_GPIOE_CLK_DISABLE()
00429 #define SD_DETECT_EXTI_IRQn
    EXTI0_IRQn
00430
00431 /*###################### AUDIO ##############
#############*/
00432 /**
00433   * @brief  AUDIO I2C Interface pins
00434   */
00435 #define AUDIO_I2C_ADDRESS
  0x94
00436
00437 /**
00438   * @}
00439   */
```

```
00440
00441 /**
00442   * @}
00443   */
00444
00445
00446
00447 /** @addtogroup STM3210C_EVAL_Exported_Funct
ions
00448   * @{
00449   */
00450 uint32_t                BSP_GetVersion(void)
;
00451 void                    BSP_LED_Init(Led_Typ
eDef Led);
00452 void                    BSP_LED_On(Led_TypeD
ef Led);
00453 void                    BSP_LED_Off(Led_Type
Def Led);
00454 void                    BSP_LED_Toggle(Led_T
ypeDef Led);
00455 void                    BSP_PB_Init(Button_T
ypeDef Button, ButtonMode_TypeDef Button_Mode);
00456 uint32_t                BSP_PB_GetState(Butt
on_TypeDef Button);
00457 #ifdef HAL_UART_MODULE_ENABLED
00458 void                    BSP_COM_Init(COM_Typ
eDef COM, UART_HandleTypeDef* huart);
00459 #endif /* HAL_UART_MODULE_ENABLED */
00460 #ifdef HAL_I2C_MODULE_ENABLED
00461 uint8_t                 BSP_JOY_Init(JOYMode
_TypeDef Joy_Mode);
00462 JOYState_TypeDef        BSP_JOY_GetState(void
);
00463 #endif /* HAL_I2C_MODULE_ENABLED */
00464
00465 /**
```

```
00466   * @}
00467   */
00468
00469
00470 #ifdef __cplusplus
00471 }
00472 #endif
00473
00474 #endif /* __STM3210C_EVAL_H */
00475
00476 /**
00477   * @}
00478   */
00479
00480 /**
00481   * @}
00482   */
00483
00484 /**
00485   * @}
00486   */
00487
00488 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

STM3210C_EVAL BSP User Manual

stm3210c_eval_io.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210c_eval_io.h
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          the stm3210c_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 __STM3210C_EVAL_IO_H
00041 #define __STM3210C_EVAL_IO_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------------------------------------------*/
00048 #include "stm3210c_eval.h"
00049 #include "../Components/stmpe811/stmpe811.h"
00050
00051
00052 /** @addtogroup BSP
00053   * @{
00054   */
00055
00056 /** @addtogroup STM3210C_EVAL
00057   * @{
00058   */
00059
00060 /** @addtogroup STM3210C_EVAL_IO
00061   * @{
00062   */
00063
00064 /* Exported types ------------------------------------------------------------*/
00065
00066 /** @defgroup STM3210C_EVAL_IO_Exported_Types Exported_Types
00067   * @{
```

```
00068   */
00069 typedef enum
00070 {
00071   IO_OK         = 0x00,
00072   IO_ERROR      = 0x01,
00073   IO_TIMEOUT    = 0x02
00074
00075 }IO_StatusTypeDef;
00076
00077 /**
00078   * @}
00079   */
00080
00081 /** @defgroup STM3210C_EVAL_IO_Exported_Cons
tants Exported_Constants
00082   * @{
00083   */
00084 /* Virtual pin offset STMPE811, IOExpander1
*/
00085 #define IO1_PIN_OFFSET                0
00086 /* Virtual pin offset STMPE811, IOExpander2
*/
00087 #define IO2_PIN_OFFSET                8
00088
00089 /* Pins definition STMPE811, IOExpander1 */
00090 #define IO1_PIN_0                      (uint3
2_t)(0x00000001 << IO1_PIN_OFFSET)
00091 #define IO1_PIN_1                      (uint3
2_t)(0x00000002 << IO1_PIN_OFFSET)
00092 #define IO1_PIN_2                      (uint3
2_t)(0x00000004 << IO1_PIN_OFFSET)
00093 #define IO1_PIN_3                      (uint3
2_t)(0x00000008 << IO1_PIN_OFFSET)
00094 #define IO1_PIN_4                      (uint3
2_t)(0x00000010 << IO1_PIN_OFFSET)
00095 #define IO1_PIN_5                      (uint3
2_t)(0x00000020 << IO1_PIN_OFFSET)
```

```
00096 #define IO1_PIN_6                        (uint3
2_t)(0x00000040 << IO1_PIN_OFFSET)
00097 #define IO1_PIN_7                        (uint3
2_t)(0x00000080 << IO1_PIN_OFFSET)
00098 #define IO1_PIN_ALL                      (uint3
2_t)(0x000000FF << IO1_PIN_OFFSET)
00099
00100 /* Pins definition STMPE16000, IOExpander2 */

00101 #define IO2_PIN_0                        (uint3
2_t)(0x00000001 << IO2_PIN_OFFSET)
00102 #define IO2_PIN_1                        (uint3
2_t)(0x00000002 << IO2_PIN_OFFSET)
00103 #define IO2_PIN_2                        (uint3
2_t)(0x00000004 << IO2_PIN_OFFSET)
00104 #define IO2_PIN_3                        (uint3
2_t)(0x00000008 << IO2_PIN_OFFSET)
00105 #define IO2_PIN_4                        (uint3
2_t)(0x00000010 << IO2_PIN_OFFSET)
00106 #define IO2_PIN_5                        (uint3
2_t)(0x00000020 << IO2_PIN_OFFSET)
00107 #define IO2_PIN_6                        (uint3
2_t)(0x00000040 << IO2_PIN_OFFSET)
00108 #define IO2_PIN_7                        (uint3
2_t)(0x00000080 << IO2_PIN_OFFSET)
00109 #define IO2_PIN_ALL                      (uint3
2_t)(0x000000FF << IO2_PIN_OFFSET)
00110
00111 /**
00112   * @}
00113   */
00114
00115
00116 /* Exported macro ---------------------------
-----------------------------------*/
00117
00118 /** @defgroup STM3210C_EVAL_IO_Exported_Macr
```

```
os Exported_Macros
00119   * @{
00120   */
00121
00122 /**
00123   * @}
00124   */
00125
00126 /* Exported functions ----------------------
---------------------------------*/
00127
00128 /** @addtogroup STM3210C_EVAL_IO_Exported_Fu
nctions
00129   * @{
00130   */
00131
00132 uint8_t  BSP_IO_Init(void);
00133 void     BSP_IO_ITClear(uint32_t IO_Pin);
00134 uint32_t BSP_IO_ITGetStatus(uint32_t IO_Pin)
;
00135 uint8_t  BSP_IO_ConfigPin(uint32_t IO_Pin, I
O_ModeTypedef IO_Mode);
00136 void     BSP_IO_WritePin(uint32_t IO_Pin, ui
nt8_t PinState);
00137 uint32_t BSP_IO_ReadPin(uint32_t IO_Pin);
00138 void     BSP_IO_TogglePin(uint32_t IO_Pin);
00139
00140 #ifdef __cplusplus
00141 }
00142 #endif
00143 #endif /* __STM3210C_EVAL_IO_H */
00144
00145 /**
00146   * @}
00147   */
00148
00149 /**
```

```
00150   * @}
00151   */
00152
00153 /**
00154   * @}
00155   */
00156
00157 /**
00158   * @}
00159   */
00160 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210C_EVAL BSP User Manual

stm3210c_eval_accelerometer.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210c_eval_accelerometer.h
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file contains all the functions prototypes for the stm3210c_eval_accelerometer.c
00008   *          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 __STM3210C_EVAL_ACCELEROMETER_H
00041 #define __STM3210C_EVAL_ACCELEROMETER_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------
-----------------------------------*/
00048 #include "stm3210c_eval.h"
00049 /* Include Accelerometer component driver */
00050 #include "../Components/lis302dl/lis302dl.h"

00051
00052    /** @addtogroup BSP
00053   * @{
00054   */
00055
00056 /** @addtogroup STM3210C_EVAL
00057   * @{
00058   */
00059
00060 /** @addtogroup STM3210C_EVAL_ACCELEROMETER
00061   * @{
00062   */
00063
00064
00065 /** @defgroup STM3210C_EVAL_ACCELEROMETER_Ex
ported_Types Exported Types
00066   * @{
```

```
00067   */
00068 typedef enum
00069 {
00070   ACCELERO_OK = 0,
00071   ACCELERO_ERROR = 1,
00072   ACCELERO_TIMEOUT = 2
00073 }
00074 ACCELERO_StatusTypeDef;
00075
00076 /**
00077   * @}
00078   */
00079
00080 /** @addtogroup STM3210C_EVAL_ACCELEROMETER_
Exported_Functions
00081   * @{
00082   */
00083 /* Acc functions */
00084 uint8_t   BSP_ACCELERO_Init(void);
00085 uint8_t   BSP_ACCELERO_ReadID(void);
00086 void      BSP_ACCELERO_Reset(void);
00087 void      BSP_ACCELERO_Click_ITConfig(void);
00088 void      BSP_ACCELERO_Click_ITClear(void);
00089 void      BSP_ACCELERO_GetXYZ(int16_t *pData
XYZ);
00090
00091 #endif /* __STM3210C_EVAL_ACCELEROMETER_H */
00092 /**
00093   * @}
00094   */
00095
00096 /**
00097   * @}
00098   */
00099
00100 /**
00101   * @}
```

```
00102   */
00103
00104 /**
00105   * @}
00106   */
00107
00108
00109 /******************** (C) COPYRIGHT 2011 STMi
croelectronics *****END OF FILE****/
```

STM3210C_EVAL BSP User Manual

stm3210c_eval_accelerometer.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210c_eval_accelerometer.c
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file provides a set of functions needed to manage the ACCELEROMETER
00008   *          MEMS accelerometer available on STM3210C_EVAL 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 /* Includes ------------------------------------------------------------------*/
00040 #include "stm3210c_eval_accelerometer.h"
00041
00042 /** @addtogroup BSP
00043   * @{
00044   */
00045
00046 /** @addtogroup STM3210C_EVAL
00047   * @{
00048   */
00049
00050 /** @addtogroup STM3210C_EVAL_ACCELEROMETER
00051   * @brief  This file includes the motion sensor driver for ACCELEROMETER motion sensor
00052   *         devices.
00053   * @{
00054   */
00055
00056 /** @defgroup STM3210C_EVAL_ACCELEROMETER_Private_TypesDefinitions Private Types Definitions
00057   * @{
00058   */
00059 /**
00060   * @}
00061   */
00062
00063 /** @defgroup STM3210C_EVAL_ACCELEROMETER_Private_Defines Private Defines
00064   * @{
00065   */
00066
```

```
00067 /**
00068   * @}
00069   */
00070
00071 /** @defgroup STM3210C_EVAL_ACCELEROMETER_Pr
ivate_Macros Private Macros
00072   * @{
00073   */
00074
00075 /**
00076   * @}
00077   */
00078
00079 /** @defgroup STM3210C_EVAL_ACCELEROMETER_Pr
ivate_Variables Private Variables
00080   * @{
00081   */
00082 static ACCELERO_DrvTypeDef *AcceleroDrv;
00083
00084 /**
00085   * @}
00086   */
00087
00088 /** @defgroup STM3210C_EVAL_ACCELEROMETER_Pr
ivate_FunctionPrototypes Private FunctionPrototypes

00089   * @{
00090   */
00091
00092 /**
00093   * @}
00094   */
00095
00096 /** @defgroup STM3210C_EVAL_ACCELEROMETER_Ex
ported_Functions Exported Functions
00097   * @{
00098   */
```

```
00099
00100 /**
00101   * @brief  Set ACCELEROMETER Initialization.

00102   * @retval None
00103   */
00104 uint8_t BSP_ACCELERO_Init(void)
00105 {
00106   uint8_t ret = ACCELERO_ERROR;
00107   uint16_t ctrl = 0x0000;
00108   LIS302DL_InitTypeDef lis302dl_initstruct;
00109   LIS302DL_FilterConfigTypeDef lis302dl_filt
er={0,0,0};
00110
00111   if(Lis302dlDrv.ReadID() == I_AM_LIS302DL)
00112   {
00113     /* Initialize the gyroscope driver struc
ture */
00114     AcceleroDrv = &Lis302dlDrv;
00115
00116     /* Set configuration of LIS302DL MEMS Ac
celerometer **********************/
00117     lis302dl_initstruct.Power_Mode = LIS302D
L_LOWPOWERMODE_ACTIVE;
00118     lis302dl_initstruct.Output_DataRate = LI
S302DL_DATARATE_100;
00119     lis302dl_initstruct.Axes_Enable = LIS302
DL_XYZ_ENABLE;
00120     lis302dl_initstruct.Full_Scale = LIS302D
L_FULLSCALE_2_3;
00121     lis302dl_initstruct.Self_Test = LIS302DL
_SELFTEST_NORMAL;
00122
00123     /* Configure MEMS: data rate, power mode
, full scale, self test and axes */
00124     ctrl = (uint16_t) (lis302dl_initstruct.O
utput_DataRate | lis302dl_initstruct.Power_Mode |
```

```
\
00125                           lis302dl_initstruct.Fu
ll_Scale | lis302dl_initstruct.Self_Test | \
00126                           lis302dl_initstruct.Ax
es_Enable);
00127
00128     /* Configure the accelerometer main para
meters */
00129     AcceleroDrv->Init(ctrl);
00130
00131     /* MEMS High Pass Filter configuration */

00132     lis302dl_filter.HighPassFilter_Data_Sele
ction = LIS302DL_FILTEREDDATASELECTION_OUTPUTREGIS
TER;
00133     lis302dl_filter.HighPassFilter_CutOff_Fr
equency = LIS302DL_HIGHPASSFILTER_LEVEL_1;
00134     lis302dl_filter.HighPassFilter_Interrupt
 = LIS302DL_HIGHPASSFILTERINTERRUPT_1_2;
00135
00136     /* Configure MEMS high pass filter cut-o
ff level, interrupt and data selection bits */

00137     ctrl = (uint8_t)(lis302dl_filter.HighPas
sFilter_Data_Selection | \
00138                           lis302dl_filter.HighPa
ssFilter_CutOff_Frequency | \
00139                           lis302dl_filter.HighPa
ssFilter_Interrupt);
00140
00141     /* Configure the accelerometer LPF main
parameters */
00142     AcceleroDrv->FilterConfig(ctrl);
00143
00144     ret = ACCELERO_OK;
00145   }
00146   else
```

```
00147   {
00148     ret = ACCELERO_ERROR;
00149   }
00150
00151   return ret;
00152 }
00153
00154 /**
00155   * @brief  Read ID of Accelerometer compone
nt
00156   * @retval ID
00157   */
00158 uint8_t BSP_ACCELERO_ReadID(void)
00159 {
00160   uint8_t id = 0x00;
00161
00162   if(AcceleroDrv->ReadID != NULL)
00163   {
00164     id = AcceleroDrv->ReadID();
00165   }
00166   return id;
00167 }
00168
00169 /**
00170   * @brief  Reboot memory content of ACCELER
OMETER
00171   * @retval None
00172   */
00173 void BSP_ACCELERO_Reset(void)
00174 {
00175   if(AcceleroDrv->Reset != NULL)
00176 {
00177     AcceleroDrv->Reset();
00178   }
00179 }
00180
00181 /**
```

```
00182   * @brief  Config Accelerometer click IT
00183   * @retval None
00184   */
00185 void BSP_ACCELERO_Click_ITConfig(void)
00186 {
00187   if(AcceleroDrv->ConfigIT!= NULL)
00188   {
00189     AcceleroDrv->ConfigIT();
00190   }
00191 }
00192
00193
00194 /**
00195   * @brief  Clear Accelerometer click IT
00196   * @retval None
00197   */
00198 void BSP_ACCELERO_Click_ITClear(void)
00199 {
00200   if(AcceleroDrv->ClearIT!= NULL)
00201   {
00202     AcceleroDrv->ClearIT();
00203   }
00204 }
00205
00206 /**
00207   * @brief  Get XYZ acceleration
00208   * @param pDataXYZ: angular acceleration on
 X/Y/Z axis
00209   * @retval None
00210   */
00211 void BSP_ACCELERO_GetXYZ(int16_t *pDataXYZ)
00212 {
00213   int16_t SwitchXY = 0;
00214
00215   if(AcceleroDrv->GetXYZ!= NULL)
00216   {
00217     AcceleroDrv->GetXYZ(pDataXYZ);
```

```
00218
00219     /* Switch X and Y Axis in case of MEMS L
IS302DL */
00220     if(AcceleroDrv == &Lis302dlDrv)
00221     {
00222       SwitchXY  = pDataXYZ[0];
00223       pDataXYZ[0] = pDataXYZ[1];
00224       /* Invert Y Axis to be conpliant with
LIS3DSH */
00225       pDataXYZ[1] = -SwitchXY;
00226     }
00227   }
00228 }
00229
00230
00231 /**
00232   * @}
00233   */
00234
00235 /**
00236   * @}
00237   */
00238
00239 /**
00240   * @}
00241   */
00242
00243 /**
00244   * @}
00245   */
00246
00247
00248 /******************* (C) COPYRIGHT 2011 STMi
croelectronics *****END OF FILE****/
```

User Manual by  1.7.6.1

STM3210C_EVAL BSP User Manual

stm3210c_eval_audio.h

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210c_eval_audio.h
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file contains the common defines and functions prototypes for
00008   *          stm3210c_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 __STM3210C_EVAL_AUDIO_H
00041 #define __STM3210C_EVAL_AUDIO_H
00042
00043 #ifdef __cplusplus
00044  extern "C" {
00045 #endif
00046
00047 /* Includes ------------------------------------------------------------------*/
00048 /* Include audio component Driver */
00049 #include "../Components/cs43l22/cs43l22.h"
00050 #include "stm3210c_eval.h"
00051
00052 /** @addtogroup BSP
00053   * @{
00054   */
00055
00056 /** @addtogroup STM3210C_EVAL
00057   * @{
00058   */
00059
00060 /** @addtogroup STM3210C_EVAL_AUDIO
00061   * @{
00062   */
00063
00064
00065 /** @defgroup STM3210C_EVAL_AUDIO_Exported_Types AUDIO_Exported_Types
00066   * @{
00067   */
00068
```

```
00069 /**
00070   * @}
00071   */
00072
00073 /** @defgroup STM3210C_EVAL_AUDIO_OUT_Export
ed_Constants AUDIO_OUT_Exported_Constants
00074   * @{
00075   */
00076
00077
00078 /*-----------------------------------------
-----------------------------------
00079              AUDIO OUT CONFIGURATION
00080 -----------------------------------------
-----------------------------------*/
00081
00082 /* I2S peripheral configuration defines */
00083 #define I2SOUT                            SPI2
00084 #define I2SOUT_CLK_ENABLE()               __HA
L_RCC_SPI2_CLK_ENABLE()
00085 #define I2SOUT_SCK_SD_CLK_ENABLE()        __HA
L_RCC_GPIOB_CLK_ENABLE()
00086 #define I2SOUT_MCK_CLK_ENABLE()           __HA
L_RCC_GPIOC_CLK_ENABLE()
00087 #define I2SOUT_WS_CLK_ENABLE()            __HA
L_RCC_GPIOB_CLK_ENABLE()
00088 #define I2SOUT_WS_PIN                     GPIO
_PIN_12             /* PB.12*/
00089 #define I2SOUT_SCK_PIN                    GPIO
_PIN_13             /* PB.13*/
00090 #define I2SOUT_SD_PIN                     GPIO
_PIN_15             /* PB.15*/
00091 #define I2SOUT_MCK_PIN                    GPIO
_PIN_6              /* PC.06*/
00092 #define I2SOUT_SCK_SD_GPIO_PORT           GPIOB

00093 #define I2SOUT_WS_GPIO_PORT               GPIOB
```

```
00094 #define I2SOUT_MCK_GPIO_PORT             GPIOC

00095
00096 /* I2S DMA Channel definitions */
00097 #define I2SOUT_DMAx_CLK_ENABLE()         __HA
L_RCC_DMA1_CLK_ENABLE()
00098 #define I2SOUT_DMAx_CHANNEL              DMA1
_Channel5
00099 #define I2SOUT_DMAx_IRQ                  DMA1
_Channel5_IRQn
00100 #define I2SOUT_DMAx_PERIPH_DATA_SIZE     DMA_
PDATAALIGN_HALFWORD
00101 #define I2SOUT_DMAx_MEM_DATA_SIZE        DMA_
MDATAALIGN_HALFWORD
00102 #define DMA_MAX_SZE                      0xFF
FF
00103
00104 #define I2SOUT_IRQHandler                DMA1
_Channel5_IRQHandler
00105
00106 /* Select the interrupt preemption priority
and subpriority for the DMA interrupt */
00107 #define AUDIO_OUT_IRQ_PREPRIO          5   /*
 Select the preemption priority level(0 is the hig
hest) */
00108
00109 /*-----------------------------------------
-----------------------------------
00110              CONFIGURATION: Audio Driver Con
figuration parameters
00111 -----------------------------------------
-----------------------------------*/
00112
00113 /* Audio status definition */
00114 #define AUDIO_OK
  0
```

```
00115 #define AUDIO_ERROR
  1
00116 #define AUDIO_TIMEOUT
  2
00117
00118 /* PDM buffer input size */
00119 #define INTERNAL_BUFF_SIZE
  128*DEFAULT_AUDIO_IN_FREQ/16000*DEFAULT_AUDIO_IN
_CHANNEL_NBR
00120
00121 /*------------------------------------------
-----------------------------------
00122                         OPTIONAL Configuration d
efines parameters
00123 ------------------------------------------
-----------------------------------*/
00124
00125
00126 /**
00127   * @}
00128   */
00129
00130 /** @defgroup STM3210C_EVAL_AUDIO_Exported_M
acros AUDIO_Exported_Macros
00131   * @{
00132   */
00133 #define DMA_MAX(_X_)                (((_X_)
<= DMA_MAX_SZE)? (_X_):DMA_MAX_SZE)
00134
00135 /**
00136   * @}
00137   */
00138
00139 /** @addtogroup STM3210C_EVAL_AUDIO_OUT_Expo
rted_Functions
00140   * @{
00141   */
```

```
00142 uint8_t BSP_AUDIO_OUT_Init(uint16_t OutputDe
vice, uint8_t Volume, uint32_t AudioFreq);
00143 uint8_t BSP_AUDIO_OUT_Play(uint16_t* pBuffer
, uint32_t Size);
00144 void    BSP_AUDIO_OUT_ChangeBuffer(uint16_t
*pData, uint16_t Size);
00145 uint8_t BSP_AUDIO_OUT_Pause(void);
00146 uint8_t BSP_AUDIO_OUT_Resume(void);
00147 uint8_t BSP_AUDIO_OUT_Stop(uint32_t Option);
00148 uint8_t BSP_AUDIO_OUT_SetVolume(uint8_t Volu
me);
00149 void    BSP_AUDIO_OUT_SetFrequency(uint32_t
AudioFreq);
00150 uint8_t BSP_AUDIO_OUT_SetMute(uint32_t Comma
nd);
00151 uint8_t BSP_AUDIO_OUT_SetOutputMode(uint8_t
Output);
00152
00153 /* User Callbacks: user has to implement the
se functions in his code if they are needed. */
00154 /* This function is called when the requeste
d data has been completely transferred.*/
00155 void    BSP_AUDIO_OUT_TransferComplete_CallB
ack(void);
00156
00157 /* This function is called when half of the
requested buffer has been transferred. */
00158 void    BSP_AUDIO_OUT_HalfTransfer_CallBack(
void);
00159
00160 /* This function is called when an Interrupt
 due to transfer error on or peripheral
00161    error occurs. */
00162 void    BSP_AUDIO_OUT_Error_CallBack(void);
00163
00164 /**
00165   * @}
```

```
00166   */
00167
00168
00169
00170 /**
00171   * @}
00172   */
00173
00174 /**
00175   * @}
00176   */
00177
00178 /**
00179   * @}
00180   */
00181
00182 #ifdef __cplusplus
00183 }
00184 #endif
00185
00186 #endif /* __STM3210C_EVAL_AUDIO_H */
00187
00188 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210C_EVAL BSP User Manual

stm3210c_eval_audio.c

Go to the documentation of this file.

```
00001 /**
00002   ******************************************************************************
00003   * @file    stm3210c_eval_audio.c
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file provides the Audio driver for the STM3210C-Eval
00008   *          board.
00009   @verbatim
00010   ==============================================================================
00011                        ##### How to use this driver #####
00012   ==============================================================================
00013     [..]
00014   (#) This driver supports STM32F107xC devices on STM3210C-Eval Kit:
00015         (++) to play an audio file (all functions names start by BSP_AUDIO_OUT_xxx)
```

```
00016
00017   [..]
00018    (#) PLAY A FILE:
00019        (++) Call the function BSP_AUDIO_OUT_
Init(
00020               OutputDevice: physical output
mode (OUTPUT_DEVICE_SPEAKER,
00021                                OUTPUT_DEVICE_HEA
DPHONE, OUTPUT_DEVICE_AUTO or
00022                                OUTPUT_DEVICE_BOT
H)
00023               Volume: initial volume to be s
et (0 is min (mute), 100 is max (100%)
00024               AudioFreq: Audio frequency in
Hz (8000, 16000, 22500, 32000 ...)
00025               this parameter is relative to
the audio file/stream type.
00026                )
00027            This function configures all the
hardware required for the audio application
00028            (codec, I2C, I2S, GPIOs, DMA and
interrupt if needed). This function returns 0
00029            if configuration is OK.
00030            If the returned value is differen
t from 0 or the function is stuck then the
00031            communication with the codec (try
 to un-plug the power or reset device in this case
).
00032               (+++) OUTPUT_DEVICE_SPEAKER: o
nly speaker will be set as output for the
00033                     audio stream.
00034               (+++) OUTPUT_DEVICE_HEADPHONE:
 only headphones will be set as output for
00035                     the audio stream.
00036               (+++) OUTPUT_DEVICE_AUTO: Sele
ction of output device is made through external
00037                     switch (implemented into
```

```
 the audio jack on the evaluation board).
00038                     When the Headphone is co
nnected it is used as output.
00039                     When the headphone is di
sconnected from the audio jack, the output is
00040                     automatically switched t
o Speaker.
00041                (+++) OUTPUT_DEVICE_BOTH: both
 Speaker and Headphone are used as outputs
00042                     for the audio stream at
the same time.
00043        (++) Call the function BSP_AUDIO_OUT_
Play(
00044                  pBuffer: pointer to the audi
o data file address
00045                  Size: size of the buffer to
be sent in Bytes
00046                 )
00047                to start playing (for the firs
t time) from the audio file/stream.
00048        (++) Call the function BSP_AUDIO_OUT_
Pause() to pause playing
00049        (++) Call the function BSP_AUDIO_OUT_
Resume() to resume playing.
00050             Note. After calling BSP_AUDIO_OU
T_Pause() function for pause,
00051             only BSP_AUDIO_OUT_Resume() shou
ld be called for resume
00052             (it is not allowed to call BSP_A
UDIO_OUT_Play() in this case).
00053             Note. This function should be ca
lled only when the audio file is played
00054             or paused (not stopped).
00055        (++) For each mode, you may need to i
mplement the relative callback functions
00056             into your code.
00057             The Callback functions are named
```

```
 BSP_AUDIO_OUT_XXXCallBack() and only
00058             their prototypes are declared in
 the stm3210c_eval_audio.h file.
00059             (refer to the example for more d
etails on the callbacks implementations)
00060         (++) To Stop playing, to modify the v
olume level, the frequency or to mute,
00061             use the functions BSP_AUDIO_OUT_
Stop(), BSP_AUDIO_OUT_SetVolume(),
00062             AUDIO_OUT_SetFrequency() BSP_AUD
IO_OUT_SetOutputMode and BSP_AUDIO_OUT_SetMute().
00063         (++) The driver API and the callback
 functions are at the end of the
00064             stm3210c_eval_audio.h file.
00065
00066         (++) This driver provide the High Aud
io Layer: consists of the function API
00067             exported in the stm3210c_eval_au
dio.h file (BSP_AUDIO_OUT_Init(),
00068             BSP_AUDIO_OUT_Play() ...)
00069         (++) This driver provide also the Med
ia Access Layer (MAL): which consists
00070             of functions allowing to access
 the media containing/providing the
00071             audio file/stream. These functio
ns are also included as local functions into
00072             the stm3210c_eval_audio.c file (
I2SOUT_Init()...)
00073
00074   [..]
00075                       ##### Known Limitations
 #####
00076   ============================================
====================================
00077    (#) When using the Speaker, if the audio
 file quality is not high enough, the
00078         speaker output may produce high and u
```

```
ncomfortable noise level. To avoid
00079        this issue, to use speaker output pro
perly, try to increase audio file
00080        sampling rate (typically higher than
48KHz).
00081        This operation will lead to larger fi
le size.
00082
00083    (#) Communication with the audio codec (t
hrough I2C) may be corrupted if it
00084        is interrupted by some user interrupt
 routines (in this case, interrupts
00085        could be disabled just before the sta
rt of communication then re-enabled
00086        when it is over). Note that this comm
unication is only done at the
00087        configuration phase (BSP_AUDIO_OUT_In
it() or BSP_AUDIO_OUT_Stop())
00088        and when Volume control modification
is performed (BSP_AUDIO_OUT_SetVolume()
00089        or BSP_AUDIO_OUT_SetMute()or BSP_AUDI
O_OUT_SetOutputMode()).
00090        When the audio data is played, no com
munication is required with the audio codec.
00091
00092    (#) Parsing of audio file is not implemen
ted (in order to determine audio file
00093        properties: Mono/Stereo, Data size, F
ile size, Audio Frequency, Audio Data
00094        header size ...). The configuration i
s fixed for the given audio file.
00095
00096    (#) Mono audio streaming is not supported
 (in order to play mono audio streams,
00097        each data should be sent twice on the
 I2S or should be duplicated on the
00098        source buffer. Or convert the stream
```

```
in stereo before playing).
00099
00100    (#) Supports only 16-bit audio data size.
00101
00102   @endverbatim
00103   ******************************************
*********************************
00104   * @attention
00105   *
00106   * <h2><center>&copy; COPYRIGHT(c) 2015 STM
icroelectronics</center></h2>
00107   *
00108   * Redistribution and use in source and bin
ary forms, with or without modification,
00109   * are permitted provided that the followin
g conditions are met:
00110   *   1. Redistributions of source code must
 retain the above copyright notice,
00111   *      this list of conditions and the fol
lowing disclaimer.
00112   *   2. Redistributions in binary form must
 reproduce the above copyright notice,
00113   *      this list of conditions and the fol
lowing disclaimer in the documentation
00114   *      and/or other materials provided wit
h the distribution.
00115   *   3. Neither the name of STMicroelectron
ics nor the names of its contributors
00116   *      may be used to endorse or promote p
roducts derived from this software
00117   *      without specific prior written perm
ission.
00118   *
00119   * THIS SOFTWARE IS PROVIDED BY THE COPYRIG
HT HOLDERS AND CONTRIBUTORS "AS IS"
00120   * AND ANY EXPRESS OR IMPLIED WARRANTIES, I
NCLUDING, BUT NOT LIMITED TO, THE
```

```
00121   * IMPLIED WARRANTIES OF MERCHANTABILITY AN
D FITNESS FOR A PARTICULAR PURPOSE ARE
00122   * DISCLAIMED. IN NO EVENT SHALL THE COPYRI
GHT HOLDER OR CONTRIBUTORS BE LIABLE
00123   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SP
ECIAL, EXEMPLARY, OR CONSEQUENTIAL
00124   * DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR
00125   * SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER
00126   * CAUSED AND ON ANY THEORY OF LIABILITY, W
HETHER IN CONTRACT, STRICT LIABILITY,
00127   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWI
SE) ARISING IN ANY WAY OUT OF THE USE
00128   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 POSSIBILITY OF SUCH DAMAGE.
00129   *
00130   ******************************************
**********************************
00131   */
00132
00133 /* Includes ------------------------------
-----------------------------------*/
00134 #include "stm3210c_eval_audio.h"
00135
00136 /** @addtogroup BSP
00137   * @{
00138   */
00139
00140 /** @addtogroup STM3210C_EVAL
00141   * @{
00142   */
00143
00144 /** @addtogroup STM3210C_EVAL_AUDIO
00145   * @brief This file includes the low layer
audio driver available on STM3210C-Eval
00146   * eval board.
```

```
00147   * @{
00148   */
00149
00150 /** @defgroup STM3210C_EVAL_AUDIO_Private_Ty
pes AUDIO_Private_Types
00151   * @{
00152   */
00153 /**
00154   * @}
00155   */
00156
00157 /** @defgroup STM3210C_EVAL_AUDIO_Private_De
fines AUDIO_Private_Defines
00158   * @{
00159   */
00160
00161 /**
00162   * @}
00163   */
00164
00165 /** @defgroup STM3210C_EVAL_AUDIO_Private_Ma
cros AUDIO Private_Macros
00166   * @{
00167   */
00168 /**
00169   * @}
00170   */
00171
00172 /** @defgroup STM3210C_EVAL_AUDIO_Private_Va
riables AUDIO_Private_Variables
00173   * @{
00174   */
00175 /*### PLAY ###*/
00176 static AUDIO_DrvTypeDef          *pAudioDrv;
00177 I2S_HandleTypeDef                hAudioOutI2s
;
00178
```

```
00179 /**
00180   * @}
00181   */
00182
00183 /** @defgroup STM3210C_EVAL_AUDIO_Private_Fu
nction_Prototypes AUDIO_Private_Function_Prototypes

00184   * @{
00185   */
00186 static void  I2SOUT_MspInit(void);
00187 static void  I2SOUT_Init(uint32_t AudioFreq)
;
00188
00189 /**
00190   * @}
00191   */
00192
00193 /** @defgroup STM3210C_EVAL_AUDIO_OUT_Export
ed_Functions AUDIO_OUT_Exported_Functions
00194   * @{
00195   */
00196
00197 /**
00198   * @brief  Configure the audio peripherals.
00199   * @param  OutputDevice: OUTPUT_DEVICE_SPEA
KER, OUTPUT_DEVICE_HEADPHONE,
00200   *                       OUTPUT_DEVICE_BOTH
 or OUTPUT_DEVICE_AUTO .
00201   * @param  Volume: Initial volume level (fr
om 0 (Mute) to 100 (Max))
00202   * @param  AudioFreq: Audio frequency used
to play the audio stream.
00203   * @retval 0 if correct communication, else
 wrong communication
00204   */
00205 uint8_t BSP_AUDIO_OUT_Init(uint16_t OutputDe
vice, uint8_t Volume, uint32_t AudioFreq)
```

```
00206 {
00207   uint8_t ret = AUDIO_ERROR;
00208   uint32_t deviceid = 0x00;
00209
00210   deviceid = cs43l22_drv.ReadID(AUDIO_I2C_AD
DRESS);
00211
00212   if((deviceid & CS43L22_ID_MASK) == CS43L22
_ID)
00213   {
00214     /* Initialize the audio driver structure
 */
00215     pAudioDrv = &cs43l22_drv;
00216     ret = AUDIO_OK;
00217   }
00218   else
00219   {
00220     ret = AUDIO_ERROR;
00221   }
00222
00223   if(ret == AUDIO_OK)
00224   {
00225     pAudioDrv->Init(AUDIO_I2C_ADDRESS, Outpu
tDevice, Volume, AudioFreq);
00226     /* I2S data transfer preparation:
00227     Prepare the Media to be used for the aud
io transfer from memory to I2S peripheral */
00228     /* Configure the I2S peripheral */
00229     I2SOUT_Init(AudioFreq);
00230   }
00231
00232   return ret;
00233 }
00234
00235 /**
00236   * @brief  Starts playing audio stream from
 a data buffer for a determined size.
```

```
00237   * @param  pBuffer: Pointer to the buffer
00238   * @param  Size: Number of audio data BYTES.

00239   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00240   */
00241 uint8_t BSP_AUDIO_OUT_Play(uint16_t* pBuffer
, uint32_t Size)
00242 {
00243   /* Call the audio Codec Play function */
00244   if(pAudioDrv->Play(AUDIO_I2C_ADDRESS, pBuf
fer, Size) != 0)
00245   {
00246     return AUDIO_ERROR;
00247   }
00248   else
00249   {
00250     /* Update the Media layer and enable it
for play */
00251     HAL_I2S_Transmit_DMA(&hAudioOutI2s, pBuf
fer, DMA_MAX(Size));
00252
00253     return AUDIO_OK;
00254   }
00255 }
00256
00257 /**
00258   * @brief Sends n-Bytes on the I2S interfac
e.
00259   * @param pData: pointer on data address
00260   * @param Size: number of data to be written

00261   * @retval None
00262   */
00263 void BSP_AUDIO_OUT_ChangeBuffer(uint16_t *pD
ata, uint16_t Size)
00264 {
```

```
00265   HAL_I2S_Transmit_DMA(&hAudioOutI2s, pData,
 Size);
00266 }
00267
00268 /**
00269   * @brief  This function Pauses the audio f
ile stream. In case
00270   *         of using DMA, the DMA Pause feat
ure is used.
00271   * @note When calling BSP_AUDIO_OUT_Pause()
 function for pause, only
00272   *          BSP_AUDIO_OUT_Resume() function
 should be called for resume (use of BSP_AUDIO_OUT
_Play()
00273   *          function for resume could lead
to unexpected behavior).
00274   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00275   */
00276 uint8_t BSP_AUDIO_OUT_Pause(void)
00277 {
00278   /* Call the Audio Codec Pause/Resume funct
ion */
00279   if(pAudioDrv->Pause(AUDIO_I2C_ADDRESS) !=
0)
00280   {
00281     return AUDIO_ERROR;
00282   }
00283   else
00284   {
00285     /* Call the Media layer pause function */

00286     HAL_I2S_DMAPause(&hAudioOutI2s);
00287
00288     /* Return AUDIO_OK if all operations are
 OK */
00289     return AUDIO_OK;
```

```
00290   }
00291 }
00292
00293 /**
00294   * @brief  This function  Resumes the audio
 file stream.
00295   * @note When calling BSP_AUDIO_OUT_Pause()
 function for pause, only
00296   *          BSP_AUDIO_OUT_Resume() function
 should be called for resume (use of BSP_AUDIO_OUT
_Play()
00297   *          function for resume could lead
to unexpected behavior).
00298   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00299   */
00300 uint8_t BSP_AUDIO_OUT_Resume(void)
00301 {
00302   /* Call the Audio Codec Pause/Resume funct
ion */
00303   if(pAudioDrv->Resume(AUDIO_I2C_ADDRESS) !=
 0)
00304   {
00305     return AUDIO_ERROR;
00306   }
00307   else
00308   {
00309     /* Call the Media layer resume function
*/
00310     HAL_I2S_DMAResume(&hAudioOutI2s);
00311     /* Return AUDIO_OK if all operations are
 OK */
00312     return AUDIO_OK;
00313   }
00314 }
00315
00316 /**
```

```
00317   * @brief  Stops audio playing and Power do
wn the Audio Codec.
00318   * @param  Option: could be one of the foll
owing parameters
00319   *           - CODEC_PDWN_HW: completely sh
ut down the codec (physically).
00320   *                                Then need to
reconfigure the Codec after power on.
00321   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00322   */
00323 uint8_t BSP_AUDIO_OUT_Stop(uint32_t Option)
00324 {
00325   /* Call DMA Stop to disable DMA stream bef
ore stopping codec */
00326   HAL_I2S_DMAStop(&hAudioOutI2s);
00327
00328   /* Call Audio Codec Stop function */
00329   if(pAudioDrv->Stop(AUDIO_I2C_ADDRESS, Opti
on) != 0)
00330   {
00331     return AUDIO_ERROR;
00332   }
00333   else
00334   {
00335     if(Option == CODEC_PDWN_HW)
00336     {
00337       /* Wait at least 100us */
00338       HAL_Delay(1);
00339
00340       /* Power Down the codec */
00341       BSP_IO_WritePin(AUDIO_RESET_PIN, GPIO_
PIN_RESET);
00342
00343     }
00344     /* Return AUDIO_OK when all operations a
re correctly done */
```

```
00345     return AUDIO_OK;
00346   }
00347 }
00348
00349 /**
00350   * @brief  Controls the current audio volum
e level.
00351   * @param  Volume: Volume level to be set i
n percentage from 0% to 100% (0 for
00352   *         Mute and 100 for Max volume leve
l).
00353   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00354   */
00355 uint8_t BSP_AUDIO_OUT_SetVolume(uint8_t Volu
me)
00356 {
00357   /* Call the codec volume control function
with converted volume value */
00358   if(pAudioDrv->SetVolume(AUDIO_I2C_ADDRESS,
 Volume) != 0)
00359   {
00360     return AUDIO_ERROR;
00361   }
00362   else
00363   {
00364     /* Return AUDIO_OK when all operations a
re correctly done */
00365     return AUDIO_OK;
00366   }
00367 }
00368
00369 /**
00370   * @brief  Enables or disables the MUTE mod
e by software
00371   * @param  Cmd: could be AUDIO_MUTE_ON to m
ute sound or AUDIO_MUTE_OFF to
```

```
00372   *         unmute the codec and restore pre
vious volume level.
00373   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00374   */
00375 uint8_t BSP_AUDIO_OUT_SetMute(uint32_t Cmd)
00376 {
00377   /* Call the Codec Mute function */
00378   if(pAudioDrv->SetMute(AUDIO_I2C_ADDRESS, C
md) != 0)
00379   {
00380     return AUDIO_ERROR;
00381   }
00382   else
00383   {
00384     /* Return AUDIO_OK when all operations a
re correctly done */
00385     return AUDIO_OK;
00386   }
00387 }
00388
00389 /**
00390   * @brief  Switch dynamically (while audio
file is played) the output target
00391   *         (speaker or headphone).
00392   * @note   This function modifies a global
variable of the audio codec driver: OutputDev.
00393   * @param  Output: specifies the audio outp
ut target: OUTPUT_DEVICE_SPEAKER,
00394   *         OUTPUT_DEVICE_HEADPHONE, OUTPUT_
DEVICE_BOTH or OUTPUT_DEVICE_AUTO
00395   * @retval AUDIO_OK if correct communicatio
n, else wrong communication
00396   */
00397 uint8_t BSP_AUDIO_OUT_SetOutputMode(uint8_t
Output)
00398 {
```

```
00399   /* Call the Codec output Device function */

00400   if(pAudioDrv->SetOutputMode(AUDIO_I2C_ADDR
ESS, Output) != 0)
00401   {
00402     return AUDIO_ERROR;
00403   }
00404   else
00405   {
00406     /* Return AUDIO_OK when all operations a
re correctly done */
00407     return AUDIO_OK;
00408   }
00409 }
00410
00411 /**
00412   * @brief  Update the audio frequency.
00413   * @param  AudioFreq: Audio frequency used
to play the audio stream.
00414   * @retval None
00415   * @note This API should be called after th
e BSP_AUDIO_OUT_Init() to adjust the
00416   * audio frequency.
00417   */
00418 void BSP_AUDIO_OUT_SetFrequency(uint32_t Aud
ioFreq)
00419 {
00420   /* Update the I2S audio frequency configur
ation */
00421   I2SOUT_Init(AudioFreq);
00422 }
00423
00424 /**
00425   * @brief Tx Transfer completed callbacks
00426   * @param hi2s: I2S handle
00427   * @retval None
00428   */
```

```
00429 void HAL_I2S_TxCpltCallback(I2S_HandleTypeDe
f *hi2s)
00430 {
00431   if(hi2s->Instance == I2SOUT)
00432   {
00433     /* Call the user function which will man
age directly transfer complete*/
00434     BSP_AUDIO_OUT_TransferComplete_CallBack(
);
00435   }
00436 }
00437
00438 /**
00439   * @brief Tx Transfer Half completed callba
cks
00440   * @param hi2s: I2S handle
00441   * @retval None
00442   */
00443 void HAL_I2S_TxHalfCpltCallback(I2S_HandleTy
peDef *hi2s)
00444 {
00445   if(hi2s->Instance == I2SOUT)
00446   {
00447     /* Manage the remaining file size and ne
w address offset: This function
00448     should be coded by user (its prototype i
s already declared in stm3210c_eval_audio.h) */
00449     BSP_AUDIO_OUT_HalfTransfer_CallBack();
00450   }
00451 }
00452
00453 /**
00454   * @brief I2S error callbacks
00455   * @param hi2s: I2S handle
00456   * @retval None
00457   */
00458 void HAL_I2S_ErrorCallback(I2S_HandleTypeDef
```

```
 *hi2s)
00459 {
00460   /* Manage the error generated on DMA FIFO:
 This function
00461      should be coded by user (its prototype
is already declared in stm3210c_eval_audio.h) */
00462   if(hi2s->Instance == I2SOUT)
00463   {
00464     BSP_AUDIO_OUT_Error_CallBack();
00465   }
00466 }
00467
00468 /**
00469   * @brief  Manages the DMA full Transfer co
mplete event.
00470   * @retval None
00471   */
00472 __weak void BSP_AUDIO_OUT_TransferComplete_C
allBack(void)
00473 {
00474 }
00475
00476 /**
00477   * @brief  Manages the DMA Half Transfer co
mplete event.
00478   * @retval None
00479   */
00480 __weak void BSP_AUDIO_OUT_HalfTransfer_CallB
ack(void)
00481 {
00482 }
00483
00484 /**
00485   * @brief  Manages the DMA FIFO error event.

00486   * @retval None
00487   */
```

```
00488 __weak void BSP_AUDIO_OUT_Error_CallBack(void
)
00489 {
00490 }
00491
00492 /**********************************************
**********************************
00493                                    Static Function
00494 **********************************************
**********************************/
00495
00496 /**
00497   * @brief AUDIO OUT I2S MSP Init
00498   * @retval None
00499   */
00500 static void I2SOUT_MspInit(void)
00501 {
00502   static DMA_HandleTypeDef  hdma_i2stx;
00503   GPIO_InitTypeDef          gpioinitstruct =
 {0};
00504   I2S_HandleTypeDef         *hi2s = &hAudioO
utI2s;
00505
00506   /* Enable I2SOUT clock */
00507   I2SOUT_CLK_ENABLE();
00508
00509   /*** Configure the GPIOs ***/
00510   /* Enable I2S GPIO clocks */
00511   I2SOUT_SCK_SD_CLK_ENABLE();
00512   I2SOUT_WS_CLK_ENABLE();
00513
00514   /* I2SOUT pins configuration: WS, SCK and
SD pins -----------------------------*/
00515   gpioinitstruct.Pin         = I2SOUT_SCK_PIN
 | I2SOUT_SD_PIN;
00516   gpioinitstruct.Mode        = GPIO_MODE_AF_
PP;
```

```
00517   gpioinitstruct.Pull         = GPIO_NOPULL;
00518   gpioinitstruct.Speed        = GPIO_SPEED_FR
EQ_MEDIUM;
00519   HAL_GPIO_Init(I2SOUT_SCK_SD_GPIO_PORT, &gp
ioinitstruct);
00520
00521   gpioinitstruct.Pin          = I2SOUT_WS_PIN
 ;
00522   HAL_GPIO_Init(I2SOUT_WS_GPIO_PORT, &gpioin
itstruct);
00523
00524   /* I2SOUT pins configuration: MCK pin */
00525   I2SOUT_MCK_CLK_ENABLE();
00526   gpioinitstruct.Pin          = I2SOUT_MCK_PIN
;
00527   HAL_GPIO_Init(I2SOUT_MCK_GPIO_PORT, &gpioi
nitstruct);
00528
00529   /* Enable the I2S DMA clock */
00530   I2SOUT_DMAx_CLK_ENABLE();
00531
00532   if(hi2s->Instance == I2SOUT)
00533   {
00534     /* Configure the hdma_i2stx handle param
eters */
00535     hdma_i2stx.Init.Direction           = DM
A_MEMORY_TO_PERIPH;
00536     hdma_i2stx.Init.PeriphInc           = DM
A_PINC_DISABLE;
00537     hdma_i2stx.Init.MemInc              = DM
A_MINC_ENABLE;
00538     hdma_i2stx.Init.PeriphDataAlignment = I2
SOUT_DMAx_PERIPH_DATA_SIZE;
00539     hdma_i2stx.Init.MemDataAlignment    = I2
SOUT_DMAx_MEM_DATA_SIZE;
00540     hdma_i2stx.Init.Mode                = DM
A_NORMAL;
```

```
00541     hdma_i2stx.Init.Priority              = DM
A_PRIORITY_HIGH;
00542
00543     hdma_i2stx.Instance                   = I2
SOUT_DMAx_CHANNEL;
00544
00545     /* Associate the DMA handle */
00546     __HAL_LINKDMA(hi2s, hdmatx, hdma_i2stx);
00547
00548     /* Deinitialize the Channel for new tran
sfer */
00549     HAL_DMA_DeInit(&hdma_i2stx);
00550
00551     /* Configure the DMA Channel */
00552     HAL_DMA_Init(&hdma_i2stx);
00553   }
00554
00555   /* I2S DMA IRQ Channel configuration */
00556   HAL_NVIC_SetPriority(I2SOUT_DMAx_IRQ, AUDI
O_OUT_IRQ_PREPRIO, 0);
00557   HAL_NVIC_EnableIRQ(I2SOUT_DMAx_IRQ);
00558 }
00559
00560 /**
00561   * @brief  Initializes the Audio Codec audi
o interface (I2S)
00562   * @param  AudioFreq: Audio frequency to be
 configured for the I2S peripheral.
00563   */
00564 static void I2SOUT_Init(uint32_t AudioFreq)
00565 {
00566   /* Initialize the hAudioOutI2s Instance pa
rameter */
00567   hAudioOutI2s.Instance         = I2SOUT;
00568
00569  /* Disable I2S block */
00570   __HAL_I2S_DISABLE(&hAudioOutI2s);
```

```
00571
00572   /* Perform MSP initialization at first fun
ction call */
00573   if(HAL_I2S_GetState(&hAudioOutI2s) == HAL_
I2S_STATE_RESET)
00574   {
00575     I2SOUT_MspInit();
00576   }
00577
00578   /* I2SOUT peripheral configuration */
00579   hAudioOutI2s.Init.AudioFreq   = AudioFreq;
00580   hAudioOutI2s.Init.CPOL        = I2S_CPOL_L
OW;
00581   hAudioOutI2s.Init.DataFormat  = I2S_DATAFO
RMAT_16B;
00582   hAudioOutI2s.Init.MCLKOutput  = I2S_MCLKOU
TPUT_ENABLE;
00583   hAudioOutI2s.Init.Mode        = I2S_MODE_M
ASTER_TX;
00584   hAudioOutI2s.Init.Standard    = I2S_STANDA
RD;
00585
00586   /* Initialize the I2S peripheral with the
structure above */
00587   HAL_I2S_Init(&hAudioOutI2s);
00588 }
00589
00590 /**
00591   * @}
00592   */
00593
00594 /**
00595   * @}
00596   */
00597
00598 /**
00599   * @}
```

```
00600   */
00601
00602 /**
00603   * @}
00604   */
00605
00606 /**
00607   * @}
00608   */
00609
00610 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210C_EVAL BSP User Manual

stm3210c_eval_io.c

Go to the documentation of this file.

```
00001 /**
00002   ********************************************************************************
00003   * @file    stm3210c_eval_io.c
00004   * @author  MCD Application Team
00005   * @version V6.0.1
00006   * @date    18-December-2015
00007   * @brief   This file provides a set of functions needed to manage the IO pins
00008   *          on STM3210C-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 module of the STM3210C-EVAL evaluation
00044      board.
00045    - The STMPE811 IO expander device component driver must be included with this
00046      driver in order to run the IO functionalities 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 BSP_IO_Init() function. This
00053        function includes the MSP layer hardware 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 calling the function BSP_IO_ConfigPin(), you
00058        must specify the desired IO mode by choosing 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 "stm3210c_eval_io.h"
00074
00075 /** @addtogroup BSP
00076   * @{
00077   */
00078
00079 /** @addtogroup STM3210C_EVAL
00080   * @{
00081   */
00082
00083 /** @defgroup STM3210C_EVAL_IO STM3210C_EVAL
 IO Expander
00084   * @{
00085   */
00086
```

```
00087 /* Private typedef -----------------------
-----------------------------------*/
00088
00089 /** @defgroup STM3210C_EVAL_IO_Private_Types
_Definitions Private Types Definitions
00090   * @{
00091   */
00092
00093 /**
00094   * @}
00095   */
00096
00097 /* Private define ------------------------
-----------------------------------*/
00098
00099 /** @defgroup STM3210C_EVAL_IO_Private_Defin
es Private_Defines
00100   * @{
00101   */
00102
00103 /**
00104   * @}
00105   */
00106
00107 /* Private macro -------------------------
-----------------------------------*/
00108
00109 /** @defgroup STM3210C_EVAL_IO_Private_Macro
s Private_Macros
00110   * @{
00111   */
00112
00113 /**
00114   * @}
00115   */
00116
00117 /* Private variables ---------------------
```

```
-----------------------------------*/
00118
00119 /** @defgroup STM3210C_EVAL_IO_Private_Variables Private_Variables
00120   * @{
00121   */
00122 static IO_DrvTypeDef *io1_driver;
00123 static IO_DrvTypeDef *io2_driver;
00124
00125
00126 /**
00127   * @}
00128   */
00129
00130 /* Private function prototypes -----------------------------------*/
00131
00132 /** @defgroup STM3210C_EVAL_IO_Private_Function_Prototypes Private_Function_Prototypes
00133   * @{
00134   */
00135
00136 /**
00137   * @}
00138   */
00139
00140 /* Private functions ---------------------------------------------*/
00141
00142 /** @defgroup STM3210C_EVAL_IO_Exported_Functions Exported_Functions
00143   * @{
00144   */
00145
00146 /**
00147   * @brief  Initializes and configures the IO functionalities and configures all
```

```
00148   *         necessary hardware resources (GP
IOs, clocks..).
00149   * @note   BSP_IO_Init() is using HAL_Delay
() function to ensure that stmpe811
00150   *         IO Expander is correctly reset.
HAL_Delay() function provides accurate
00151   *         delay (in milliseconds) based on
 variable incremented in SysTick ISR.
00152   *         This implies that if BSP_IO_Init
() is called from a peripheral ISR process,
00153   *         then the SysTick interrupt must
have higher priority (numerically lower)
00154   *         than the peripheral interrupt. O
therwise the caller ISR process will be blocked.
00155   * @retval IO_OK: if all initializations ar
e OK. Other value if error.
00156   */
00157 uint8_t BSP_IO_Init(void)
00158 {
00159   uint8_t ret = IO_ERROR;
00160
00161   /* Initialize IO Expander 1*/
00162   if(stmpe811_io_drv.ReadID(IO1_I2C_ADDRESS)
 == STMPE811_ID)
00163   {
00164     /* Initialize the IO Expander 1 driver s
tructure */
00165     io1_driver = &stmpe811_io_drv;
00166
00167     io1_driver->Init(IO1_I2C_ADDRESS);
00168     io1_driver->Start(IO1_I2C_ADDRESS, IO1_P
IN_ALL >> IO1_PIN_OFFSET);
00169
00170     ret = IO_OK;
00171   }
00172
00173   /* Initialize IO Expander 2*/
```

```
00174   if(stmpe811_io_drv.ReadID(IO2_I2C_ADDRESS)
 == STMPE811_ID)
00175   {
00176     /* Initialize the IO Expander 2 driver s
tructure */
00177     io2_driver = &stmpe811_io_drv;
00178
00179     io2_driver->Init(IO2_I2C_ADDRESS);
00180     io2_driver->Start(IO2_I2C_ADDRESS, IO2_P
IN_ALL >> IO2_PIN_OFFSET);
00181
00182     ret = IO_OK;
00183   }
00184
00185   return ret;
00186 }
00187
00188 /**
00189   * @brief  Gets the selected pins IT status.

00190   * @param  IO_Pin: Selected pins to check t
he status.
00191   *          This parameter can be any combi
nation of the IO pins.
00192   * @retval Status of the checked IO pin(s).
00193   */
00194 uint32_t BSP_IO_ITGetStatus(uint32_t IO_Pin)
00195 {
00196   uint32_t status = 0;
00197   uint32_t io1_pin = 0;
00198   uint32_t io2_pin = 0;
00199
00200   io1_pin = (IO_Pin & IO1_PIN_ALL) >> IO1_PI
N_OFFSET;
00201   io2_pin = (IO_Pin & IO2_PIN_ALL) >> IO2_PI
N_OFFSET;
00202
```

```
00203   if (io1_pin != 0)
00204   {
00205     /* Return the IO Expander 1 Pin IT statu
s */
00206     status |= (io1_driver->ITStatus(IO1_I2C_
ADDRESS, io1_pin)) << IO1_PIN_OFFSET;
00207   }
00208
00209   if (io2_pin != 0)
00210   {
00211     /* Return the IO Expander 2 Pin IT statu
s */
00212     status |= (io2_driver->ITStatus(IO2_I2C_
ADDRESS, io2_pin)) << IO2_PIN_OFFSET;
00213   }
00214
00215   return status;
00216 }
00217
00218 /**
00219   * @brief  Clears the selected IO IT pendin
g bit.
00220   * @param  IO_Pin: Selected pins to check t
he status.
00221   *          This parameter can be any combi
nation of the IO pins.
00222   * @retval None
00223   */
00224 void BSP_IO_ITClear(uint32_t IO_Pin)
00225 {
00226   uint32_t io1_pin = 0;
00227   uint32_t io2_pin = 0;
00228
00229   io1_pin = (IO_Pin & IO1_PIN_ALL) >> IO1_PI
N_OFFSET;
00230   io2_pin = (IO_Pin & IO2_PIN_ALL) >> IO2_PI
N_OFFSET;
```

```
00231
00232   if (io1_pin != 0)
00233   {
00234     /* Clears the selected IO Expander 1 pin
(s) mode */
00235     io1_driver->ClearIT(IO1_I2C_ADDRESS, io1
_pin);
00236   }
00237
00238   if (io2_pin != 0)
00239   {
00240     /* Clears the selected IO Expander 2 pin
(s) mode */
00241     io2_driver->ClearIT(IO2_I2C_ADDRESS, io2
_pin);
00242   }
00243 }
00244
00245 /**
00246   * @brief  Configures the IO pin(s) accordi
ng to IO mode structure value.
00247   * @param  IO_Pin: Output pin to be set or
reset.
00248   *          This parameter can be any combi
nation of the IO pins.
00249   * @param  IO_Mode: IO pin mode to configure

00250   *          This parameter can be one of th
e following values:
00251   *            @arg  IO_MODE_INPUT
00252   *            @arg  IO_MODE_OUTPUT
00253   *            @arg  IO_MODE_IT_RISING_EDGE
00254   *            @arg  IO_MODE_IT_FALLING_EDGE
00255   *            @arg  IO_MODE_IT_LOW_LEVEL
00256   *            @arg  IO_MODE_IT_HIGH_LEVEL
00257   * @retval IO_OK: if all initializations ar
e OK. Other value if error.
```

```
00258   */
00259 uint8_t BSP_IO_ConfigPin(uint32_t IO_Pin, IO
_ModeTypedef IO_Mode)
00260 {
00261   uint32_t io1_pin = 0;
00262   uint32_t io2_pin = 0;
00263
00264   io1_pin = (IO_Pin & IO1_PIN_ALL) >> IO1_PI
N_OFFSET;
00265   io2_pin = (IO_Pin & IO2_PIN_ALL) >> IO2_PI
N_OFFSET;
00266
00267   if (io1_pin != 0)
00268   {
00269     /* Configure the selected IO Expander 1
pin(s) mode */
00270     io1_driver->Config(IO1_I2C_ADDRESS, io1_
pin, IO_Mode);
00271   }
00272
00273   if (io2_pin != 0)
00274   {
00275     /* Configure the selected IO Expander 2
pin(s) mode */
00276     io2_driver->Config(IO2_I2C_ADDRESS, io2_
pin, IO_Mode);
00277   }
00278
00279   return IO_OK;
00280 }
00281
00282 /**
00283   * @brief  Sets the selected pins state.
00284   * @param  IO_Pin: Selected pins to write.
00285   *          This parameter can be any combi
nation of the IO pins.
00286   * @param  PinState: New pins state to writ
```

```
e
00287   * @retval None
00288   */
00289 void BSP_IO_WritePin(uint32_t IO_Pin, uint8_
t PinState)
00290 {
00291   uint32_t io1_pin = 0;
00292   uint32_t io2_pin = 0;
00293
00294   io1_pin = (IO_Pin & IO1_PIN_ALL) >> IO1_PI
N_OFFSET;
00295   io2_pin = (IO_Pin & IO2_PIN_ALL) >> IO2_PI
N_OFFSET;
00296
00297   if (io1_pin != 0)
00298   {
00299     /* Sets the IO Expander 1 selected pins
state */
00300     io1_driver->WritePin(IO1_I2C_ADDRESS, io
1_pin, PinState);
00301   }
00302
00303   if (io2_pin != 0)
00304   {
00305     /* Sets the IO Expander 2 selected pins
state */
00306     io2_driver->WritePin(IO2_I2C_ADDRESS, io
2_pin, PinState);
00307   }
00308 }
00309
00310 /**
00311   * @brief  Gets the selected pins current s
tate.
00312   * @param  IO_Pin: Selected pins to read.
00313   *          This parameter can be any combi
nation of the IO pins.
```

```
00314   * @retval The current pins state
00315   */
00316 uint32_t BSP_IO_ReadPin(uint32_t IO_Pin)
00317 {
00318   uint32_t pin_state = 0;
00319   uint32_t io1_pin = 0;
00320   uint32_t io2_pin = 0;
00321
00322   io1_pin = (IO_Pin & IO1_PIN_ALL) >> IO1_PI
N_OFFSET;
00323   io2_pin = (IO_Pin & IO2_PIN_ALL) >> IO2_PI
N_OFFSET;
00324
00325   if (io1_pin != 0)
00326   {
00327     /* Gets the IO Expander 1 selected pins
current state */
00328     pin_state |= (io1_driver->ReadPin(IO1_I2
C_ADDRESS, io1_pin)) << IO1_PIN_OFFSET;
00329   }
00330
00331   if (io2_pin != 0)
00332   {
00333     /* Gets the IO Expander 2 selected pins
current state */
00334     pin_state |= (io2_driver->ReadPin(IO2_I2
C_ADDRESS, io2_pin)) << IO2_PIN_OFFSET;
00335   }
00336
00337   return pin_state;
00338 }
00339
00340 /**
00341   * @brief  Toggles the selected pins state
00342   * @param  IO_Pin: Selected pins to toggle.

00343   *          This parameter can be any combi
```

```
nation of the IO pins.
00344   * @retval None
00345   */
00346 void BSP_IO_TogglePin(uint32_t IO_Pin)
00347 {
00348   uint32_t io1_pin = 0;
00349   uint32_t io2_pin = 0;
00350
00351   io1_pin = (IO_Pin & IO1_PIN_ALL) >> IO1_PI
N_OFFSET;
00352   io2_pin = (IO_Pin & IO2_PIN_ALL) >> IO2_PI
N_OFFSET;
00353
00354   if (io1_pin != 0)
00355   {
00356     /* Toggles the IO Expander 1 selected pi
ns state */
00357     if(io1_driver->ReadPin(IO1_I2C_ADDRESS,
io1_pin) == RESET) /* Set */
00358     {
00359       BSP_IO_WritePin(io1_pin, GPIO_PIN_SET)
; /* Reset */
00360     }
00361     else
00362     {
00363       BSP_IO_WritePin(io1_pin, GPIO_PIN_RESE
T);  /* Set */
00364     }
00365   }
00366
00367   if (io2_pin != 0)
00368   {
00369     /* Toggles the IO Expander 2 selected pi
ns state */
00370     if(io2_driver->ReadPin(IO2_I2C_ADDRESS,
io2_pin) == RESET) /* Set */
00371     {
```

```
00372         BSP_IO_WritePin(io2_pin, GPIO_PIN_SET)
; /* Reset */
00373       }
00374       else
00375       {
00376         BSP_IO_WritePin(io2_pin, GPIO_PIN_RESE
T);  /* Set */
00377       }
00378     }
00379 }
00380
00381 /**
00382   * @}
00383   */
00384
00385 /**
00386   * @}
00387   */
00388
00389 /**
00390   * @}
00391   */
00392
00393 /**
00394   * @}
00395   */
00396
00397 /************************ (C) COPYRIGHT STMi
croelectronics *****END OF FILE****/
```

STM3210C_EVAL BSP User Manual

Modules

BSP

Modules

STM3210C-EVAL

Generated on Thu Dec 10 2015 17:13:52 for STM3210C_EVAL BSP
User Manual by doxygen 1.7.6.1

STM3210C_EVAL BSP User Manual

STM3210C-EVAL

BSP

Modules

- **STM3210C-EVAL Common**
- **STM3210C_EVAL IO Expander**
- **STM3210C_EVAL LCD**
- **STM3210C_EVAL SD**
- **STM3210C_EVAL Touch Screen**
- **STM3210C_EVAL_ACCELEROMETER**

  This file includes the motion sensor driver for ACCELEROMETER motion sensor devices.

- **STM3210C_EVAL_AUDIO**

  This file includes the low layer audio driver available on STM3210C-Eval eval board.

- **STM3210C_EVAL_EEPROM**

  This file includes the I2C and SPI EEPROM driver of STM3210C-EVAL board.

STM3210C_EVAL BSP User Manual

STM3210C-EVAL Common

STM3210C-EVAL

Modules

STM3210C_EVAL BSP User Manual

Exported Constants

STM3210C-EVAL Common

Modules

STM3210C_EVAL_LED

Define for STM3210C_EVAL board.

STM3210C_EVAL_BUTTON

STM3210C_EVAL_COM

STM3210C_EVAL_BUS

STM3210C_EVAL_COMPONENT

STM3210C_EVAL BSP User Manual

STM3210C_EVAL LCD

STM3210C-EVAL

Modules